Machine Learning and Pattern Recognition Project:

Wine and Quality

Abstract

In this project, we were asked to experiment what we've learnt during the course with a real-world dataset publicly available for research and we were expected to submit a report about the dataset and the algorithms used.
I chose the "Wine Quality" [1] dataset.
After performing the required tasks on a dataset of my choice, here lies my final report.

1. Description of the problem and data visualization

The task requires verifying whether a wine is of good or bad quality. The dataset used comes from the UCI Machine Learning Repository and is related to red and white variants of the Portuguese "Vinho Verde" wine. The training set is made of 1839 instances of merged red and white wine, 1226 of them are classified as "Bad Quality" (L==0), 613 as "Good Quality" (L==1), so the dataset is slightly imbalanced.
The dataset contains 11 features, as described below:

Input variables
(Based on physicochemical tests):

1. Fixed Acidity
2. Volatile Acidity
3. Citric Acid
4. Residual Sugar
5. Chlorides
6. Free Sulfur Dioxide
7. Total Sulfur Dioxide
8. Density
9. pH
10. Sulphates
11. Alcohol

Output variables
(Based on sensory data):

12. Quality
(0 – low quality, 1- high quality)

Figure 1 shows the distribution of the original features between the two classes. From a first view, features don't have zero mean, so we can apply Z-normalization to center the data and normalize the variance.

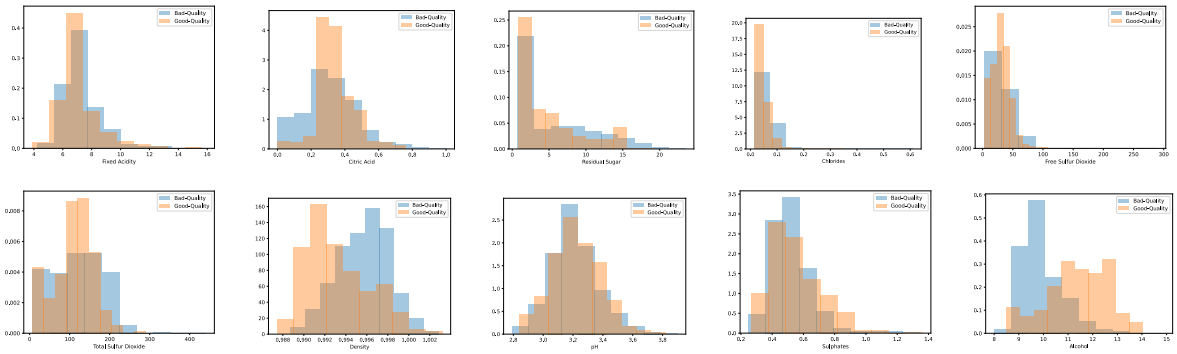


Figure 1: Histograms of features of train dataset

Secondarily, most of the features don't seem to resemble gaussian-like distributions, due to the presence of outliers, so Gaussianization pre-processing can be applied.
The main idea of Gaussianization is to transform some data distribution $\mathcal{D}$ into an approximate Gaussian distribution $\mathcal{N}$, by first computing the rank of a feature x and then passing it to the inverse of the cumulative distribution function.
The results are shown below.

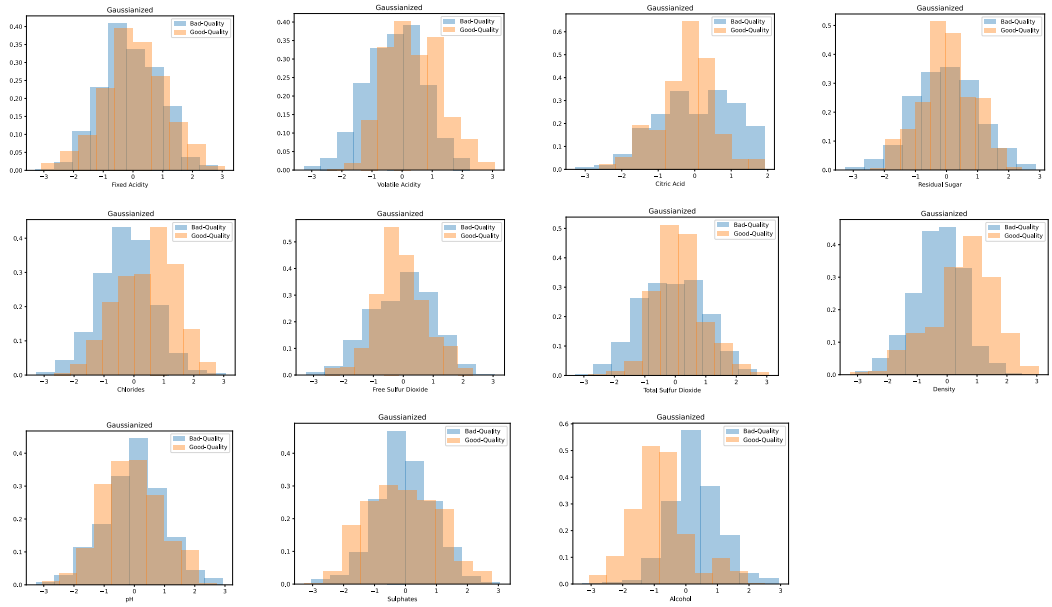


Figure 2: Histograms of Gaussianized features of train

We can see a significant overlap in both non-Gaussianized and Gaussianized plots through most of the features (with exception to Density and Alcohol), so we expect them to be not so discriminant on their own.
At the end, plotting the Pearson correlation matrix, an analysis of the correlation between the features can be provided. By looking at the results, most of the values are small (<0.3 in absolute term), but some features appear to be correlated (such as Free Sulfur Dioxide and Total Sulfur Dioxide, and also Alcohol and Density seems to be slightly correlated), so PCA can be tested to see if retaining (possibly) 9 or 10 uncorrelated dimensions helps our model perform better.

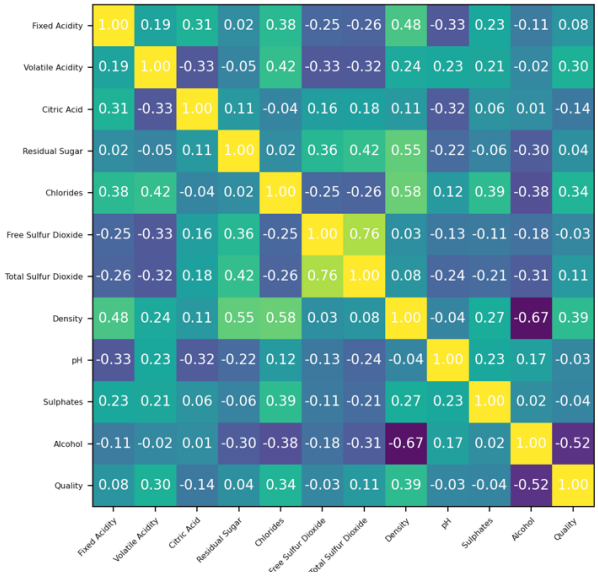


Figure 3: Pearson Correlation Coefficient Matrix

2. Classification

To decide which model is the most promising, both single-fold and k-fold validations have been used with different classifiers, evaluating their performance in terms of minimum Detection Cost Function (*minDCF,* from now on), or the cost paid by selecting the optimal threshold.

The following analysis is based on three applications (1 balanced and 2 unbalanced):

1. $(\pi, C_{fp}, C_{fn}) = (0.5,1,1)$
2. $(\pi, C_{fp}, C_{fn}) = (0.9,1,1)$
3. $(\pi, C_{fp}, C_{fn}) = (0.1,1,1)$

2.1 Gaussian models

The first classifier analyzed are the Gaussian classifiers, which assumes that the data, given the class, can be described by a Gaussian distribution.
The following table shows the results in terms of minimum DCF for a Full-Covariance model, a Naïve-Bayes model (Diag), a Tied-Full Covariance model and a Tide Naïve-Bayes model.
The models have been tested on both raw and Gaussianized features, applying dimensionality reduction at the end to assess its utility.

The Naïve-Bayes and Tied Naïve-Bayes models (with or without Gaussianization and PCA) didn't give good results in comparison to the rest.
Applying PCA slightly improves their performance, probably because the within-class correlation decreases by removing directions with low variances, but in general values are pretty much comparable, so it does not bring much benefit. This assess our idea that there were a couple features strongly correlated with one another.

The best performance is achieved by the full-covariance models, probably because the dataset is large enough to estimate the entire covariance matrix for each class instead of relying on the assumptions that the diagonal and tied models make.
And since the full-covariance has a quadratic surface rule, I decided to proceed by analyzing also the quadratic models, after the linear ones.

	Single Fold			**5-Fold**		
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
	Raw - No PCA					
Full - Cov	**0.304**	0.812	0.777	**0.313**	0.843	0.779
Diag - Cov	0.437	0.875	0.818	0.420	0.921	0.846
Tied Full-Cov	0.334	0.733	0.779	0.333	0.748	0.811
Tied Diag-Cov	0.412	0.901	0.832	0.403	0.932	0.866
	Gaussianized - No PCA					
Full - Cov	**0.272**	0.762	0.802	**0.307**	0.867	0.841
Diag - Cov	0.449	0.771	0.859	0.425	0.922	0.862
Tied Full-Cov	0.348	0.807	0.829	0.345	0.758	0.854
Tied Diag-Cov	0.434	0.903	0.835	0.422	0.894	0.880

	Gaussianized - PCA (m = 10)					
Full - Cov	**0.287**	0.728	0.760	**0.311**	0.815	0.812
Diag - Cov	0.373	0.786	0.733	0.388	0.867	0.839
Tied Full-Cov	0.346	0.711	0.776	0.348	0.711	0.823
Tied Diag-Cov	0.346	0.711	0.786	0.348	0.714	0.818
	Gaussianized - PCA (m = 9)					
Full - Cov	**0.288**	0.769	0.780	**0.309**	0.799	0.841
Diag - Cov	0.379	0.766	0.735	0.373	0.822	0.826
Tied Full-Cov	0.361	0.709	0.817	0.342	0.687	0.831
Tied Diag-Cov	0.369	0.711	0.812	0.344	0.706	0.832

Since the applications 2 and 3 look already too risky at this early stage, I've decided not to proceed analyzing them to only focus on the balanced uniform one: $(\pi, C_{fp}, C_{fn}) = (0.5, 1, 1)$

2.2 Logistic Regression

In this section the focus turns to the discriminative models. The first one is the linear regression, which models class posterior distribution by looking for the linear hyperplanes that maximizes the likelihood of the training labels. Which means that we try minimizing the average cross entropy between the empirical distribution of the data and the predicted labels distribution, and we try to minimize the empirical risk of misclassifying the data as well.

Since our classes are unbalanced and the logistic regression model embeds the prior empirical probability, the model as seen in the laboratories needs to be modified as it follows to reflect the prior of our application:

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

The application to be tested has $\pi_t = 0.5$ and the first step was tuning the hyperparameter λ. The following table shows how the minDCF varies with respect to λ (using balanced/unbalanced models).
Since this model adapts well also to features whose distribution is not Gaussian, Gaussianization won't be of too much help, so I've decided not to use it. While Normalization only could be useful to avoid numerical errors and improve the minDCF.
PCA hasn't been used either, because it's reasonable to think that since the samples have only 10 dimensions, the model can benefit from using all of them.

Single Fold		5-Fold	
without balancing RAW			
1e-06	0.366	1e-06	0.342
1e-03	**0.352**	1e-03	**0.345**
0.1	0.393	0.1	0.411
1.0	0.391	1.0	0.466
without balancing Normalized			
1e-06	0.361	1e-06	0.344
1e-03	**0.358**	1e-03	**0.340**
0.1	0.368	0.1	0.355

1.0	0.392	1.0	0.403
with balancing RAW			
1e-06	0.365	1e-06	0.355
1e-03	**0.366**	1e-03	**0.352**
0.1	0.398	0.1	0.416
1.0	0.390	1.0	0.458
with balancing Normalized			
1e-06	0.368	1e-06	0.357
1e-03	**0.368**	1e-03	**0.355**
0.1	0.351	0.1	0.354
1.0	0.392	1.0	0.392

Best results have been achieved with $\lambda = 10^{-3}$.
It can be observed that linear logistic regression model has performances comparable to tied-covariance Gaussian models, because they both rely on a linear separation rule.
Moreover, since Full-Covariance model is still the one to perform better and it relies on a quadratic separation rule, I've decided to proceed analyzing Quadratic Logistic Regression.

2.3 Quadratic Logistic Regression

The following table show how the minDCF varies with respect to λ (using balanced/unbalanced model).

Single Fold		**5-Fold**	
without balancing RAW			
1e-06	0.426	1e-06	0.397
1e-03	0.417	1e-03	**0.403**
0.1	**0.402**	0.1	0.412
1.0	0.397	1.0	0.419
without balancing Normalized			
1e-06	0.264	1e-06	0.277
1e-03	**0.262**	1e-03	**0.276**
0.1	0.307	0.1	0.320
1.0	0.375	1.0	0.370
with balancing RAW			
1e-06	1.0	1e-06	1.0
1e-03	1.0	1e-03	1.0
0.1	1.0	0.1	1.0
1.0	1.0	1.0	1.0
with balancing Normalized			
1e-06	0.274	1e-06	**0.274**
1e-03	**0.270**	1e-03	0.352
0.1	0.309	0.1	0.417
1.0	0.372	1.0	0.458

Quadratic models generally perform better, since their separation surfaces are more complex and can describe better the real distribution of data.
It can be observed that Quadratic Logistic Regression model results are similar and even outperforms the full-covariance Gaussian model one.
We can also assess that balancing was quite useless in our case and, without the use of Normalization in the balanced version, it leads my application to a numerical error, so a non-balanced version with $\lambda = 10^{-3}$ is still the best option at this moment.

2.4 SVM

The SVM classifier is a discriminative classifier that looks for a separation hyperplane with the maximum margin between two classes, so the maximum distance between the hyperplane and their closest points.

Since classes are unbalanced, also class rebalancing is tested again:

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

Subjected to:

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

$$C_i = C_T \; for \; samples \; of \; class \; H_T$$

$$C_i = C_F \; for \; samples \; of \; class \; H_F$$

Specifically, two different values of C (proportional to the actual priors of the two classes) have been used:

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

where π_T^{emp} and π_F^{emp} are the two empirical priors for the two classes over the training set.

We start by analyzing linear SVM, although I expect the results to be not as good as the ones we would get by using a quadratic kernel.

Blue: unbalanced, **Orange**: balanced

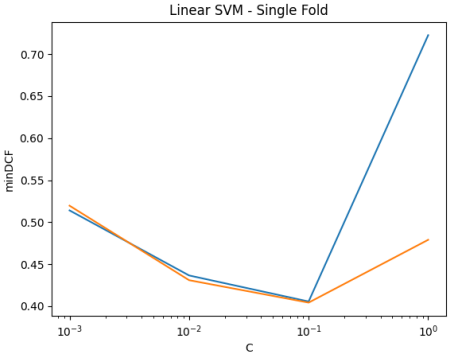


Figure 4: Single Fold Linear SVM with different C values (RAW)

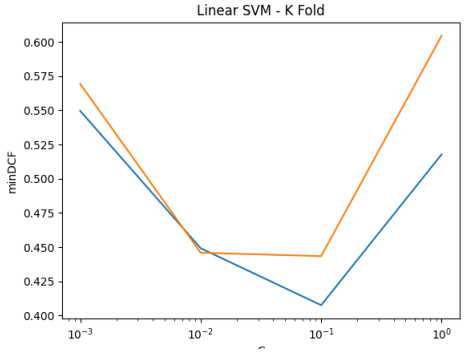


Figure 5: 5-Fold Linear SVM with different C values (RAW)

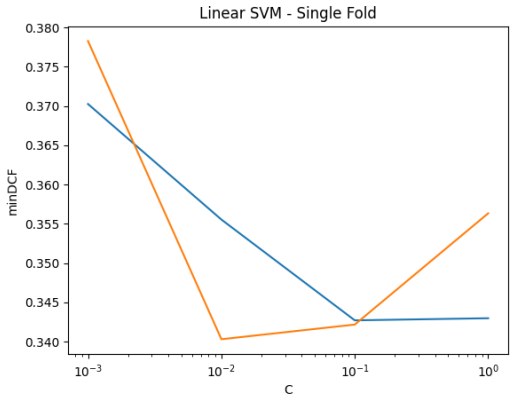


Figure 6: Single Fold Linear SVM with different C values (Normalized)

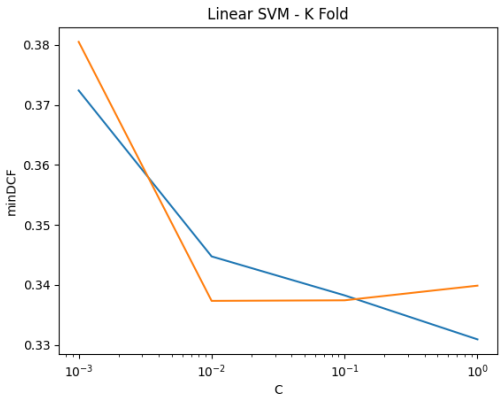


Figure 7: K-Fold Linear SVM with different C values (Normalized)

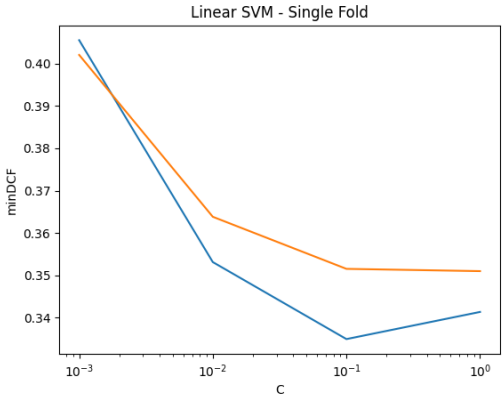


Figure 8: Single Fold Linear SVM with different C values (Gaussianized)

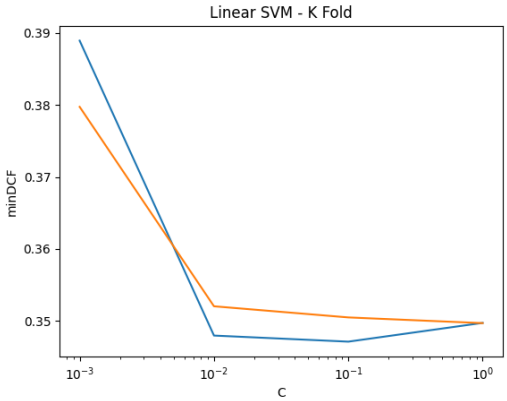


Figure 9: K-Fold Linear SVM with different C values (Gaussianized)

The hyperparameter C has been set to $C = 0.1$ since it is the biggest value that assures good performances in most of the cases.
The graphs shows that performances are in general comparable to the ones from Linear Logistic Regression and Tied-Covariance gaussian models. Moreover, Normalization appears to be quite helpful, while re-balancing the model does not bring too much benefit as well as Gaussianization so they won't be used in the analysis of the following SVM models, not because they're detrimental, but just to reduce computational time.

For the next analysis I focus on quadratic kernel SVM models which I expect to perform like Quadratic Logistic Regression, as they both rely on more complex rules for class separation and already proved to reach better results.

- **SVM with quadratic kernel**

The first non-linear SVM model analyzed is the polynomial quadratic kernel:

$$k(x_1, x_2) = (x_1^T x_2 + c)^d$$

Where d = 2 (quadratic) and c is a hyper-parameter selected through cross-validation.

To add a regularized bias, a constant value K has been added to the kernel function and kept set to 1.

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

Blue: RAW, **Orange**: Normalized

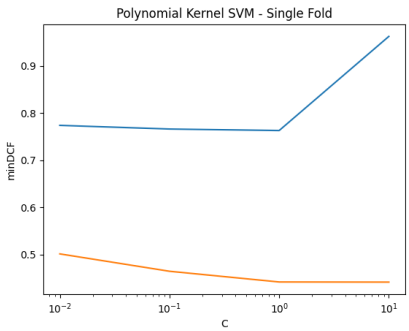


Figure 10: Single Fold Polynomial SVM with $\mathbf{c = 0}$

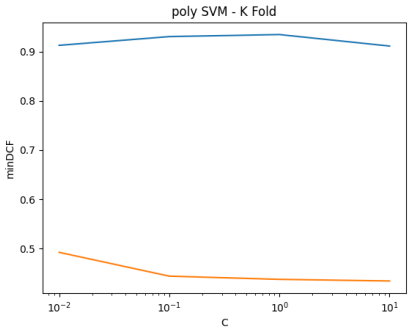


Figure 11: 5-Fold Polynomial SVM with $\mathbf{c = 0}$

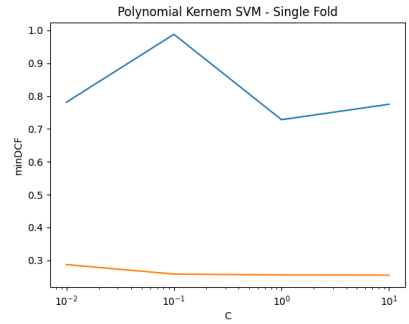


Figure 12: Single Fold Polynomial SVM with $\mathbf{c = 1}$

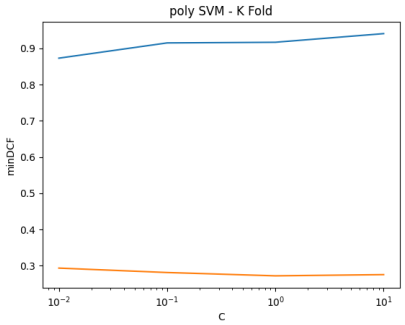


Figure 13: 5-Fold Polynomial SVM with $\mathbf{c = 1}$

The second one is RBF (Radial Basis Function) kernel:

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

Where γ is a hyper-parameter selected through cross-validation.

Blue: RAW, **Orange**: Normalized

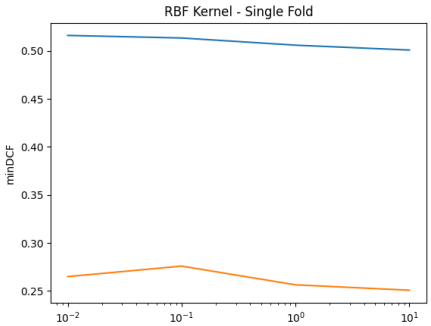


Figure 14: Single Fold RBF SVM with **gamma = 1**

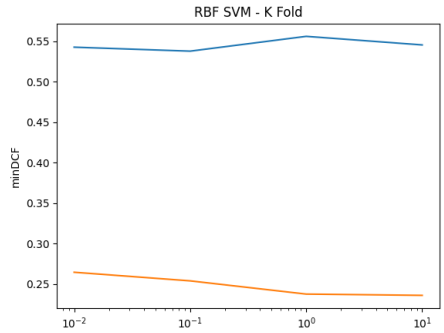


Figure 15: 5-Fold RBF SVM with **gamma = 1**

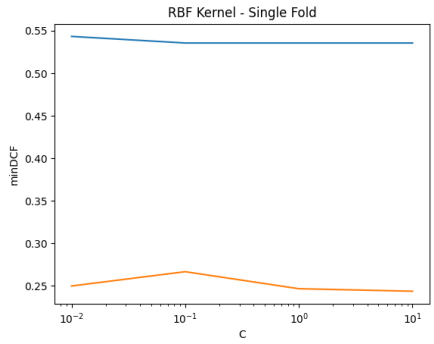


Figure 16: Single Fold RBF SVM with **gamma = 2**

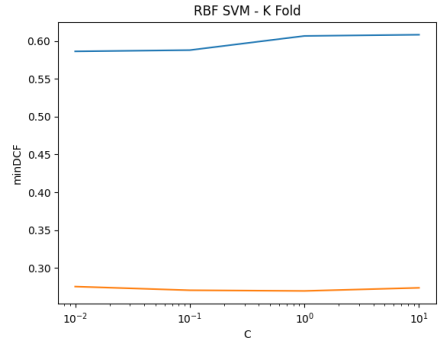


Figure 17: 5-Fold RBF SVM with **gamma = 2**

The best results have been achieved with K=1, c=1, C=1.0 for the Polynomial kernel SVM, while K=1, γ=1, C=1.0 for the RBF kernel SVM, by using the normalized version of the dataset both in single fold and k-fold approaches.
The values of minDCF reached confirm that quadratic surfaces are better at discriminating the classes of our dataset.

In particular, the RBF results with the previously described hyperparameters outperforms our previous models for the target application.
Therefore the best model found so far is the RBF SVM trained with Normalized features.

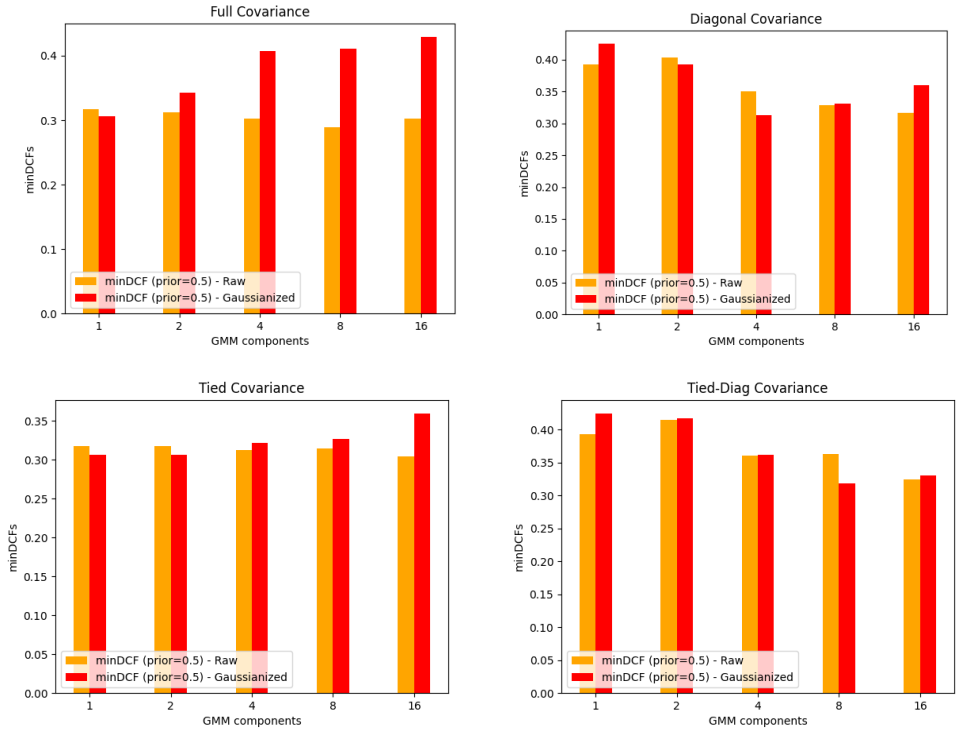
Comparison between quadratic models	**minDCF (5-fold)**
MVG Full-Cov (Gaussianized - No PCA)	0.307
Quadratic Log Reg (λ=1e-3 - Normalized)	0.276
Quad SVM (K=1, c=1, C=0.1 - Normalized)	0.281
RBF SVM ($\gamma = 1, C = 1.0$ - Normalized)	**0.237**

2.5 GMM

The last model to be tested is the GMM, which is a probabilistic model that allows approximating the density of a R.V. $\boldsymbol{X}$ when the density of $\boldsymbol{X}$ is not known.
The expectation for them is to perform better than standard Gaussian models, as they can approximate any sufficiently regular distribution and be used as a classification model by

assuming that each class can be approximated by a GMM with a given number of sub-components.

Different variants of GMMs have been analyzed (full covariance, diagonal covariance and tied ones). The graphs below show the different performances in terms of minDCF, the optimal number of components has been selected through cross-validation and 5-Fold approach only, since GMM benefits from having large amount of training data.



The models trained with Gaussianized features have similar performances to the ones trained with raw features.
Best results have been achieved with GMM Full-Covariance (RAW) and 8 components.

In the specific:

Number of components	**1**	**2**	**4**	**8**	**16**
Raw					
Full Cov	0.317	0.312	0.303	**0.289**	0.302
Diag	0.393	0.404	0.350	0.329	0.316
Tied-Full	0.317	0.317	0.312	0.314	0.304
Tied-Diag	0.393	0.415	0.361	0.363	0.324
Gaussianized					
Full Cov	0.306	0.343	0.407	0.411	0.429
Diag	0.425	0.393	0.313	0.331	0.360
Tied-Full	0.306	0.306	0.321	0.327	0.359
Tied-Diag	0.424	0.417	0.362	0.318	0.330

2.7 Conclusions

It can be concluded that the most promising model is the RBF kernel SVM with hyperparameters C = 1.0, γ = 1, using Z-normalized features.
However, also the logistic regression with quadratic separation surfaces (λ = 1e-3, Normalized features) and Quad SVM (K=1, c=1, C=0.1, Normalized features) provide good results. Therefore, all three will be taken into consideration in the following part. Decisions are made considering the results of cross-validation, since it is expected to provide more robust results. However, the results achieved with the single-fold approach agree with those of cross-validation in most cases.

2.8 Analysis in terms of actual DCF

Up to now the analysis was based on the minDCF, which measures the cost we would pay if we made optimal decisions for the evaluation set using the recognizer scores.

However, in our case, the cost that we actually pay depends on the goodness of the threshold we use in practice to perform class assignment.
If the scores are well calibrated, the optimal threshold that optimizes the Bayes risk is:

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

We can evaluate the actDCF to assess how good models would be if we were using the theoretical threshold for the target application (considering the scores as if they were calibrated):

5-Fold ($\tilde{\pi} = 0.5$)		
	minDCF	**actDCF**
Quad Log-Reg	0.276	0.288
Poly SVM	0.281	0.295
RBF SVM	0.237	0.341

We can observe that the Quad Log-Reg and Poly SVM provide scores that are almost calibrated, while the calibration of RBF SVM scores is quite critic.

The reason for that could be related to the fact that for SVM Quad models, I preferred using the unbalanced version to reduce computational time, since the balanced one didn't seem to bring a lot of benefit. But a balanced approach (with $\tilde{\pi} = 0.5$) as well as recalibrating our scores would have probably helped to reach a tighter gap between the values of minDCF and actDCF.

3. Evaluation

In conclusion, after analyzing the performances of different models on the training set and choosing the best one, the effectiveness of the decisions is evaluated by analyzing the results of the previously tested models also on the test set. Hyperparameters were chosen according to the best choice of hyperparameters on the training set. Results are given in the form of minimum DCF.

3.1 Gaussian models

Below are the Gaussian classifiers using both single-fold protocol (the model trained over 66% of the data) and the 5- fold cross validation protocol (the final model that was re-trained using the whole dataset):

	66% Data	**100% Data**
	$\pi = 0.5$	$\pi = 0.5$
	Raw - No PCA	
Full - Cov	0.332	0.336
Diag - Cov	0.376	0.366
Tied Full-Cov	0.315	0.315
Tied Diag-Cov	0.370	0.368
	Gaussianized - No PCA	
Full - Cov	0.329	0.335
Diag - Cov	0.382	0.377
Tied Full-Cov	0.332	0.320
Tied Diag-Cov	0.389	0.379
	Gaussianized - PCA (m = 10)	
Full - Cov	0.319	0.320
Diag - Cov	0.345	0.331
Tied Full-Cov	0.320	0.313
Tied Diag-Cov	0.318	0.310
	Gaussianized - PCA (m = 9)	
Full - Cov	0.333	0.326
Diag - Cov	0.360	0.338
Tied Full-Cov	0.327	0.322
Tied Diag-Cov	0.324	0.317

3.2 Logistic Regression

66% Data		100% Data	
without balancing RAW			
1e-03	0.321	1e-03	0.322
without balancing Normalized			
1e-03	0.332	1e-03	0.325

3.3 Quadratic Logistic Regression

66% Data		100% Data	
without balancing RAW			
1e-03	0.409	1e-03	0.415
without balancing Normalized			
1e-03	0.255	1e-03	0.256

3.5 SVM

66% Data		100% Data
Linear		
(C=0.1, unbalanced)	0.324	0.315
Quadratic - Poly		
(K=1, c=1, C=0.1 - Normalized)	0.251	0.278
Quadratic - RBF		
(K=1, gamma = 1, C=1.0 - Normalized)	0.272	0.274

3.6 GMM

Number of components	1	2	4	8	16
Raw (66% Data)					
Full Cov	0.333	0.322	0.333	0.308	0.371
Diag	0.361	0.364	0.328	0.324	0.342
Tied-Full	0.333	0.333	0.329	0.295	0.294
Tied-Diag	0.361	0.401	0.311	0.358	0.304
Gaussianized (66% Data)					
Full Cov	0.329	0.301	0.322	0.363	0.389
Diag	0.382	0.338	0.293	0.304	0.318
Tied-Full	0.329	0.329	0.320	0.274	0.265
Tied-Diag	0.382	0.355	0.317	0.275	0.307

Number of components	1	2	4	8	16
Raw (100% Data)					
Full Cov	0.341	0.320	0.305	0.289	0.330
Diag	0.367	0.369	0.341	0.337	0.319
Tied-Full	0.341	0.340	0.320	0.316	0.288
Tied-Diag	0.367	0.389	0.348	0.320	0.311
Gaussianized (100% Data)					
Full Cov	0.336	0.308	0.317	0.328	0.382
Diag	0.377	0.332	0.289	0.325	0.343
Tied-Full	0.336	0.335	0.357	0.282	0.284
Tied-Diag	0.377	0.408	0.362	0.292	0.292

Comparison between quadratic models (using same decisions from validation)	**minDCF (66% Data)**	**minDCF (100% Data)**
MVG Full-Cov (Gaussianized - No PCA)	0.329	0.335
Quadratic Log Reg (λ=1e-3 - Normalized)	0.255	0.256
Quad SVM (K=1, c=1, C=0.1 - Normalized)	0.251	0.278
RBF SVM ($\gamma = 1, C = 1.0$ - Normalized)	0.272	0.274

The results are quite consistent with those achieved on validation set, even though Quadratic Logistic Regression and Quad SVM seems to perform slightly better than the RBF SVM model on the evaluation set using the same parameters found in the previous phase.

The results achieved by using a partition of the data and the whole dataset are very close and this means that for these models 66% of the data is enough to obtain good estimates of the model parameters.

3.7 Analysis in terms of actual DCF

	66% Data		**100% Data**	
	minDCF	**actDCF**	**minDCF**	**actDCF**
Quad Log-Reg	0.255	0.307	0.256	0.330
Poly SVM	0.251	0.328	0.278	0.320
RBF SVM	0.272	0.455	0.274	0.416

We can observe that all the models provide uncalibrated scores, some more critical than others and the same considerations made in the validation phase apply to these results.

4. Conclusions

The Wine Quality dataset can be classified effectively using all the three models selected above (Quad Log-Reg, Quad SVM, RBF SVM), achieving a DCF cost of ≈ 0.300.

Overall, the similarity between validation and evaluation results suggests that the evaluation population is sufficiently similar to the training population and the choices made on the training/validation sets proved effective also for the evaluation data.

A few things could have been done to achieve a deeper and more precise analysis, such as analyzing all the models with the three applications initially prefixed to understand better the behavior of our models in different conditions, but I expected them to be relatively ineffective given the poor results gotten on the first model analyzed.