Classification of the UCI Wine Quality Data Set

Massimiliano Pronesti

Politecnico di Torino
s287646@studenti.polito.it

Abstract

This report provides an analysis of the effectiveness of different classification approaches applied to the popular wine quality prediction problem on the wine dataset from the UCI repository. Specifically, the goal is to predict whether the wine has a good or a bad quality, given a set of features related to its chemical composition, such as the pH and the percentage of free sulfur dioxide. The original dataset consists of 10 classes, however, for this work, the dataset has been binarized, collecting all wines with low quality (lower than 6) into class 0, and good quality (greater than 6) into class 1, while those with quality 6 have been discarded. In addition, the dataset contains both red and white wines (merged for the sake of this analysis). There are 11 features, that represent physical properties of the wine, with partially balanced classes.

I. Data Analysis

In this section, we are going to conduce an analysis on the main characteristics of the features contained in the training dataset. The training set consists of 1126 bad quality samples and 613 good quality samples, then one class is twice as much present as the other. A visualization of how raw features are distributed is shown in Figure 1.

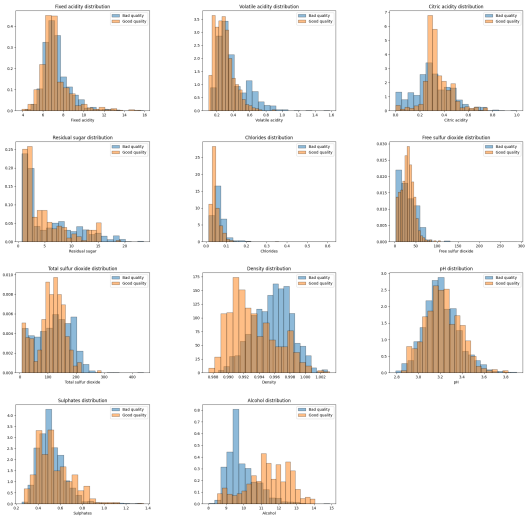


Figure 1: *Raw features distribution of the UCI Wine Quality Dataset*

We can observe that features don't have a zero mean, therefore we might consider standardizing them, i.e. centering data and scaling it by the variance, so that the obtained random variable has zero mean and unitary variance.

In addition, features don't expose a Gaussian trend, with the presence of some outliers.

Therefore, we gaussianize the features, computing the cumulative rank $r(x)$ over the training set

$$r(x) = \frac{\sum_{i=1}^{N} I[x < x_i] + 1}{N + 2}$$

being x_i the value of the considered feature for the i-th sample, and transforming the features computing the inverse of the cumulative distribution function Φ fed with the rank $r(x)$

$$X_{gauss} = \Phi^{-1}(r(x))$$

The distribution of the gaussianized features is shown in Figure 2.

Moreover, we provide an analysis of the correlation between the features, exploiting the Pearson product-moment correlation coefficient, defined as

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

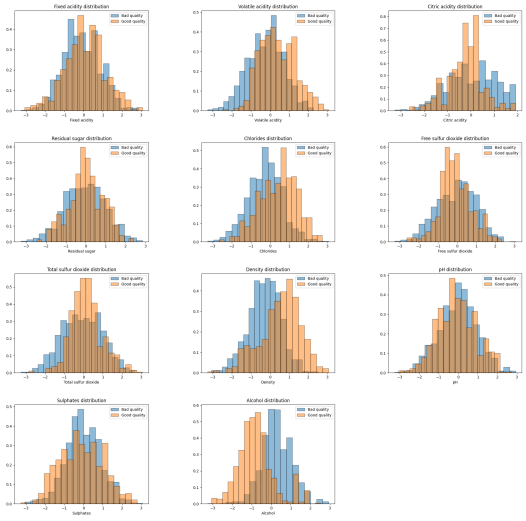


Figure 2: *Gaussianized features distribution of the UCI Wine Quality Dataset*

being $cov(X, Y)$ the covariance matrix of X and Y, expressible as the expectation of the product of X and Y centered using their respective mean.

$$cov(X, Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$$

The obtained heatmaps among the gaussianized features are shown in Figure 3.

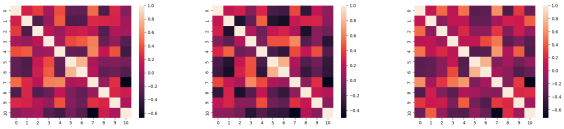


Figure 3: *Heatmaps among Gaussianized features*

where we can observe some features are correlated (darker colors in the heatmap), thus exploring dimensionality reduction techniques such as the PCA, could prove beneficial.

II. Method

In this section we are going to first describe the approach followed towards model selection and model evaluation. Then, we will analyse the explored classification methods and the results they yielded.

II.I. Approach

In order to perform our analysis, we will adopt two approaches towards model selection:

- **single split**: the training set is divided into two chunks, where the 80% of the samples are used for fitting the classifer and the remaining 20% for testing it.
- **k-fold cross validation**: the training set is split into k folds, one of whom is used for validation and the other $k-1$ for fitting the model. The process is repeated k times. This approach usually makes the process of model selection more reliable as, one by one, all the chunks will be used as unseen data. For this specific application, we set $k = 5$, i.e. we used 5 folds.

In both cases, we make sure no transformation is applied on the whole training data before splitting it, **not** to **bias** the validation and introduce **data leakages**.

As regards model evalution, we want to be Bayesian and adopt the minimum of the normalized Bayesian risk as metric, which measures the cost we would pay if we made optimal decisions using the recognizer scores. The application of interest is a uniform prior one

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

being $\tilde{\pi}$ the (unbiased, in the specified case) prior, C_{fp}, C_{fn} the costs of the false positive and false negative case, respectively. This is the case we will analyze extensively. However, we also consider biased cases towards one of the qualities

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

II.II. Gaussian Classifiers

Gaussian classifiers are the first class of methods we take into considerations. In particular we analyse the performances yielded by a Multivariate Gaussian and a Naive Bayes, both

with full and tied covariance, for a total of 4 classifiers.

Table 1 shows the results obtained for these models both for the single split and for the k-fold cross validation on raw and gaussianized data and applying PCA.

	single-split		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
Full-Cov	0.702	0.264	0.802
Diag-Cov	0.822	0.409	0.941
Tied Full-Cov	0.754	0.327	0.706
Tied Diag-Cov	0.820	0.391	0.908
Gaussianized features			
Full-Cov	0.761	0.255	0.729
Diag-Cov	0.840	0.460	0.866
Tied Full-Cov	0.726	0.344	0.797
Tied Diag-Cov	0.834	0.441	0.928
Gaussianized features, PCA(m=10)			
Full-Cov	0.726	0.242	0.655
Diag-Cov	0.707	0.372	0.789
Tied Full-Cov	0.709	0.359	0.698
Tied Diag-Cov	0.709	0.362	0.710
Gaussianized features, PCA(m=9)			
Full-Cov	0.740	0.253	0.698
Diag-Cov	0.720	0.384	0.790
Tied Full-Cov	0.719	0.361	0.731
Tied Diag-Cov	0.726	0.361	0.746
	5-fold		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
Full-Cov	0.781	0.312	0.842
Diag-Cov	0.847	0.420	0.922
Tied Full-Cov	0.810	0.334	0.746
Tied Diag-Cov	0.866	0.402	0.932
Gaussianized features			
Full-Cov	0.799	0.301	0.773
Diag-Cov	0.848	0.440	0.872
Tied Full-Cov	0.791	0.348	0.856
Tied Diag-Cov	0.869	0.441	0.940
Gaussianized features, PCA(m=10)			
Full-Cov	0.789	0.290	0.715
Diag-Cov	0.819	0.389	0.874
Tied Full-Cov	0.794	0.352	0.802
Tied Diag-Cov	0.793	0.351	0.822
Gaussianized features, PCA(m=9)			
Full-Cov	0.812	0.304	0.716
Diag-Cov	0.819	0.392	0.825
Tied Full-Cov	0.810	0.351	0.791
Tied Diag-Cov	0.809	0.353	0.813

Table 1: *min DCF for Gaussian models for different values of $\tilde{\pi}$*

We notice that the full-covariance Multivariate Gaussian classifier yields the better performances, both with and without PCA. In particular, the dimensionality reduction is beneficial when applyied with 10 components and doesn't degrade the performances with 9.

This suggests the dataset is large enough to estimate a covariance matrix for each class and loosening our assumptions (introducing diagonalization or tying) is not needed. This is furtherly suggested by the fact that the diagonal covariance models yield the worst results, as the uncorrelation hypotesis does not hold.

Eventually, it should be noted that, as expected, gaussianization improves the performances with respect to raw data. The results obtained for the single split and 5-fold are consistent to one another, conveying that, in this specific scenario, the size of our training set is appropriate even for a single fold evaluation (i.e. a single split).

II.III. Logistic Regression

In this section, we assess the performances of a Logistic Regression classifier, both with linear and quadratic kernel.

Being classes unbalanced, their costs are rebalanced using a loss function accounting for the number of samples for each class, used as weights for the two terms deriving from the split of the summation of the original loss

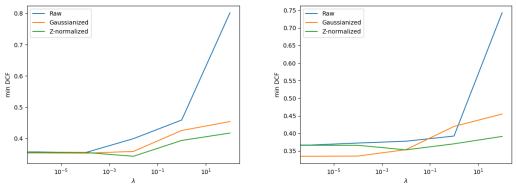
$$J(w,b) = \frac{\lambda}{2}\|w\|^2 + \frac{\pi_T}{n_T}\sum_{i|c_i=1} log\ \sigma(y_i)^{-1} + \frac{1-\pi_T}{n_F}\sum_{i|c_i=0}^{n} log\ \sigma(y_i)^{-1}$$

being $\sigma(.)$ the sigmoid function, y_i the output of the model $y_i = -z_i(w^T x_i + b)$, z_i a variable equal to ± 1 depending on the class, w, b the parameters of the model.

The most significant results obtained for both the models for different values of λ are shown in Table 2, while a complete visualization of both the models for the balanced prior is shown in Figure 4.

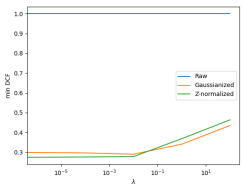
	single-split		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
Log-Reg ($\lambda = 10^{-4}$)	0.743	0.364	0.677
Quad-LR ($\lambda = 0$)	1.000	1.000	0.900
Z-normalized features			
Log-Reg($\lambda = 10^{-2}$)	0.753	0.353	0.754
Quad-LR($\lambda = 0$)	1.000	0.237	0.488
Gaussianized features			
Log-Reg($\lambda = 0$)	0.758	0.335	0.724
Quad-LR($\lambda = 10^{-2}$)	0.560	0.256	0.395
Z-normalized features, PCA(m=10)			
Log-Reg($\lambda = 10^{-2}$)	0.760	0.353	0.752
Quad-LR($\lambda = 0$)	0.601	0.243	0.400
	5-fold		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
Log-Reg ($\lambda = 10^{-4}$)	0.826	0.355	0.670
Quad-LR ($\lambda = 0$)	1.000	1.000	1.000
Z-normalized features			
Log-Reg($\lambda = 10^{-2}$)	0.834	0.343	0.670
Quad-LR($\lambda = 0$)	0.764	0.273	0.687
Gaussianized features			
Log-Reg($\lambda = 0$)	0.825	0.354	0.727
Quad-LR($\lambda = 10^{-2}$)	0.679	0.289	0.590
Z-normalized features, PCA(m=10)			
Log-Reg($\lambda = 10^{-2}$)	0.834	0.348	0.670
Quad-LR($\lambda = 0$)	0.759	0.280	0.678

Table 2: *min DCF for Logistic Regression models*

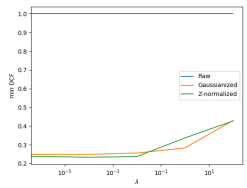


(a) *linear, 5-fold*

(b) *linear, single split*



(c) *quadratic, 5-fold*



(d) *quadratic, single split*

Figure 4: *min DCF for Linear and Quadratic Logistic Regression over different values of λ*

Differently from the previous case, Gaussianization doesn't prove this helpful, however standardizing data improved the performances. On the other hand, PCA does not bring any advantage, even if it doesn't prove detrimental.

Overall, among the different values of λ and the two models in the 3 different scenario analysed, the quadratic logistic regression with $\lambda = 0$ yields the best results. This is, to some extent, expected: quadratic models are characterized by a "curved" surface, which usually allow to better describe and learn the real distribution of the data, without overfitting.

II.IV. Support Vector Classifier

In this section, we test the performances of different flavours of support vector classifiers. In particular, we are going to use a linear kernel, a quadratic polynomial kernel and a radial basis function (RBF) kernel.

	Single split	**5-fold**
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.5$
Raw features		
SVC($C = 0.1$)	0.816	0.760
SVC($C = 0.1, \pi_T^{emp}$)	0.695	0.920
Z-normalized features		
SVC($C = 1$)	0.334	0.341
SVC($C = 0.1, \pi_T^{emp}$)	0.338	0.348
Gaussianized features		
SVC($C = 0.1$)	0.339	0.350
SVC($C = 0.1, \pi_T^{emp}$)	0.352	0.360

Table 3: *min DCF for Support Vector Classifier models, with and without feature balancing.*

As done for the previous class of models, we will also take into account the possibility of rebalancing the classes introducing an empirical prior π_T^{emp} over the training set, thus using two different values of C for the bad and good quality wines

$$C_T = C\frac{\pi_T}{\pi_T^{emp}} \qquad C_F = C\frac{1-\pi_T}{1-\pi_T^{emp}}$$

We will first conduct our analysis emplying the linear SVC, even if we already discussed above that linear models don't perform that good for our classification task.

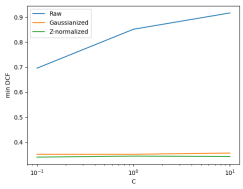
The best results obtained for the linear kernel are reported in Table 3, where we specified π_T^{emp} where we balanced the features using the

empirical prior. We observe that standardization improved greatly the results with respect to the raw case, where the classifier performs poorly both for the single split and the 5-fold approach. Conversely, Gaussianization and features balancing don't prove advantageous. A complete visualization is provided in Figure 5.

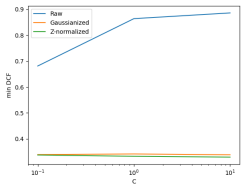
A further analysis is, now, conducted using a polynomial kernel of grade 2 (quadratic) and a radial basis function (RBF) kernel. We tune again C for both the cases and for the RBF kernel we perform a grid search involving γ as well, to select the best hypermameters. Overall, we try the following values:

$$C = \{10^{-1},\ 1,\ 10\}$$
$$\gamma = \{1/e,\ 1/e^2\}$$
$$\pi_T = \{0.5,\ \pi_T^{emp}\}$$

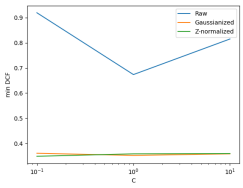
Place notice that π_T is included above, even if it's not an hyperparameter, only to convey the idea that we're searching the best combination of parameters and, at the same time, assessing the benefits of rebalancing features, if any. A visualization of the grid is shown in Figure 6.



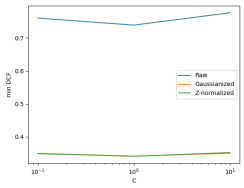
(a) *balancing, 5-fold*



(b) *no balancing, 5-fold*



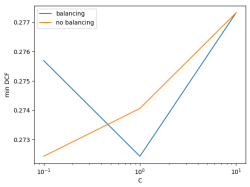
(c) *balancing, single split*



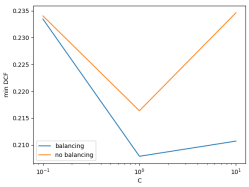
(d) *no balancing, single split*

Figure 5: *min DCF for Linear SVC with and without features balancing, over different values of C*

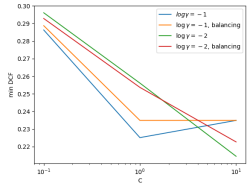
We can notice that, for the RBF SVC, we obtain similar results regardless of features balancing. Therefore, balancing is not beneficial nor detrimental in our specific case. Table 4 shows the results of the 2nd degree polynomial kernel and RBF kernel for the best choices of the hyperparameters, both with and without balancing.



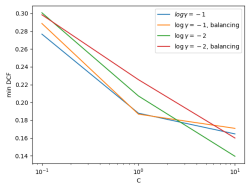
(a) *quadratic SVC, 5-fold*



(b) *quadratic SVC, single split*



(c) *RBF SVC, 5-fold*



(d) *RBF SVC, single split*

Figure 6: *min DCF for Quadratic and RBF SVC*

	Single split $\tilde{\pi} = 0.5$	**5-fold** $\tilde{\pi} = 0.5$
Z-normalized features		
Poly-SVC($C = 1$)	0.208	0.272
Poly-SVC($C = 0.1, \pi_T^{emp}$)	0.234	0.272
RBF-SVC($C = 10, log\gamma = -2$)	0.140	0.215
RBF-SVC($C = 10, log\gamma = -2, \pi_T^{emp}$)	0.160	0.223

Table 4: *min DCF for Support Vector Classifier models, with and without feature balancing.*

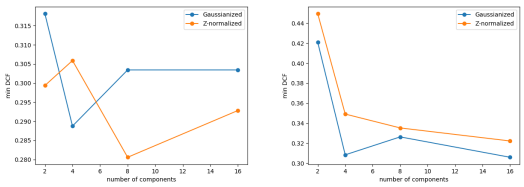
Overall, we can conclude that the RBF kernel yields the best results for the chosen metric. This model is, in addition, the most performing this far.

II.V. Gaussian Mixture Models

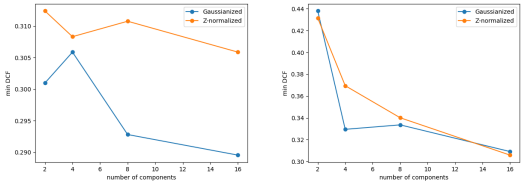
Eventually, Gaussian Mixtures are the last type of classifier employed for this task, for which we expect to yield generally better results than Gaussian models, as GMMs can approximate generic distributions.

We consider both full and diagonal covariance models, with and without covariance

tying, where tying takes place at class level (i.e. different classes have different covariance matrices) for a total of 4 models. The best number of components is selected via cross-validation on gaussianized and standardized features. The results obtained for the described case are shown in Figure 7, where we preferred a line plot over a bar plot for a better visualization of the trend of the minimum DCF as the number of components increases.



(a) *Full GMM* **(b)** *Diag GMM*



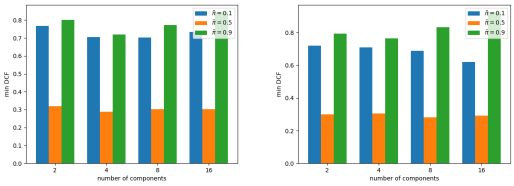
(c) *Tied Full GMM* **(d)** *Tied Diag GMM*

Figure 7: *min DCF for different covariance types of Gaussian Mixtures Models*

	5-fold		
	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Guassianized features			
Tied-GMM (4 components)	0.741	0.306	0.754
GMM(8 components)	0.701	0.298	0.771
Tied-Diag-GMM (16 components)	0.714	0.301	0.690
Diag-GMM (16 components)	0.694	0.306	0.634
Z-normalized features			
Tied-GMM (8 components)	0.769	0.308	0.842
GMM (8 components)	0.689	0.280	0.834
Tied-Diag-GMM (16 components)	0.752	0.306	0.793
Diag-GMM (16 components)	0.774	0.322	0.684

Table 5: *min DCF for Gaussian Mixture Models*

The above plot shows how for diagonal-covariance models (a, b) the min DCF decreases as the number of components increases, whereas it does not appear to be the case for full-covariance models. Table 5 reports the results for the selected models. We notice how the full-covariance GMM yields the minimum detection cost function when applied to Z-normalized features and, for the balanced prior application, provides valid results in order to conduct further analysis using this model. The overall scores of the 8-components full-cov GMM for the 3 applications are shown in Figure 8 .



(a) *Gaussianized data* **(b)** *Z-normalized data*

Figure 8: *min DCF for the full GMM over different priors and for different number of components*

II.VI. Model selection

At the end of our preliminary analysis, we can conclude that the Support Vector Classifier with a radial basis function kernel having $log\gamma = -2$, applied to standardized features yields, globally, the best performances. Nevertheless, the Quadratic Logistic Regression and the Gaussian Mixture with 8 components - both applied to standardized features - performed well. We will mainly take these into account in the following analysis.

To sum up, here are our best choices:

- GMM(8 components)
- Quad-LR ($\lambda = 0$)
- RBF SVC($log\gamma = -2$)

III. Calibration

So far, we only considered the minimum detection cost as a metric to evaluate the different models. However, the cost we pay depends on the threshold we use to perform the class assignments. In this section, we aim at assessing

the performances of the best selected classifiers using the optimal theoretical threshold $t_{opt} = -log\frac{\tilde{\pi}}{1-\tilde{\pi}}$ as well as assessing the performances after calibrating the scores and after fusing models.

III.I. Scores on different thresholds

The first analysis we conduct refers to the detection cost function using as threshold the optimal theoretical one. This metric is referred to as the **actual DCF**.

The obtained results applying t_{opt} are reported in Table 6, where we notice a slight worsening of the performances, which is compatible with the fact that the tested models are non-probabilistic, which implies they do not account for variations in the data, leading to uncalibrated scores.

5-fold ($\tilde{\pi} = 0.5$)

	min DCF	act DCF (t_{opt})
RBF SVC	0.215	0.234
Quad LR	0.273	0.286
GMM	0.280	0.298

Table 6: *minimum and actual DCF for the best performing models*

To tackle this problem we will employ a twofold approach: on the one hand, we will compute a threshold t^* for each application, i.e. for each fold we select the threshold that gives the minimum DCF for an application on the validation set; on the other hand, we will choose a transformation function performing the mapping $s \rightarrow s_{cal} = f(s)$, being $f(\cdot)$ a linear function

$$f(s) = \alpha s + \beta \underbrace{-log\frac{\tilde{\pi}}{1-\tilde{\pi}}}_{t_{opt}}$$

where α and β are estimated via a Linear Logistic Regression. In fact, since the score of the Linear Logistic Regression acts as a posterior log-likelihood ratio, we will recover the calibrated score just subctracting the theoretical optimal threshold t_{opt}. We apply this approach trying different values of λ and eventually picking the best one for each scenario.

The calibrated scores are report in Table 7

5-fold ($\tilde{\pi} = 0.5$)

	act DCF (t^*)	calibrated (LR)
RBF SVC	0.223	0.229
Quad LR	0.284	0.289
GMM	0.292	0.301

Table 7: *actual estimated and calibrated DCF for the best performing models*

We notice that the estimated threshold improves the scores, while the calibration with the LR model proves ineffective, as we experience a worsening of the results.

III.II. Combining the best classifiers

In this section, we analyze the performances of the chosen models, in terms of min and actual DCF, when they're fused, i.e. we combine the scores yielded by the different classifier as follows

$$S = w^T s + b$$

being s the array of scores and w, b the parameters of a Linear Logistic Regression. We again do a sweep on different values of λ and operate using a 5-fold cross validation approach. The "fusions" taken into account are those involving the RBF SVC (the best performing model we have), i.e. we discard the fusion between the Quadratic Logistic Regression an the Gaussian Mixture. Table 8 reports the obtained calibrated scores.

5-fold ($\tilde{\pi} = 0.5$)

	min DCF	act DCF (t_{opt})
SVC + QLR	0.214	0.219
SVC + GMM	0.220	0.231
SVC + GMM + QLR	0.215	0.222

Table 8: *minimum and actual DCF for the combination of best performing models*

We notice that the min DCF improves w.r.t. all single models but the RBF Support Vector Classifier, whose performances are as good as the fusion scenario [1].

III.III. Discussion

From the above results, we conclude that the fusion of the RBF SVC with the Quadratic Logistic Regression yields similar results of the same models fused with an 8 components full-covariance Gaussian Mixture Model. Our choice is, thereby, the simplest of the two as, given the same performances, the simpler of the two reduces the odds of incurring into overfitting and introduces, overall, less computational overheads.

IV. Experimental Results

After analyzing the results of the implemented models on the training set, we aim to assess the performances yielded on the test set. Table 9 shows the results of the Gaussian classifiers for different applications.

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
Full-Cov	0.656	0.337	0.700
Diag-Cov	0.731	0.367	0.867
Tied Full-Cov	0.681	0.315	0.705
Tied Diag-Cov	0.737	0.368	0.930
Gaussianized features			
Full-Cov	0.679	0.336	0.727
Diag-Cov	0.760	0.377	0.911
Tied Full-Cov	0.698	0.319	0.798
Tied Diag-Cov	0.824	0.379	0.942
Gaussianized features, PCA(m=10)			
Full-Cov	0.709	0.320	0.681
Diag-Cov	0.767	0.332	0.886
Tied Full-Cov	0.717	0.313	0.797
Tied Diag-Cov	0.720	0.310	0.804
Gaussianized features, PCA(m=9)			
Full-Cov	0.737	0.326	0.705
Diag-Cov	0.787	0.338	0.790
Tied Full-Cov	0.733	0.322	0.827
Tied Diag-Cov	0.745	0.317	0.839

Table 9: *min DCF for Gaussian models for different values of $\tilde{\pi}$ on the test set.*

The above results prove compatible with those yielded on the training test, especially when applying a k-fold cross validation approach (which is more realiable). On the test set, the tied diagonal-covariance model, which proved less effective in the training phase, yields now the best score. Nevertheless, the full-covariance classifier - which was previously the best performing Gaussian model, keeps producing good results. Applying a 10-components PCA proved again beneficial.

Table 10 shows the evaluation results of the Logistic Regression models on the test set.

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
Log-Reg ($\lambda = 10^{-4}$)	0.704	0.337	0.642
Quad-LR ($\lambda = 0$)	1.000	1.000	0.900
Z-normalized features			
Log-Reg($\lambda = 10^{-2}$)	0.682	0.331	0.654
Quad-LR($\lambda = 0$)	0.638	0.264	0.533
Gaussianized features			
Log-Reg($\lambda = 0$)	0.691	0.341	0.715
Quad-LR($\lambda = 10^{-2}$)	0.657	0.284	0.553
Z-normalized features, PCA(m=10)			
Log-Reg($\lambda = 10^{-2}$)	0.688	0.329	0.651
Quad-LR($\lambda = 0$)	0.670	0.260	0.573

Table 10: *min DCF for Logistic Regression models for different values of $\tilde{\pi}$ on the test set.*

As already discussed in Chapter 2, quadratic models are expected to outperform linear ones for this specific classification task. The results yielded by the Linear and Quadratic Logistic Regression confirm this statement.

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Raw features			
SVC($C = 0.1$)	1.000	0.696	0.932
SVC($C = 0.1, \pi_T^{emp}$)	0.964	0.575	0.927
Z-normalized features			
SVC($C = 1$)	0.670	0.318	0.664
SVC($C = 0.1, \pi_T^{emp}$)	0.685	0.332	0.652
Gaussianized features			
SVC($C = 0.1$)	0.705	0.309	0.841
SVC($C = 0.1, \pi_T^{emp}$)	0.695	0.323	0.787

Table 11: *min DCF for Support Vector Classifier models, with and without re-balancing, on test set.*

[1] The fusion between RBF SVC and Quad LR is unnoticeably better, at the expense of a more complex model

As experienced in the training stage, applying a Quadratic LR on Z-normalized features yields the best results. Exploiting a 10-components PCA proved to be of (slight) benefit in this case.

The test scores associated to Support Vector Machine models are report in Table 11 (linear SVC) and Table 12 (Kernel SVC).

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Z-normalized features			
Poly-SVC($C = 0.1$)	0.617	0.277	0.606
Poly-SVC($C = 0.1, \pi_T^{emp}$)	0.650	0.278	0.572
RBF-SVC($C = 10, log\gamma = -2$)	0.705	0.258	0.610
RBF-SVC($C = 10, log\gamma = -2, \pi_T^{emp}$)	0.752	0.260	0.516

Table 12: *min DCF for Kernel Support Vector Classifier models, with and without re-balancing, on test set.*

We notice that, differently from the training phase, gaussianization yields better results with linear SVC, for the application of main interest, even though Z-normalization still performs well. On the contrary, it performs poorly on raw data. Balancing the features does not introduce visible advantages. Moreover, as expected, kernel methods perform better than linear SVC with Radial Basis Function Kernel SVC still being the most performing method. Rebalancing is again not detrimental nor beneficial.

Eventually, Table 13 shows the results of different GMM models on the test set. We confirm the 8-components full-covariance GMM model as the most effective on Z-normalized features, however on the test set we experience better results on gaussianized features applying a tied-covariance model.

	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$
Guassianized features			
GMM(8 components)	0.701	0.328	0.730
Tied-GMM (16 components)	0.742	0.283	0.642
Diag-GMM (16 components)	0.697	0.320	0.590
Tied-Diag-GMM (16 components)	0.730	0.305	0.649
Z-normalized features			
GMM (8 components)	0.785	0.306	0.850
Tied-GMM (8 components)	0.659	0.297	0.690
Diag-GMM (16 components)	0.781	0.328	0.711
Tied-Diag-GMM (16 components)	0.789	0.325	0.704

Table 13: *min DCF for Gaussian Mixture Models on test set.*

IV.I. Model Fusion

As done for the training step, we compute the scores in terms of actual DCF using the optimal theoretical threshold and the estimated threshold on the evaluation set. In addition we calibrate the scores using a Linear Logistic Regression Model as described in Section III (with DCF computed on test set). Table 14 and 15 show the the results for the single selected models, whereas 16 for their fusion.

	$\tilde{\pi} = 0.5$	
	min DCF	act DCF (t_{opt})
RBF SVC	0.260	0.278
Quad LR	0.264	0.280
GMM	0.306	0.340

Table 14: *minimum and actual DCF for the best performing models*

	$\tilde{\pi} = 0.5$	
	act DCF (t^*)	calibrated (LR)
RBF SVC	0.278	0.282
Quad LR	0.293	0.285
GMM	0.325	0.315

Table 15: *actual estimated and calibrated DCF for the best performing models*

We observe how selecting the optimal threshold is not effective (expect for the GMM model) and that, overall, calibration is beneficial, especially for the Quadratic LogReg and the GMM.

	$\tilde{\pi} = 0.5$	
	min DCF	act DCF (t_{opt})
SVC + QLR	0.252	0.261
SVC + GMM	0.260	0.268
SVC + GMM + QLR	0.247	0.258

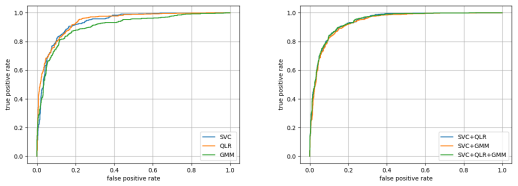
Table 16: *minimum and actual DCF for the combination of best performing models*

As regards model fusions, we compute the actual DCF using the theoretical threshold and re-use the best values of the norm scaler of the LR model (λ) during the hyperparameter tuning. We observe how joining models' score proves definitely more effective than using single models and, as already experienced, combining the three best models yields again the

best score, both in temrs of min DCF and act DCF, despite obtaining slightly worse results than those yielded on the training set.

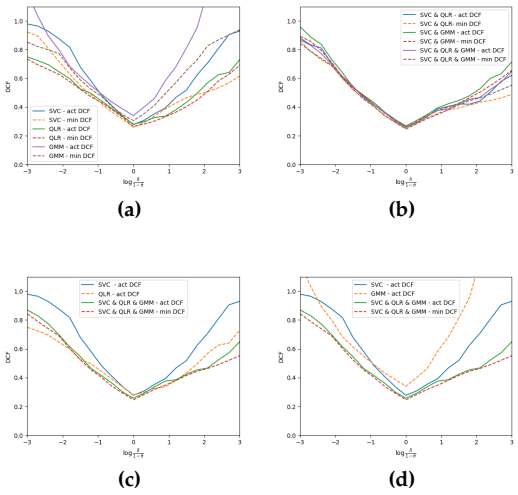
IV.II. Different Applications

In this last section, we assess the performance of our best models for different applications. Figure 9 shows the ROC curves for the single models and for their fusions. As regards the latter, the curves are almost overlapped, which implies they yield very similar scores, whereas for the former, the RBF SVC and the Quad LogReg perform better than the GMM.



(a) *single models* **(b)** *fusion*

Figure 9: *ROC plots*



(a) **(b)**

(c) **(d)**

Figure 10: *Bayes error plots against different values of log prior odds*

Figure 10 shows the mininum and actual DCF against the different applications, i.e. the prior log-odds. We compare single models **(a)**, fusions **(b)**, SVC and QLR with fusions **(c)**, SVC and GMM against fusions **(d)**. Fusions yield similar performances across all the applications and, overall, always outperform single models, with the slight exception of the Quadratic Logistic Regression being better for some applications **(a, c)**.

V. Conclusion

In this work we analyzed the UCI Wine Quality Dataset applying a set of "traditional" Machine Learning Models and using the minum detection cost function and its derivatives as metrics. We showed how linear models are not very well suited for classifying wine features and how other type of space transformations adapt better to the data. We eventually explored the benefits of combining the most performing models and proved how our approach is effective on test data, yielding results comparable to the expected ones in light of the analysis conducted towards model selection and tuning.