Wine Project Report

Nocera Ruggero (S292442)
Quarta Matteo (S292477)

Abstract In this report we analyze how effective are different classifiers and different preprocessing techniques applied to a binary classification task. The dataset used to train and test the models taken in consideration is a modified version of the one provided by *Modeling wine preferences by data mining from physicochemical properties* from *Decision Support Systems, Elsevier* (P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.). Due to limited time and computational power, results may be just close-to or far-from optimal, depending on the time required to train a model. Although reasonable results have been obtained, the purpose of this report is not to describe the best possible application, but to make an analysis of the various techniques presented during the course.

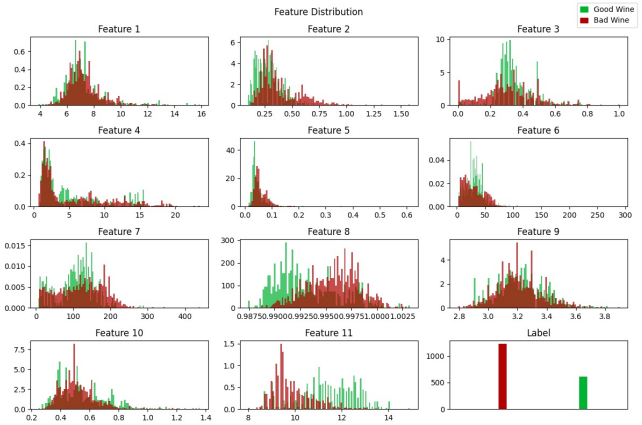
Contents

1 Preliminary Data Analysis

1.1 Feature Distribution

Before discussing the models, their implementation and their effectiveness we briefly take a look at how the features are distributed. For convenience we shall now report the legend just once, but keep in mind that in the following pictures the red color is associated to class 0 (which we will also be referring to as class *Bad*) and the green color is associated to class 1 (which will be class *Good*).

Figure 1: Histogram of Class' Features



First of all, our training dataset is unbalanced. In the next pages we will be classifying samples obtained from a K-Fold[1]Validation approach, using a theoretical threshold given by:

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

For the threshold to be optimal, we should use the empirical prior $\pi \approx 0.33$ based on a frequentist approach; Instead we will be using a non-optimal prior $\tilde{\pi} = .5$ as it is the application we are going to be targeting .

We can notice some things: while some features are similar between class Good and class Bad (e.g. Feature 1) others differ substantially and could be helpful in discriminating samples (e.g. Feature 8).

The features are also distributed in different ways: some do look Gaussian distributed (e.g. Feature 9) or regular enough (e.g. Feature 5) and could thus be well estimated by Gaussian models[2], other act in a more irregular way.

.

To take a closer look we project the data on the two dimensional plane. We will be using 3 methods: PCA[3], Normalization + PCA, Normalization + Whitening.

[1]More on the value of K later
[2]Both Gaussian Classifiers and GMMs
[3]No data centering applied before

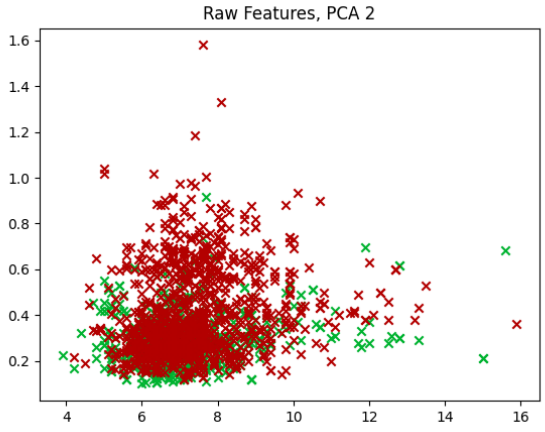


Figure 2: 2D-PCA Projection

The points don't seem much correlated, we expect Whitening to not be of much use.

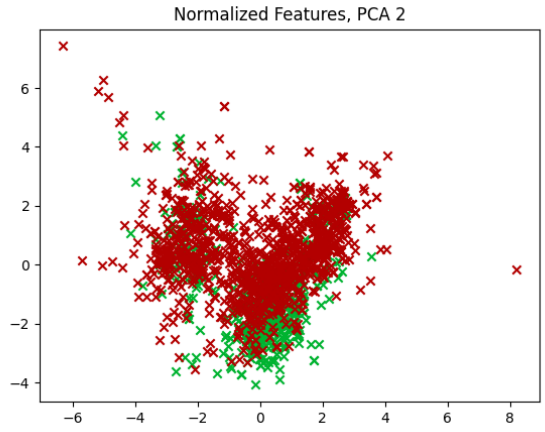


Figure 3: 2D-PCA Projection, Normalized Data

The normalized projection splits data in two clusters, each containing some samples of each class, but some points of different class also do get apart one another: normalization could be of use

Our chosen normalization technique is Z-Normalization. From a preliminary, off-record analysis we found Gaussianization to produce similar results while being slower.

Lastly, the Whitened data.

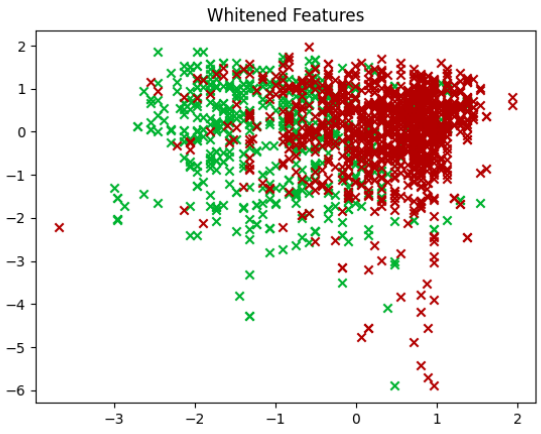


Figure 4: 2D-PCA Projection, Whitened Data

Results are more interesting than expected, doing well in getting points of different class apart one another.

2 Pre-Processing Analysis

In this section we will run the models without trying to optimize them as much as we can.

2.1 Pre-Processing MVG Classifiers

Since MVG classifiers are fairly similar, we will discard the less promising ones now.

Type	PCA	DCF	minDCF
Raw	8	0.375	**0.321**
Raw	9	**0.360**	0.326
Norm.	10	0.419	0.330
Norm.	11	0.424	0.322
Whit.	11	0.424	0.322
Whit.	8	0.392	0.342

Table 1: Full-Covariance MVG - Best Results

Type	PCA	DCF	minDCF
Raw	11	**0.342**	0.332
Norm.	**5**	0.414	**0.330**
Norm.	11	**0.342**	0.332
Whit.	11	**0.342**	0.332

Table 2: Tied-Covariance MVG - Best Results

Type	PCA	DCF	minDCF
Raw	9	0.404	0.365
Raw	7	**0.391**	0.370
Norm.	9	0.428	0.350
Whit.	11	0.460	**0.344**
Whit.	5	0.394	0.374

Table 3: Naive-Bayes MVG - Best Results

The results are in line with our assumptions. Looking again at picture 2 we wouldn't expect the Naive Bayes to perform much differently from the Full Covariance one. The best model, with a modest margin, is the Tied Covariance one so this will be our chosen one.

Notice that while the Tied Covaraiance model achieved the best DCF[4]of all classifiers, the MVG achieved the lowest Minimum DCF values, hiting that it may be the model wit the best theoretical separation capabilities.

Whitening as expected is of no use, we won't further experiment it.

2.2 Pre-Processing for GMMs

For the GMM model we arbitrarily train it with four components. Optimizations will be made later.

The starting point for our EM algorithm is identity covariance matrices and means placed around the dataset mean.

Type	PCA	DCF	minDCF
Raw	No	0.410	**0.394**
Normalized	9	**0.407**	0.402
Whitened	4	0.429	0.420

Table 4: GMM - Best Results

[4]Normalized DCF, $\pi_T = .5$, equal costs

As for our Gaussian Classifiers, whitening is not useful, but we will further experiment with normalization. High dimensionality seems preferred.

2.3 Pre-Processing for SVMs

Differently for what we did up to now, given the non-probabilistic nature of SVM scores, we will be deciding our optimal model for both types of kernel SVMs on just minimum DCFs values.

2.3.1 Pre-Processing for Linear SVM

Type	K	C	PCA	minDCF
Raw	0	10.0	6	0.568
Norm.	10	0.1	9	**0.330**
Whit.	1.0	1.0	9	0.331

Table 5: Linear SVM - Best Results

The Linear SVM model performed much better with normalization. Whitening looks again irrelevant.

PCA is useful to obtain lower values of minimum DCF in all cases, even if the model trained with raw features performs worse than models trained with preprocessed features.

2.3.2 Pre-Processing for Polynomial Kernel SVMs

The dummy parameters chosen for execution are:

$$k(\mathbf{x_1}, \mathbf{x_2}) = (\mathbf{x}_1^T \mathbf{x}_2 + 1)^2 + 0.5$$

Type	PCA	minDCF
Raw	7	0.545
Normalized	6	**0.381**
Whitened	11	0.393

Table 6: Polynomail Kernel SVM - Best Results

Here we notice some differences with previous models: normalization greatly improves our classification, and small dimensionality obtain lower DCFs, even if not significanlty.

Again, whitening does not make any difference.

2.4 Pre-Processing for RBF Kernel SVMs

The dummy parameters for the RBF Kernel are:

$$k(\mathbf{x_1}, \mathbf{x_2}) = e^{-\gamma||\mathbf{x}_1 - \mathbf{x}_2||^2} + b = e^{-0.05||\mathbf{x}_1 - \mathbf{x}_2||^2} + 0.05$$

Type	PCA	minDCF
Raw	10	0.579
Normalized	No	**0.352**
Whitened	No	**0.352**

Table 7: RBF Kernel SVM - Best Results

Opposed to polynomial SVMs, RBF prefers high dimensionality. Also in this case normalization plays and important role while whitening does not look useful.

So far, this is our most promising model.

2.5 Pre-Processing for Logistic Regression Model

2.5.1 Linear Logistic Regression

Normalization helps in obtaining lower minimum DCFs compared to raw features, and higher dimensionality is preferred. Whitening in this case looks extremely harmful and we won't further analyze it.

Type	PCA	DCF	minDCF
Raw ($\lambda = 0.0001$)	10	0.369	0.342
Norm. ($\lambda = 0.01$)	9	0.369	**0.336**
Whit. ($\lambda = 0.0001$)	No	0.549	0.503

Table 8: Linear Logistic Regression - Best Results

The reported results are obtained through optimized parameters.

2.5.2 Quadratic Logistic Regression

In the Quadratic Logistic Regression, after the preprocessing stage, the features space was expanded through

$$\phi(\mathbf{x}) = \begin{bmatrix} vec\langle \mathbf{x}\mathbf{x}^{\mathbf{T}} \rangle \\ \mathbf{x} \end{bmatrix}$$

Whitening does not harm as in the linear case but it isn't helpful either, and we will again discard it. PCA is more helpful when working with raw features.

Type	PCA	DCF	minDCF
Raw ($\lambda = 0.1$)	6	0.385	0.366
Norm. ($\lambda = 0.001$)	10	0.324	**0.307**
Whit. ($\lambda = 0.001$)	10	0.324	**0.307**

Table 9: Quadratic Logistic Regression - Best Results

3 Optimizing Models

3.1 Optimizing Logistic Regression

To find the optimal value of λ several have been tried in range $[10^{-4}, 10^{3}]$ for both linear and quadratic model.

Model	Type	λ	PCA	DCF	minDCF
Linear	Norm.	0.01	9	0.369	0.336
Quadratic	Norm.	0.001	10	**0.324**	0.307

Table 10: Best λ values for Logistic Regression

To lower the difference between the DCF and the minimum DCF, we try recalibrating scores. To do so, we scale and shift as in:

$$s_{new} = f(s_{old}) = \alpha s + \beta$$

Where α, β are extimated through logistic regression.

Model	Type	λ	PCA	DCF	minDCF
Linear	Normalized	0.01	9	0.373	0.336
Quadratic	Normalized	0.001	10	0.312	0.307

Table 11: Calibrated scores for logistic regression

While the DCF lowered for the quadratic model, it increased for the linear one.

3.2 Optimizing Gaussian Mixture Models

For GMMs we decided inspect both raw and normalized features.

Type	DCF	DCF_{min}	PCA	# Comp.
Raw	**0.347**	**0.308**	No	2
Norm	0.365	0.311	6	3
Norm	0.348	0.317	No	2

Table 12: GMM Optimization Results

As in table 4, raw and normalized version perform similarly, so both could be of use, depending on the application.

3.3 Optimizing SVMs

3.3.1 Optimizing Linear SVMs

For Linear SVMs, we search the best value of K (bias) and C (boundary) that give the minimum DCFvalue. Different combinations of values for $K = [0, 100]$ and $C = [0.1, 10]$ were tried.

Type	PCA	K	C	DCF	DCF_{min}
Norm.	9	10	0.1	0.350	0.330

Table 13: Best Hyperparameters for Linear SVM

Given the difference between DCF and minimum DCF, we also try calibration:

Type	PCA	K	C	DCF	DCF_{min}
Norm.	9	10	0.1	**0.345**	0.330

Table 14: Calibration for Linear SVM

The DCF lowered, even if not significanlty.

3.3.2 Optimizing Polynomial SVMs

Type	DCF_{min}	PCA
Raw	0.380	9
Norm.	**0.367**	No
Whit.	0.381	No

Table 15: Polynomial K. Optimization Results

Opposed to table 6, we find out that the normalization is actually better than the raw features.

However, our best performing polynomial model is of degree $c = 1$, a linear model. This does not surprise given the effectiveness of other linear models, but we won't further discuss polynomial kernels.

3.3.3 Optimizing RBF SVMs

Type	DCF_{min}	PCA
Norm.	0.288	10
Whit.	0.488	No

Table 16: RBF K. Optimization Results

Whitening has no benefits, we discard it.

4 Experimental Results

After running our models to find the best parameters, we now test them on previously unseen data.

All the parameters here reported have been found through a K-Fold Cross-Validation approach. The value of K used differs with the model, from low values (3, heavier models, e.g. Kernel SVMs) up to higher ones (10, lighter models, e.g. Gaussian Classifiers).

As mentioned earlier, up to now all models have been optimized for a balanced application ($\pi_T = 0.5$). We will now examine how these models perform under different assumptions, and what can be done to get better predictions.

To do so, we compare our DCF with the ones obtained from:

- The best threshold for the test set (MinDCF)
- The best threshold for the training set (DCF Validation)
- The theoretical best threshold and calibrating the scores (DCF Log. Reg.)

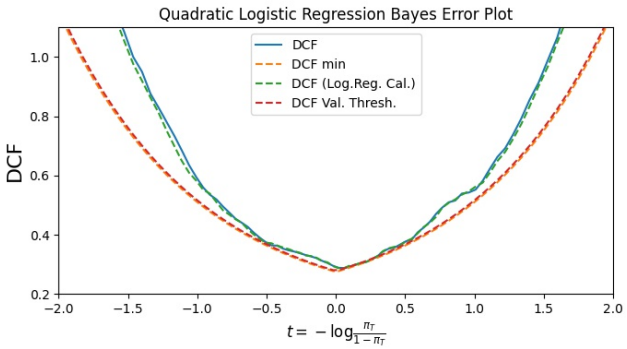


Figure 5: Quadratic LR BEP

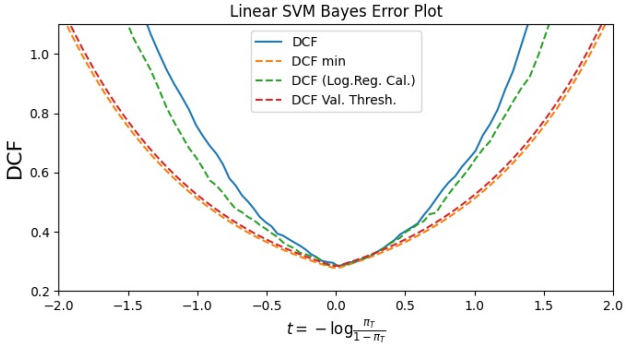


Figure 6: Linear SVM BEP

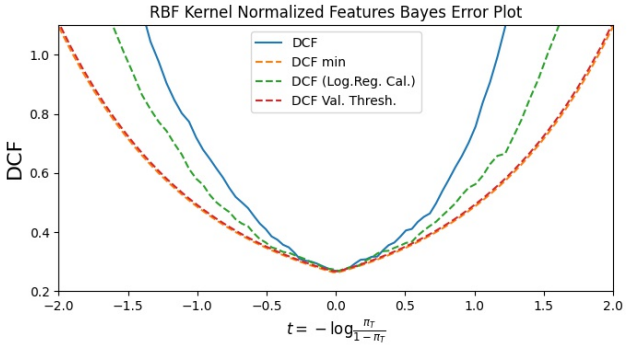


Figure 7: RBF Kernel BEP

Recalibrating scores helps most of the time, but our best option is using a threshold computed from the validatiion scores.

Note that we trained our model and selected the hyper-parameters for a balanced case: normally we should also tune them considering

Model	Calibration	Features	PCA	Hyper-Parameters	DCF	DCF_{min}
Linear LogReg	No	Normalized	9	$\lambda = 0.01$	0.303	0.299
Quadratic LogReg	Yes	Normalized	10	$\lambda = 0.001$	**0.285**	0.274
Linear SVM	Yes	Normalized	9	K = 10, C = 0.1	**0.276**	0.270
MVG Tied Cov.	No	Normalized	5	-	0.390	0.331
GMM	No	Raw	No	$N_{Comp} = 2$	0.327	0.294
RBF Kernel	No	Normalized	10	$\gamma = 0.05, C = 1.5, \xi = 0.1$	**0.271**	0.261

Table 17: Final Parameters and Results on Test Set

the target application, meaning we should have trained the models differently for different applications. In that case, score recalibrating may have been less important.