WINE QUALITY DETECTION

Giacomo Galliano, s292482

Machine Learning and Pattern Recognition course at Politecnico di Torino
a.y. 2021/2022

1. Introduction

In this project we're going to perform a binary classification analysis to discriminate good and bad wines. The dataset taken in consideration is a modified version of the "Vinho verde" dataset from the UCI repository, which has been binarized collecting wines with low quality (with original scores from 0 to 6) into class 0 and good quality (original scores greater then 6) into class 1. There are 11 features, that represent physical properties of the wine.

1.1 Dataset analysis

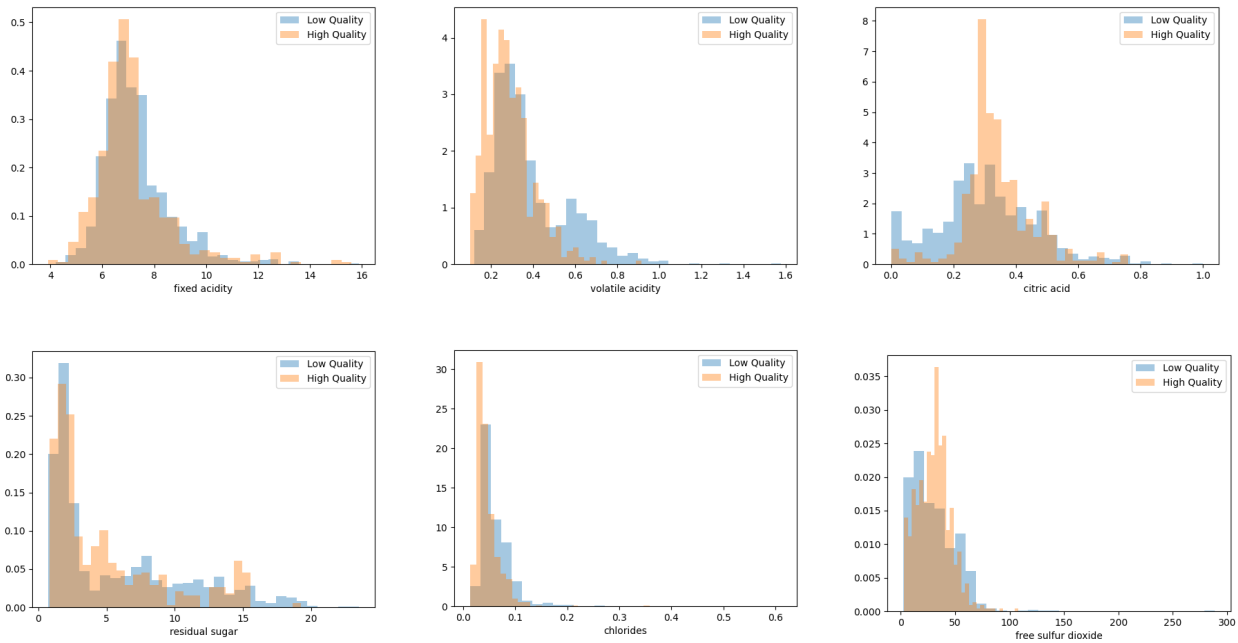
The first step of our analysis consists of splitting the dataset into training and test set; it is important to observe how those splits are made to have a better understanding of the data that we're working with.

The composition of the datasets is the following:

- **Training set**: **613** high_quality samples, **1226** low_quality samples
- **Test set**: **664** high_quality samples, **1158** low_quality samples

After that, we procede by analyzing the raw features, in order to observe the existence of irregular distributions, characterized by a presence of significantly large outliers. These could lead to sub-optimal results in classification approaches (especially Gaussianbased methods).

The 11 features are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.



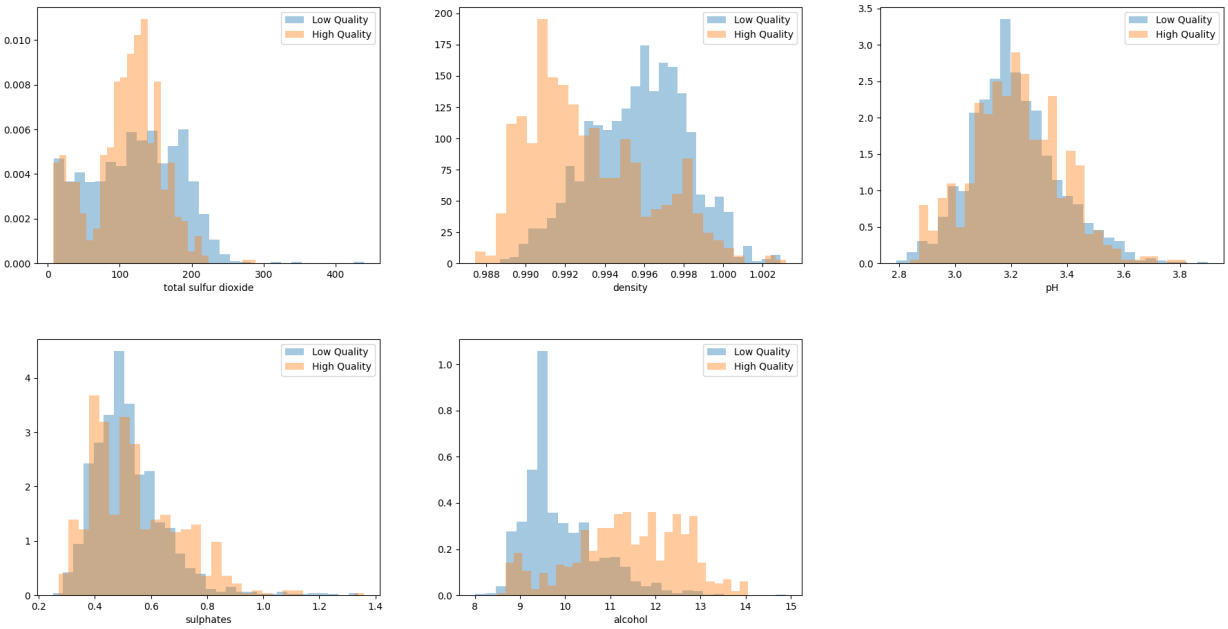
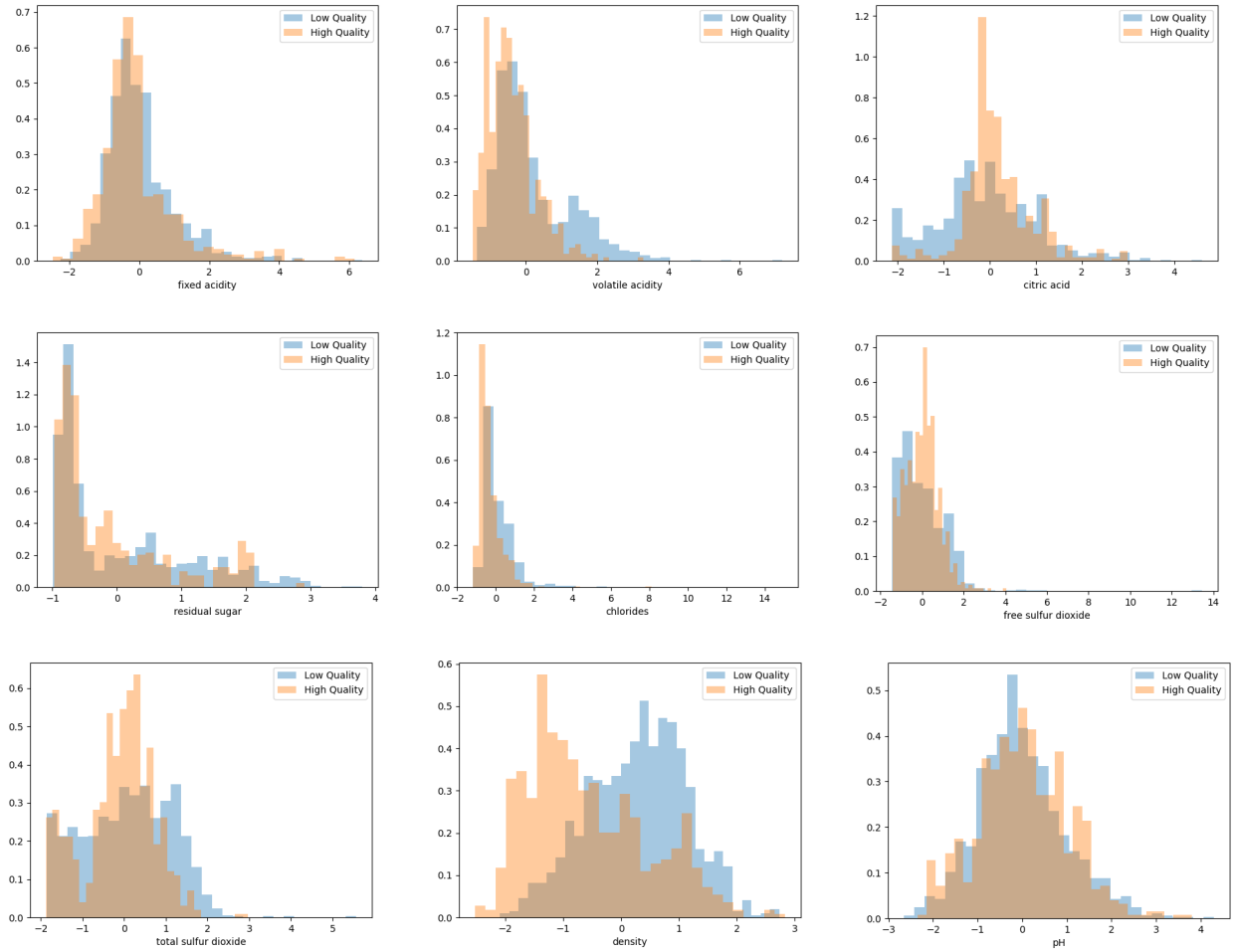


Fig. 1 *raw data features distribution.In orange high qulity wines, in blue low quality wines.*

To compare features between each other it's necessary to perform a z-normalization by centering and scaling the data.



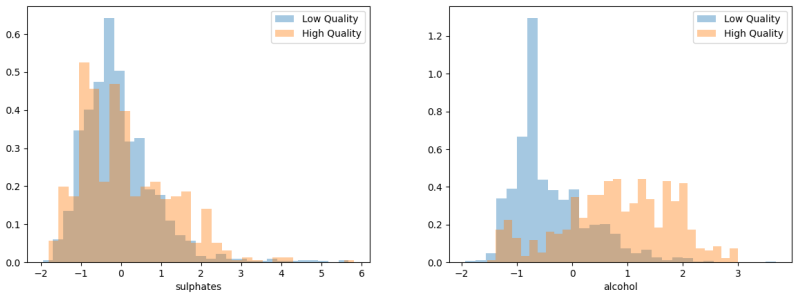


Fig. 2 *nomalized features distribution.In orange high qulity wines, in blue low quality wines.*

As we can see from the plots, the feature distribution is not "gaussian" and, as said before, we can notice the presence of outliers that could affect classification results.

We therefore further pre-process data by "Gaussianizing" the features. Gaussianization is a procedure that allows mapping a set of features to values whose empirical cumulative distribution function is well approximated by a Gaussian c.d.f.

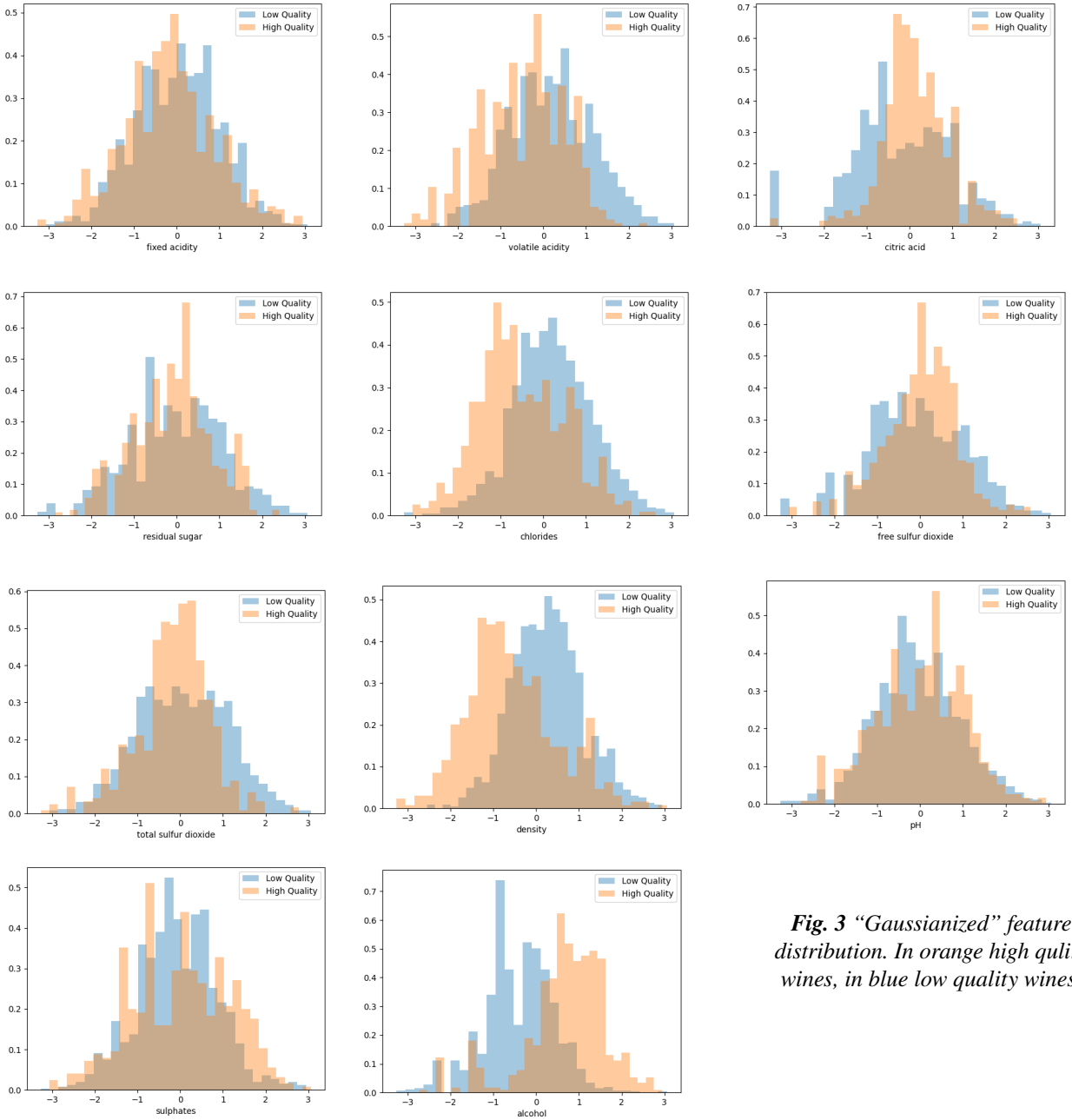


Fig. 3 *"Gaussianized" feature distribution. In orange high qulity wines, in blue low quality wines.*

We proceed our dataset analysis performing a correlation analysis of Gaussianized features using heatmaps.

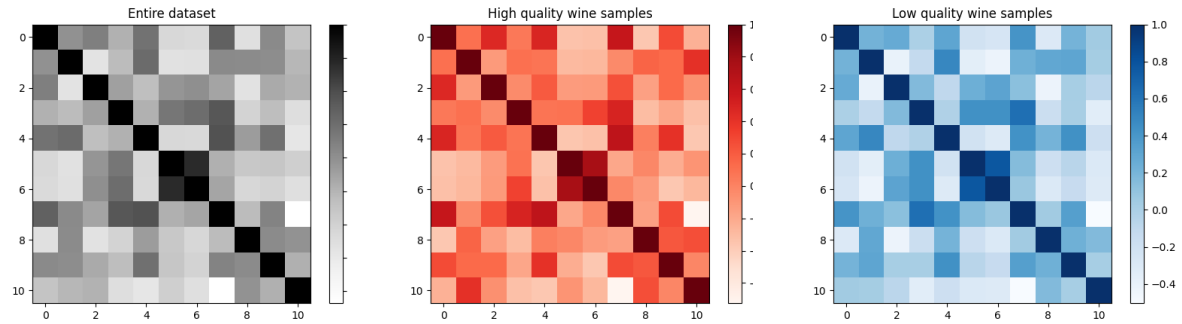


Fig. 4 *Heatmaps for "Gaussianized" features correlation analysis. Grey, entire dataset; Red, high quality wines; Blue, low quality wines.*

As we can see from the plots, features 5 and 6 are strongly correlated. This suggests we may benefit from using PCA to map data to 10 or less, uncorrelated features to reduce the number of parameters to estimate.

Heatmaps on raw data and heatmaps on normalized data are not reported here because they're very similar to the gaussinized ones. So, the same observations would be valid for those situations too.

2. Classification phase

2.1 Methodology

To understand which model is the most promising, and to asses the effects of applying PCA we could adopt two methodologies:

- Single-fold, in which the training set is splitted into development (80%, used to training the model) and validation (20%, used to test the model) subsets.
- K-Fold cross-validation, in which the training set is splitted in *k* chunks: k-1 are used to train the model and the remaining one to test it. This process Is repeated k times, selecting every time a different chunk for the validation part.

In this project we'll focus only on the second approach due to the fact that we don not have too much data and also for the resources at our disposal.

For what regards the model evaluation we want to measure performances in terms of normalized minimum detection costs, in other words, this is the cost we would pay if we made optimall decisions for the test set using the recognizer scores.

Our main application will be a uniform prior one

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1)$$

but we will also take into consideration unbalanced applications, when cases are biased towards high or low quality.

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1)\ , \quad (\tilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1)$$

2.2 MVG Classifier

We will first consider the Multivariate Gaussian classifier.

	5-Fold		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	Raw Features – no PCA		
Full - Cov	0.312	0.778	0.843
Diag - Cov	0.420	0.846	0.922
Tied Full - Cov	0.334	0.812	0.749
Tied Diag - Cov	0.403	0.866	0.932
	Gaussianized features – no PCA		
Full - Cov	0.307	0.785	0.790
Diag - Cov	0.449	0.834	0.914
Tied Full - Cov	0.355	0.803	0.884
Tied Diag - Cov	0.451	0.879	0.943
	Gaussianized features – PCA (m=10)		
Full - Cov	0.329	0.807	0.863
Diag - Cov	0.378	0.804	0.825
Tied Full - Cov	0.329	0.809	0.754
Tied Diag - Cov	0.334	0.822	0.782
	Gaussianized features – PCA (m=9)		
Full - Cov	0.320	0.800	0.814
Diag - Cov	0.378	0.802	0.815
Tied Full - Cov	0.329	0.846	0.752
Tied Diag - Cov	0.334	0.855	0.783

Table 1. *min DCF for MVG classifier with different values of* $\tilde{\pi}$

As we can see from the results in the table above, the *Full-Cov* is the one that performs best in k-fold cross validation, both with and without PCA on Raw and Gaussinaized features. We can also notice that PCA allow us to reduce the number of parameters, but this does not improve our results. Hower, given the fact that the results are pretty consistent between m=10 and m=9, we can affirm that PCA with m=9 don't lead to a reduction in terms of performances.

The diagonal covariance model do not give good results in any of the situation taken into consideration. The naive-bayes-assumption of uncorrelation between different components on which they rely on, does not hold in this case.

Other than that, we notice that the Tied models perform similarly to the Full-Cov, but slightly worse.

The gaussianization process improves performances over raw features only in the Full-Cov case; given that, we can say that gaussianization is not very helpful in this case.

2.3 Logistic Regression

We turn now our attention on discriminative approaches, focusing on Linear and Quadratic Logistic Regression.

Given the poor impact of applying PCA for generatuve models, we'll only consider raw and gaussianized data on the whole set of features.

2.3.1 Linear Logistic Regression

Since classes are not balanced, we re-balance the cost of the different classes so that we are actually training the logistic regression model for a target prior that is equal to 0.5. To do that, we take the average costs for each class and we multiply it for the prior of the corresponding class.

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

Now we have to perform the tuning of the λ parameter. We choose some values of λ on the logaritmic scale and then we see how the minDCF changes; then we will select the value that gives us best results. To reduce the risk of obtaining a too confident model, we decide to choose a λ value that gives us good results but that is not too small.

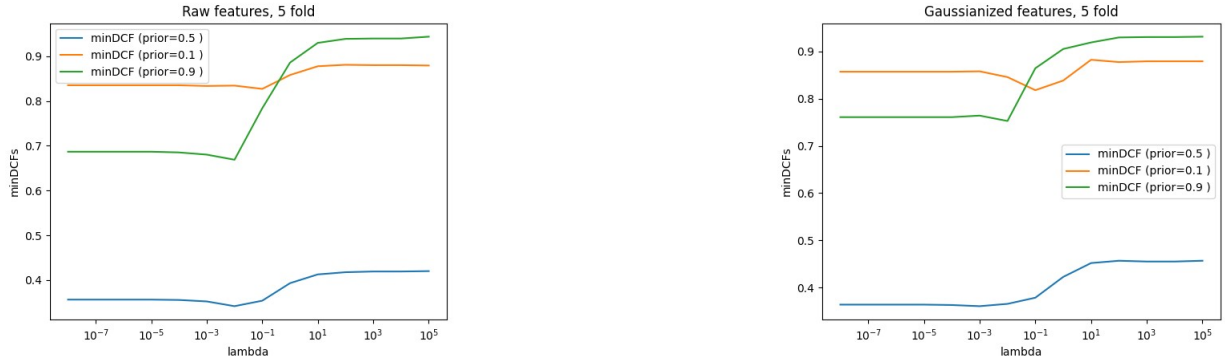


Fig. 5 *min DCF for Linear Logistic regression over different values of λ. Left raw features, right gaussianized features.*

As we can observe from the followig table, Gaussianization does not help to improve the results in this case, but it worsen them instead.

	5-Fold		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	Raw Features		
LR ($\lambda=10^{-3}$, π_T=0.5)	0.352	0.834	0.680
LR ($\lambda=10^{-3}$, π_T=0.1)	0.337	0.818	0.733
LR ($\lambda=10^{-3}$, π_T=0.9)	0.369	0.852	0.653
LR ($\lambda=10^{-3}$, π_{emp})	0.340	0.839	0.674
	Gaussianized features		
LR ($\lambda=10^{-3}$, π_T=0.5)	0.361	0.858	0.764
LR ($\lambda=10^{-3}$, π_T=0.1)	0.340	0.781	0.933
LR ($\lambda=10^{-3}$, π_T=0.9)	0.376	0.899	0.700
LR (λ=10-3, π_{emp})	0.360	0.830	0.837

Table 2. *Linear Logistic Regression results*

Taking into consideration different priors π_T we can see how this will affect the respective applications. The only case in which class rebalancing ha s a positive impact is with π_T=0.1, whereas in the other cases the unbalanced classes performed better

Comparing the results with the MVG classifier, we can say that overall logistic regression model performs slightly worse.

Given the fact that MVG corresponds to linear separation rules, and the results obtained with these first two classifiers are not that good, we can say that linear classification rules are not very good for this kind of data.

We repeat the analysis for Quadratic Logistic Regression.

2.3.2 Quadratic Logistic Regression

Following the previous procedure, we obatain these results.

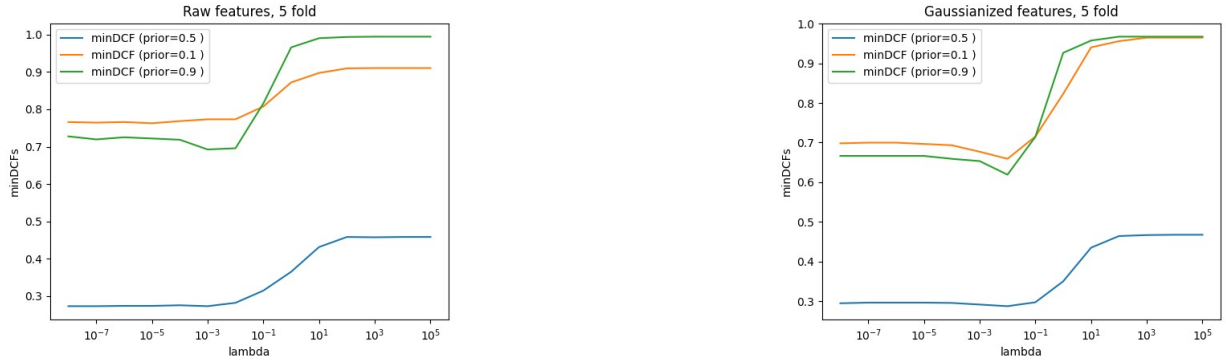


Fig. 6 *min DCF for Quadratic Logistic regression over different values of* λ*. Left raw features, right gaussianized features.*

As before, the selected value is $\lambda=10^{-3}$.

Taking a look at the results obtained with the Quadratic Logistic Regression, we see how this last model outperforms the MVG classifier, so this suggests that linear separation rule could be worse than the quadratic one for this specific task.

This time class rebalancing has basically zero effects on the target application.

Also in this case, Gaussianization does not improve the results with respect to the raw features.

	5-Fold		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	Raw Features		
LR ($\lambda=10^{-3}$, π_T=0.5)	0.273	0.772	0.692
LR ($\lambda=10^{-3}$, π_T=0.1)	0.273	0.752	0.704
LR ($\lambda=10^{-3}$, π_T=0.9)	0.288	0.806	0.643
LR ($\lambda=10^{-3}$, π_{emp})	0.273	0.769	0.687
	Gaussianized features		
LR ($\lambda=10^{-3}$, π_T=0.5)	0.291	0.677	0.653
LR ($\lambda=10^{-3}$, π_T=0.1)	0.293	0.701	0.644
LR ($\lambda=10^{-3}$, π_T=0.9)	0.308	0.746	0.626
LR ($\lambda=10^{-3}$, π_{emp})	0.289	0.677	0.637

Table 3. *Quadratic Logistic Regression results*

2.3 Support Vector Machines (SVM)

We proceed our analysis with the Support Vector Machine model, that allow us to perform non linear classification.

Also in this case we consider a re-balanced version of the model in which we use a different value of C for the different classes

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

subject to $0 \leq \alpha i \leq Ci\ , i = 1, \ldots, n$

where $C_i = C_T$ for samples of class H_T and $C_i = C_F$ for samples of class H_F .

$$C_T = C\frac{\pi_T}{\pi_T^{emp}} \text{ and } C_F = C\frac{\pi_F}{\pi_F^{emp}}$$

C_T and C_F have been selected, where π_T^{emp} and π_F^{emp} are the empirical priors for the two classes computed over the training set. The costs are proportional to the actual prior that we decided to use.

We'll analyze Linear SVM, SVM with quadratic kernel and SVM with RBF kernel.

2.3.1. Linear SVM

Linear SVMs are charachteized by the C parameter, which determines how wide the margin is, and how "strict" his behaviour is regarding the inside points.

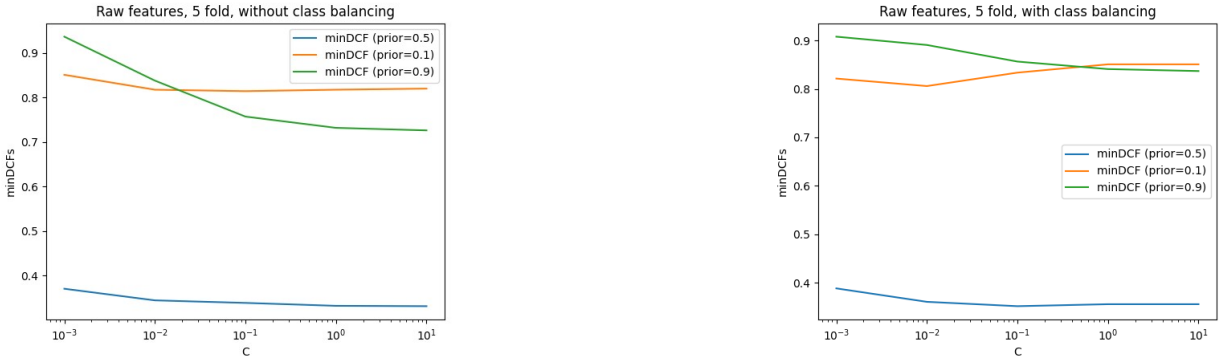


Fig. 7 *min DCF for different values of C. First, without class balancing, second, with class balancing*

We choose $C=10^{-1}$ because this will allow us to have a less confident solution but that generalizes better, so it should give us good performances.

	5-Fold		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	Raw Features		
SVM(C=0.1, π_T=0.5)	0.339	0.849	0.668
SVM(C=0.1, π_T=0.1)	0.892	1.000	0.997
SVM(C=0.1, π_T=0.9)	0.387	0.877	0.693
SVM(C=0.1, π_{emp})	0.338	0.814	0.757
	Gaussianized Features		
SVM(C=0.1, π_T=0.5)	0.351	0.834	0.856
SVM(C=0.1, π_T=0.1)	0.516	0.953	0.996
SVM(C=0.1, π_T=0.9)	0.397	0.954	0.670
SVM(C=0.1, π_{emp})	0.345	0.772	0.943

Table 4. *Linear SVM results*

Also in this case we do not get great performances. At this point we can confirm that linear-model do not perform well on this dataset, so it's better to shift our focus on non-linear SVM formulations.
Again, balancing the SVM does not bring particular benefits for this specific dataset.

2.3.2 SVM with quadratic kernel

We'll use a polynomial quadratic kernel, similar to the quadratic Logistic Regression model, so we expect similar results.

$$k(x_1, x_2) = (x_1^T x_2 + c)^d \qquad \hat{k}(x_1, x_2) = k(x_1, x_2) + K^2$$

In this case, *c* represents an hyperparameter obatained through cross-validation and *K* represents a costant value added to the kernel function to add a regularized bias to the non-linear SVM.

To estimate these values, we looked at the results obatined trying different numbers. As we can see from the following graph, the best values are *c=1*, *K=0* with *C=0.1* as before.

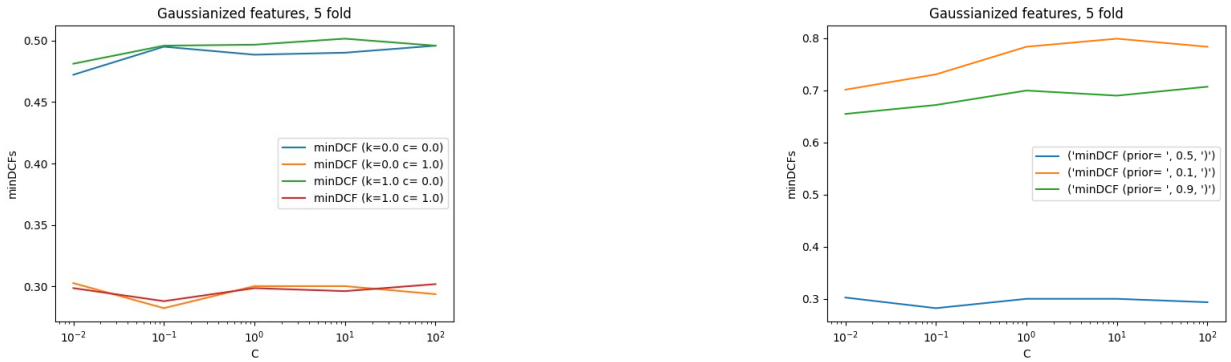


Fig. 8 *First, results for different values of c and K, min DCF for different values of C.*

In the following table we can see the results obatained with the Quadratic kernel SVM.

	5-Fold		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	Raw Features		
SVM Quad (K=0, c=1, C=0.1, π_T=0.5)	0.273	0.799	0.692
SVM Quad (K=0, c=1, C=0.1, π_T=0.1)	0.892	1.000	0.997
SVM Quad (K=0, c=1, C=0.1, π_T=0.9)	0.387	0.877	0.693
SVM Quad (K=0, c=1, C=0.1, π_{emp})	0.338	0.814	0.757
	Gaussianized features		
SVM Quad (K=0, c=1, C=0.1, π_T=0.5)	0.282	0.731	0.672
SVM Quad (K=0, c=1, C=0.1, π_T=0.1)	0.516	0.953	0.996
SVM Quad (K=0, c=1, C=0.1, π_T=0.9)	0.397	0.954	0.670
SVM Quad (K=0, c=1, C=0.1, π_{emp})	0.346	0.772	0.943

Table 5. *SVM with quadratic kernel results*

Looking at the table we can confirm that quadratic surfaces are more suitable to discriminate classes of this dataset. Again, gaussinaization does not improve the performances. We can also see that the balnced version of the model provides better results.
Comparing these results with the ones obtained with the Quadratic Logistic Regression we can see that they are very similar.

2.3.2 SVM with RBF kernel

In this case, the kernel function is represented by

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

Here, γ corresponds to the kernel width; larger values lead to narrow kernel, while lower values to a wider one.
We have to tune both γ and C. They both influence the results of the minDCFs.

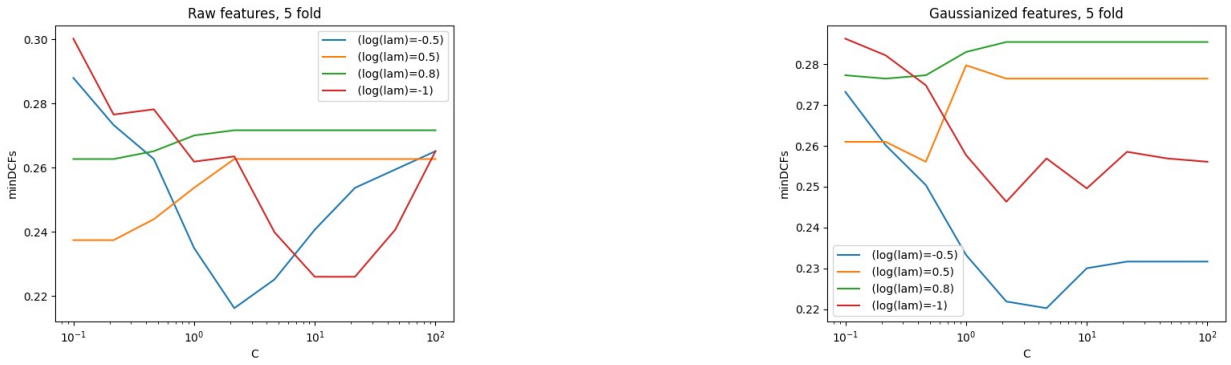


Fig. 9 *minDCF for raw features and gaussianized features with different γ*

Aiming to optimize as much as possible both parameters, we choose $\gamma=10^{-0.5}$, and $C=10^{0.1}$.

In the following table we can see the results of the SVM with RBF kernel model.

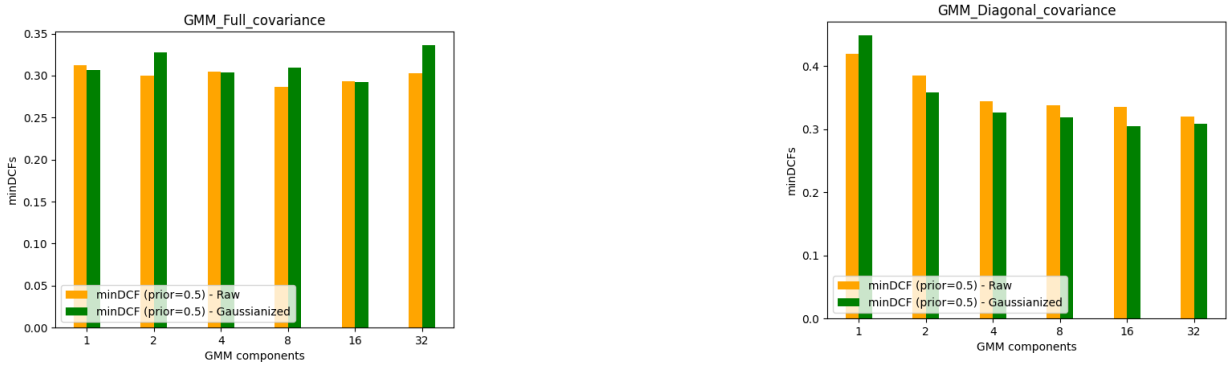
	5-Fold		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	Raw Features		
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.5)	0.229	0.573	0.610
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.1)	0.299	0.595	0.880
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.9)	0.261	0.764	0.583
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_{emp})	0.242	0.570	0.774
	Gaussianized features		
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.5)	0.231	0.524	0.615
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.1)	0.303	0.575	0.820
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.9)	0.242	0.692	0.607
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_{emp})	0.232	0.539	0.657

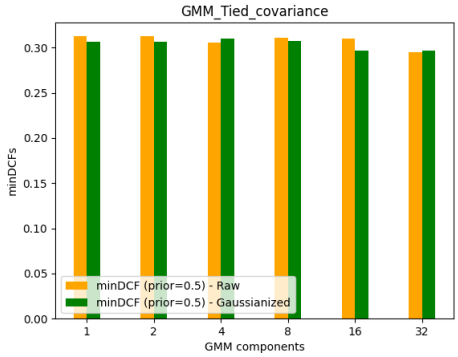
Table 6. *SVM with RBF kernel results*

This model provides better results than previous model for both balnced and unbalnced scenarios. We can also notice that class re-balance affects positively the performances, even if it is a minimal improvement.
Is needed to notice that in this case the model performs better on raw features than on gaussianized ones after class rebalancing.

2.4 Gaussian Mixture Model (GMM)

This model is able to approximate generic distributions by assuming that all the data points are generated from a finite number of Gaussian distributions.
We will consider both full covariance and diagonal models, with and without covariance tying.
Again we perform an analysis of the *minDCF* over raw and gaussianized features changing the number of components for each GMM.





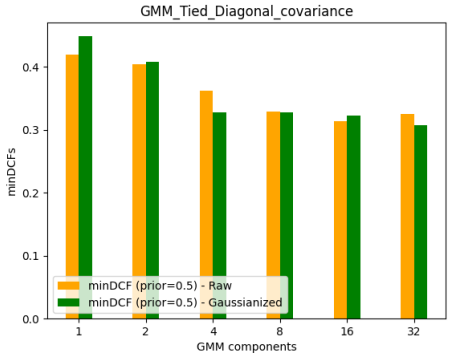


Fig. 10 *minDCF for raw features and gaussianized features with different number of components for each GMM. Full_Cov, Diag_Cov, Tied_Full_Cov, Tied_Diag_Cov*

The behaviour between raw features and gaussianized feature is more or less the same.

# features	1	2	4	8	16	32
			Raw Features			
Full Cov	0.310	0.300	0.305	0.286	0.293	0.303
Diag Cov	0.420	0.385	0.344	0.338	0.335	0.320
Tied Full Cov	0.312	0.312	0.306	0.311	0.310	0.295
Tied Diag Cov	0.420	0.405	0.362	0.329	0.314	0.325
			Gaussianized features			
Full Cov	0.307	0.328	0.303	0.310	0.292	0.336
Diag Cov	0.449	0.358	0.327	0.319	0.305	0.309
Tied Full Cov	0.307	0.307	0.310	0.308	0.297	0.297
Tied Diag Cov	0.449	0.408	0.328	0.328	0.323	0.308

Table 7. *GMM results*

Looking at the table we can see how the best performing model is the Full Covariance GMM with 8 components trained on raw features; also with the gaussianized features is the one that has better performances, even tough these aren't as good as the model trained on raw features.
Observing the tied models, the best performing one is the Tied Full Covariance model, both on raw and gaussianized features, with similar results.
These results are similar to the one obrained with the SVM with quadratic kernel but not as good as the ones with the SVM with RBF kernel.

2.5 Aactual DCF analysis

Up to now we have considered only minimum DCF metrics. *Min DCF* 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 goodness of the decisions we make using those scores (in the binary case, on the goodness of the threshold we use in practice to perform class assignment).
We therefore turn our attention to actual DCFs.

We have seen that, if scores are well calibrated, the optimal threshold that optimizes the Bayes risk is

$$t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

Taking the two best perfoming models, we compare the values in terms of minDCF and actDCF in order to understand if our classifiers are already well calibrated or if we need to re-calibrate our scores or to find for each application a good threshold.

	$\tilde{\pi}$=0.5		$\tilde{\pi}$=0.1		$\tilde{\pi}$=0.9	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
SVM RBF	0.229	0.234	0.573	0.993	0.610	0.983
SVM Quad	0.273	0.297	0.799	0.852	0.692	0.750

Table 8. *minDCF vs actDCF*

The table results shows that the models are both quite well calibrated for the balanced scenario. In the unbalanced cases we can notice a greater miscalibration, especially in the SVM with RBF kernel model.
This could be due to overfitting or underfitting a bit over some area of the feature space.
The model is able to discriminate between the two classes but if we have to use the theoretical threshold we'll obtain bad results.
These results can be plotted into a Bayes Error Plot.

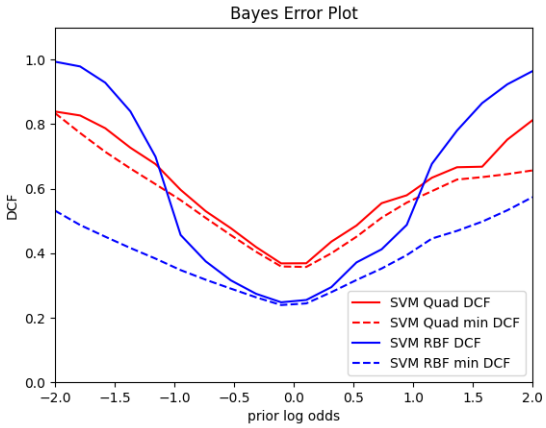


Fig. 11 *Bayes error plot to compare minDCF and actDCF*

Looking at the graph we can see in a better way that the models (especially the *SVM RBF*) are well calibrated mainly in the balanced situation. We notice very huge loss for RBF SVM in unbalanced scenarios, while a noticeable loss for Quadratic SVM with negative values.

To address the problem of miscalibration, we estimate an application dependent threshold. Given a target application in mind, we estimate a "close-to optimal" threshold.
We consider computing the minDCF for that particular application, and then we can just take the corresponding threshold of the min DCF on the validation set.
This will not produce well-calibrated scores for a different number of applications; if we change application, we'll need to perform again the process.

	minDCF	actDCF	est. t DCF
		$\tilde{\pi}$=0.5	
SVM RBF	0.229	0.234	0.249
SVM Quad	0.273	0.297	0.393
		$\tilde{\pi}$=0.1	
SVM RBF	0.573	0.993	0.597
SVM Quad	0.799	0.852	0.970
		$\tilde{\pi}$=0.9	
SVM RBF	0.610	0.983	0.733
SVM Quad	0.692	0.750	0.702

Table 9. *minDCF vs actDCF vs est. t DCF*

We can see from the table that we get more or less the same calibration in the case of the balanced scenario, while an improvement is noticeable where the miscalibration was higher.

2.6 Classification phase observations

At the end of this phase of our analysis we can mark as the best model for our dataset the *SVM with RBF kernel.*

3. Experimental results

We will now verify the quality of the models on unseen data in the balanced scenario.

3.1 MVG Classifiers

	5-fold
	$\tilde{\pi}$=0.5
Raw Features – no PCA	
Full - Cov	0.345
Diag - Cov	0.374
Tied Full - Cov	0.320
Tied Diag - Cov	0.369
Gaussianized features – no PCA	
Full - Cov	0.348
Diag - Cov	0.387
Tied Full - Cov	0.333
Tied Diag - Cov	0.387
Gaussianized features – PCA (m=10)	
Full - Cov	0.569
Diag - Cov	0.550
Tied Full - Cov	0.326
Tied Diag - Cov	0.328
Gaussianized features – PCA (m=9)	
Full - Cov	0.325
Diag - Cov	0.336
Tied Full - Cov	0.325
Tied Diag - Cov	0.325

Table 10. *min DCF for MVG classifier*

We can observe that in this case, the best results are obtained with *Tied Full–Cov* classifier, even if the difference with the *Full-Cov* is not that huge.
Also in this case we can see that PCA doesn't have a huge impact, both in positive or negative way.

3.2 Logistic Regression

3.2.1 Linear Logistic Regression

	5-fold
Raw Features	
LR ($\lambda=10^{-3}$, $\pi_T=0.5$)	0.338
LR ($\lambda=10^{-3}$, $\pi_T=0.1$)	0.318
LR ($\lambda=10^{-3}$, $\pi_T=0.9$)	0.350
LR ($\lambda=10^{-3}$, π_{emp})	0.340
Gaussianized features	
LR ($\lambda=10^{-3}$, $\pi_T=0.5$)	0.345
LR ($\lambda=10^{-3}$, $\pi_T=0.1$)	0.311
LR ($\lambda=10^{-3}$, $\pi_T=0.9$)	0.367
LR ($\lambda=10^{-3}$, π_{emp})	0.360

Table 11. *Linear Logistic Regression results - eval*

In this case the results can be considered consitent. The effect of Gaussianization remains pretty much irrelevant.

3.2.1 Quadratic Logistic Regression

5-fold	
Raw Features	
LR ($\lambda=10^{-3}$, $\pi_T=0.5$)	0.253
LR ($\lambda=10^{-3}$, $\pi_T=0.1$)	0.264
LR ($\lambda=10^{-3}$, $\pi_T=0.9$)	0.259
LR ($\lambda=10^{-3}$, π_{emp})	0.254
Gaussianized features	
LR ($\lambda=10^{-3}$, $\pi_T=0.5$)	0.304
LR ($\lambda=10^{-3}$, $\pi_T=0.1$)	0.303
LR ($\lambda=10^{-3}$, $\pi_T=0.9$)	0.311
LR ($\lambda=10^{-3}$, π_{emp})	0.312

Table 12. *Quadratic Logistic Regression results – eval*

In this case the results are consistent, but this time the best result is obtained in both cases in the class balanced situation (even if the difference is minimal). Important to notice the very good score on raw features.

3.3 Support Vector Machines (SVM)

3.3.1 Linear SVM

5-fold	
Raw Features	
SVM(C=0.1, $\pi_T=0.5$)	0.335
SVM(C=0.1, $\pi_T=0.1$)	0.821
SVM(C=0.1, $\pi_T=0.9$)	0.357
SVM(C=0.1, π_{emp})	0.313
Gaussianized Features	
SVM(C=0.1, $\pi_T=0.5$)	0.333
SVM(C=0.1, $\pi_T=0.1$)	0.447
SVM(C=0.1, $\pi_T=0.9$)	0.394
SVM(C=0.1, π_{emp})	0.307

Table 13. *Linear SVM results – eval*

Also in this case the results are in line with our expectations, with re-balanced scenarios that perform slightly worse.

3.3.2 SVM with quadratic kernel

5-fold	
Raw Features	
SVM Quad (K=0, c=1, C=0.1, $\pi_T=0.5$)	0.271
SVM Quad (K=0, c=1, C=0.1, $\pi_T=0.1$)	0.305
SVM Quad (K=0, c=1, C=0.1, $\pi_T=0.9$)	0.283
SVM Quad (K=0, c=1, C=0.1, π_{emp})	0.275
Gaussianized features	
SVM Quad (K=0, c=1, C=0.1, $\pi_T=0.5$)	0.314
SVM Quad (K=0, c=1, C=0.1, $\pi_T=0.1$)	0.316
SVM Quad (K=0, c=1, C=0.1, $\pi_T=0.9$)	0.311
SVM Quad (K=0, c=1, C=0.1, π_{emp})	0.304

Table 14. *SVM with Quadratic kernel results – eval*

Here the difference between class re-balncing and not are less evident than in the training phase.

3.3.3 SVM with RBF kernel

	5-fold
	Raw Features
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.5)	0.259
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.1)	0.387
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.9)	0.268
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_{emp})	0.298
	Gaussianized features
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.5)	0.269
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.1)	0.370
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_T=0.9)	0.270
SVM RBF ($\gamma=10^{-0.5}$, $C=10^{0.1}$, π_{emp})	0.298

Table 15. *SVM with RBF kernel results – eval*

The results are once again consistent and in this case the SVM with RBF kernel confirms itself as one of the best perfoming model.

3.4 Gaussian Mixture Models (GMM)

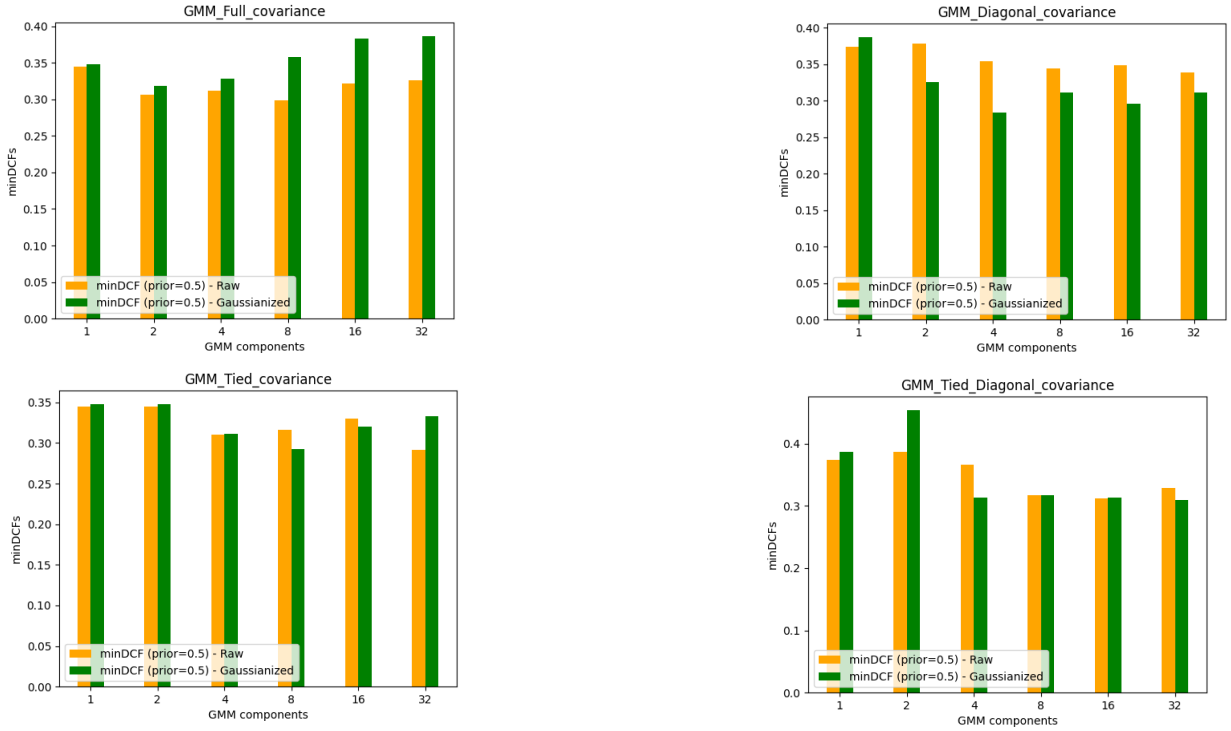


Fig. 12 *minDCF for raw features and gaussianized features with different number of components for each GMM. Full_Cov, Diag_Cov, Tied_Full_Cov, Tied_Diag_Cov - eval*

# features	1	2	4	8	16	32
			Raw Features			
Full Cov	0.345	0.306	0.312	0.299	0.321	0.326
Diag Cov	0.374	0.379	0.354	0.344	0.349	0.339
Tied Full Cov	0.345	0.345	0.310	0.316	0.330	0.292
Tied Diag Cov	0.374	0.386	0.366	0.317	0.312	0.329
			Gaussianized features			
Full Cov	0.348	0.318	0.330	0.358	0.383	0.386
Diag Cov	0.387	0.326	0.284	0.312	0.296	0.311
Tied Full Cov	0.348	0.348	0.311	0.292	0.320	0.333
Tied Diag Cov	0.387	0.453	0.314	0.317	0.313	0.310

Table 16. *GMM results*

Here we can notice how on raw features the best performance is obtained by the Full Cov with 8 components, while, in this case, for gaussianized features the best performing one is the Diag Cov with 4 components.

3.5 Aactual DCF analysis

	minDCF	actDCF
SVM RBF	0.259	0.287
SVM Quad	0.271	0.301

Table 17. *minDCF vs actDCF*

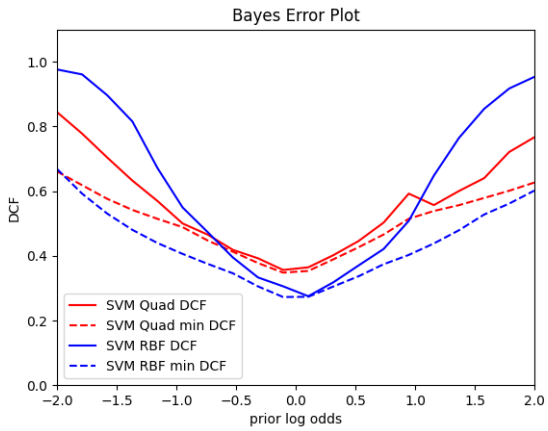


Fig. 13 *Bayes error plot to compare minDCF and actDCF - evaluation*

As we can see from the table results and the plot, the scores that we've obtained for the target application are quite good calibrated. Given that we can affirm that re-calibration is not needed in this case.

4. Conclusion

In this project we analyzed the Wine Quality dataset applying different models.
The results tell us that the best model for the given dataset is the *Support Vector Machine with the RBF kernel*, followed by the *Quadratic Logistic Regression* that also provides good results.
Given that, we observed that linear models are not suited for this type of dataset.
If we compare the results obtained on the validation and the evaluation sets, we can say that the decision we took were correct in order to obtain results in line with our expectations.