Wine Project Report

Ruggero Nocera (SXXXXX1)
Quarta Matteo (SXXXXXX)

Abstract This paper is an analysis of the effectiveness of the various models studied during the course applied to a binary classification model. The dataset is a transformed version of the one provided in *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.), where grades span from 0 to 10. Grade 6 has been removed, grades above 6 have been mapped to class 1 and grades below 6 to class 0. Due to limited time and computanional power, results may be just close-to or far-from optimal, depending on the time required to train a model.

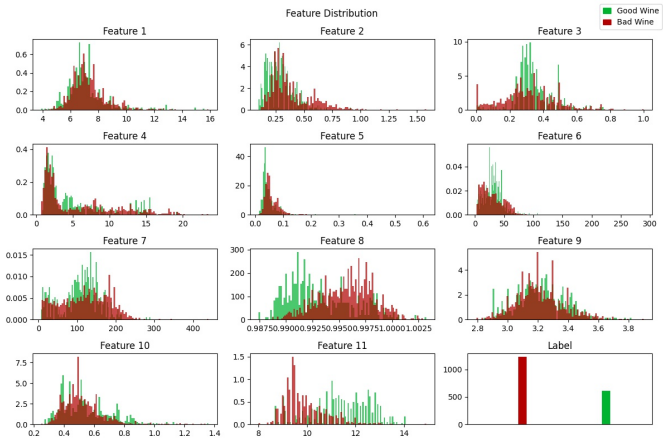
Contents

1 Preliminary Data Analysis

1.1 Feature Distribution

Before discussing the model, 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 one, but keep in mind that in all pictures red color is associated to class 0 (which we will be referring to as class *Bad*) and 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 $\pi \approx .33$ 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 .

Coming back to features distributions, some things are to be noticed. While some features are similar between class Good and class Bad (see feature *1*) others differ substantially and can be very helpful in discriminating samples (see feature *8*).

Moreover, the features are distributed in various ways: while some do look pretty Gaussian (see feature *9*) and others are instead fairly regular (see 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 now project the data on the two dimensional plane. We will be using 3 methods to do so: PCA[3], Normalization + PCA, Normalization + Whitening.

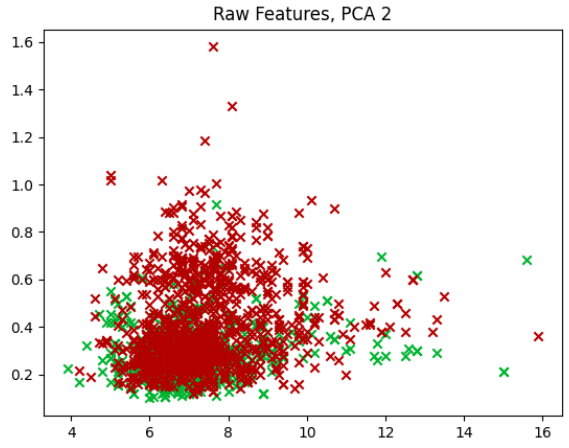


Figure 2: 2D-PCA Projection

[1]K varies through models. For fast ones, 5 or 10 is used. For slower ones, 3 is used.

[2]Meaning both Gaussian Classifiers and Gaussian Mixture Models

[3]Without data centering to appreciate the correlation

The points are very close one another, we thus expect linear models to be not so effective compared to others. The data is pretty *circularly* distributed, so correlation may not play an important role in discriminating samples.

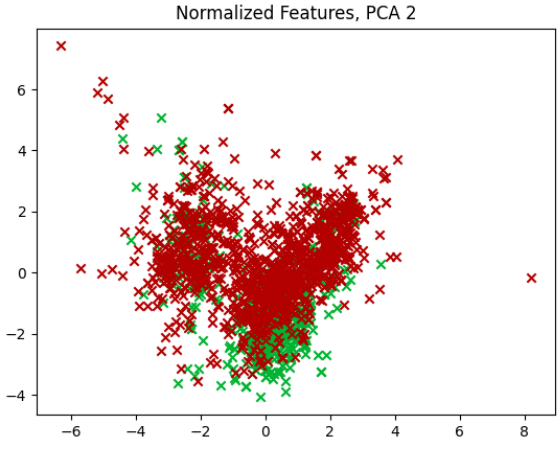


Figure 3: 2D-PCA Projection, Normalized Data

The normalized projection seems to split data in two clusters, each containing some samples of either class, but some points of different class seem to get far apart from the other. So normalization is a technique worth trying.

Note that for Normalization we refer to *Z-Normalization*. From some early tests we found that *Min-Max* normalization was not very effective, and with *Gaussianization* centering and scaling data in the same way as Z while being slower, we decided to use this method.

Lastly we take a look at normalized and whitened data.

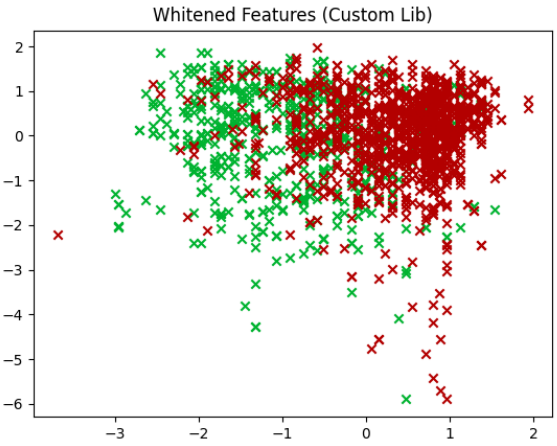


Figure 4: 2D-PCA Projection, Whitened Data

Results do look interesting: points to get much apart, even if we can spot many outliers. With the distribution being this way, we could expect even linear models to achieve decent results.

2 Pre-Processing Analysis

In this section we will run some dummy[4]models to infer wheter pre-processing, by the means of PCA, normalization, etc. can be useful or worth trying.

For now we will run all possible combinations of preprocessing (Raw, Normalized, Whitened) with PCA reductions (2-PCA, 3-PCA, ...).

2.1 Pre-Processing MVG Classifiers

Type	PCA	DCF	minDCF
Raw	9	0.401	**0.362**
Normalized	7	0.464	0.420
Whitened	2	0.430	0.410

Table 1: Full-Covariance MVG - Best Results

[4]Referring to models requiring parameter tuning or score recalibration

Type	PCA	DCF	minDCF
Raw	7	**0.375**	0.366
Normalized	9	**0.375**	0.371
Whitened	11	0.402	0.401

Table 2: Tied-Covariance MVG - Best Results

Type	PCA	DCF	minDCF
Raw	7	0.383	0.366
Normalized	10	0.429	0.391
Whitened	5	0.402	0.386

Table 3: Naive-Bayes MVG - Best Results

The result validate some of our assumptions and invalidate others. By looking at the DCF[5]values, our best model seems to be the Tied-Covariance ones. We are not very surprised to se that the Naive Bayes does sometimes outperform the Full-Covariance model as we noticed in Figure 2 that some features are not very correlated. The same can said about the the Tied covariance by looking at the same projection or feature *8* of Figure 1.

Normalization and Whitening do not help, they harm, sometimes even significantly the classification, while it looks like PCA can be of modest help.

An interesting fact that is that while the Full-Covariance model looks like the worse perfoming one, it is the model with the lowest minDCF values. While it is not an important difference compared with other models, it is the one with the biggest difference between the DCF and minDCF values, hinting that score calibration could be required.

2.2 Pre-Processing for GMMs

For GMMs we decide the number of components (for this dummy execution 4 was picked arbitrarily). We start with identity covariance matrices and by placing the means near the dataset mean.[6]

[5]By DCF we actually mean the *normalized* DCF, considering a prior $\pi = .5$ and equal costs

[6]The starting points are used in the same way also in the optimization part.

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

As we would expect after the results of the Gaussian Classifiers, also here whitening does not look like a good choice and normalization looks ineffective. High dimensionality still looks preferred.

The DCF values are close to the minDCF values, score calibration may be not necessary.

2.3 Pre-Processing for Polynomial Kernel SVMs

The polynomial kernel mapping $\phi(\cdot)$ we will use is so defined:

$$\phi(\mathbf{x}) = \begin{bmatrix} vec(\mathbf{x}\mathbf{x}^T) \\ \sqrt{2}\mathbf{x} \\ 1 \end{bmatrix} \tag{1}$$

And we will be using the kernel function so described:

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

This formulation would require us to strictly use $c = 1$, as we do in our dummy model, but later on this will be an hyperparameter to be optimized.

The results are interesting:

Type	PCA	DCF	minDCF
Raw	7	0.554	0.545
Normalized	6	**0.393**	**0.381**
Whitened	11	0.399	0.393

Table 5: Polynomail Kernel SVM - Best Results

While competitiveness with other models has to be made later, when each one will be fully used, here we see an interesting change: normalization greatly improves our classification. While a lower PCA has obtained the lowest DCF, it is not significantly lower than the one obtained with higher dimensionality, so we can say that here it's normalization helping us out. Whitening, again, does not make any difference, as the results are same or close to the ones obtained with normalization only.

Notice also how scores seems well calibration with our theoretical threshold even if they do not have a probabilistic interpretation.

2.4 Pre-Processing for RBF Kernel SVMs

The dummy RBF function used here is the following:

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

Small values of γ and b were picked to avoid numerical issues when using non-normalized models.

Type	PCA	DCF	minDCF
Raw	10	0.588	0.579
Normalized	11	**0.356**	**0.352**
Whitened	11	**0.356**	**0.352**

Table 6: RBF Kernel SVM - Best Results

Just like polynomial kernel, RBF prefers high dimensionality , so further testing in this direction is required. Again, normalization plays an important role while whitening seems to have no effect whatsoever and scores look extremely well calibrated.

Just like polynomial kernel SVMs, here normalization has no effect whatsoever whilte normalization greatly improves our performance. The RBF Kernel works significantly better when working with all or most components.

We are not yet discussing perfomance in detail but so far, this is our most promising model.

2.5 Pre-Processing for Logistic Regression Model

Regarding the Logistic Regression, we will consider both linear and quadratic model.

2.5.1 Linear Logistic Regression

In the Linear model, dimensionality reduction through PCA and prepocessing through normalization helped to obtain lower minimum DCF compared to raw features. As the Gaussian Classifiers, only a small number of features can be removed.

Only results with already optimized hyperparameters are reported.

Type	PCA	DCF	minDCF
Raw ($\lambda = 0.01$)	10	0.369	0.342
Normalized ($\lambda = 0.0001$)	9	0.369	**0.336**
Whitened ($\lambda = 0.001$)	6	0.554	0.537

Table 7: Linear Logistic Regression - Best Results

Whitening didn't help so much to improve results, so it won't be considered in further analysis.

2.5.2 Quadratic Logistic Regression

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

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

In contrast to linear model, whitening helped as well as normalization, PCA was useful in both cases, with and without preprocessing, but with a notable difference in the number of features used to obtain following results.

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

Table 8: Quadratic Logistic Regression - Best Results

2.6 Pre-Processing for Linear SVM

The Linear SVM model performed better with normalization and whitening rather than raw features. Applying PCA to raw features and whitened ones was useful to obtain low values of minimum DCF, in contrast to normalization which performed better without dimensionality reduction.

Type	K	C	PCA	DCF	minDCF
Raw	100	0.1	8	0.369	0.363
Normalized	0.1	1.0	11	0.344	**0.336**
Whitened	1.0	0.1	10	0.366	**0.336**

Table 9: Linear SVM - Best Results

Optimizing Models

3.1 Optimizing MVG Classifiers

Since the MVG Classifiers we will be only briefly discussing score calibration. In particular we will focus on Full-Covariance and Naive Bayes MVG Classifiers, as Tied Covariance already is very close to being optimal.

The scores will be rebalanced using logistic regression as it follows:

$$f(s) = \alpha s + \beta \tag{2}$$

The scores are the ones that produced the best result for both models as in table 1 and 3.

minDCF	DCF_{Before}	DCF_{After}
0.362	0.401	**0.390**
0.367	**0.383**	**0.412**

Table 10: MVG Recalibrated Scores DCFs

Recalibration heavily damages our Naive Bayes classifiers so it's not a good choice. The Full Covariance one has a modest DCF decrement, but it's still worse than the plain Naive Bayes one. Lastly, by looking at table 2, we notice that both models perform worse than our Tied Covariance one, so we will be discarding them in our final evaluation.

3.2 Optimizing Gaussian Mixture Models

For GMMs we need to optimize the number of subcomponents each density has. We should also inspect normalization more and try recalibrating scores as we did for gaussian classifiers with Gaussian Classifiers.

DCF	Type	# Components	PCA
0.346	Norm.	3	No
0.348	Raw	3	10
0.359	Raw	2	No
0.360	Raw	3	No
0.365	Raw	2	9
0.375	Norm.	3	8
0.377	Norm.	5	9
0.378	Norm.	3	10

Table 11: GMM Optimization Results

Our two top models are very similar in some ways, like using 3 components, high dimensionality and DCF-wise, but differ for Normalization.

Type	minDCF	DCF_{Before}	DCF_{After}
Norm.	0.316	0.346	**0.324**
Raw	0.328	0.348	**0.334**

Table 12: GMM Optimization Results

Contrary to what we have seen in table 4, here the scores are fairly distant from the minimum possible DCF, and recalibrating the scores did indeed help.

The normalized GMM is the best one, but data does not suggest that in general normalized GMMs are better, so both model will be used later on.