

Wine Project Report

Ruggero Nocera (SXXXXXX1)

Quarta Matteo (SXXXXXXX)

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 computational 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

2 Pre-Processing Analysis

2.1 Pre-Processing MVG Classifiers . .

2.2 Pre-Processing for GMMs

2.3 Pre-Processing for Polynomial Kernel SVMs

2.4 Pre-Processing for RBF Kernel SVMs

2.5 Pre-Processing for Logistic Regression Model

2.5.1 Linear Logistic Regression . .

2.5.2 Quadratic Logistic Regression

2.6 Pre-Processing for Linear SVM . .

3 Optimizing Models

3.1 Optimizing MVG Classifiers

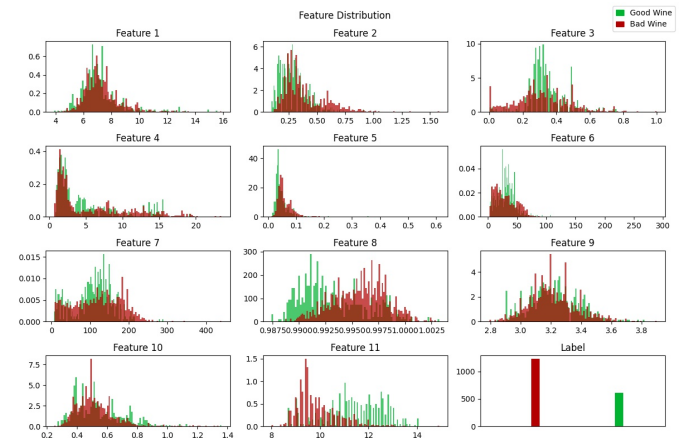
3.2 Optimizing Gaussian Mixture Models

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-Fold¹Validation 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 models², 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: PCA³, Normalization + PCA, Normalization + Whitening.

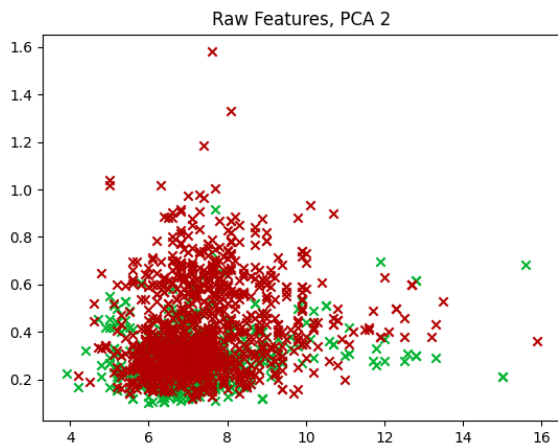


Figure 2: 2D-PCA Projection

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

²Meaning both Gaussian Classifiers and Gaussian Mixture Models

³Without 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.

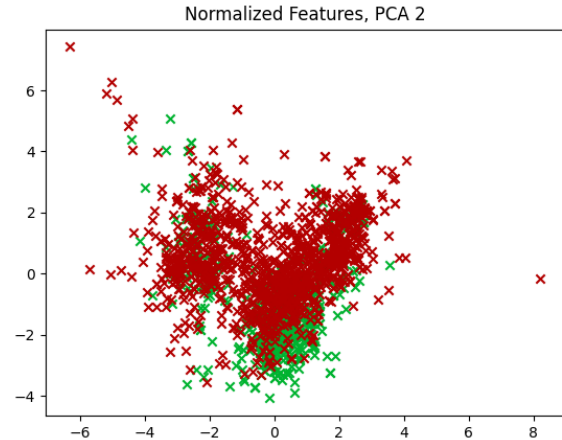


Figure 3: 2D-PCA Projection, Normalized Data

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.

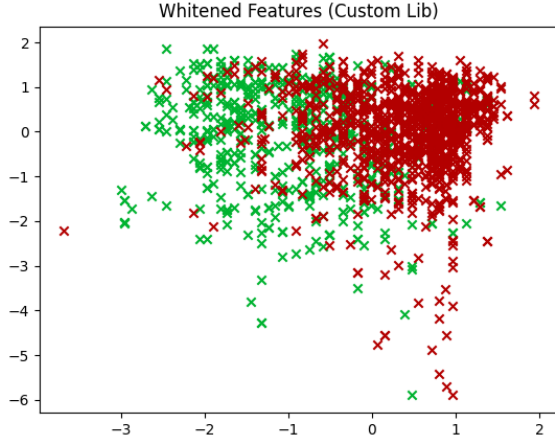


Figure 4: 2D-PCA Projection, Whiten Data

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 dummy⁴ models to learn how pre-processing affects the models for our task.

Note that we will be discarding combinations or entire models targeting our final application. A discussion about different applications will be made in the ending.

2.1 Pre-Processing MVG Classifiers

Since MVG classifiers are fairly similar, we will now pick the most promising one and discarding the others, while trying to infer its optimal working condition.

⁴Simple model that run with arbitrary, non-optimized hyper-parameters

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 seem in line with our assumptions. In fact, if we look 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 we will keep it, and removing some dimensions seems to be helpful.

Notice that while the Tied Covariance model achieved the best DCF⁵ of all classifiers, the MVG achieved the lowest Minimum DCF values, hitting that it is actually the one out of the three that could theoretically best separate classes.

Normalization and Whitening look irrelevant so we won't further experiment them.

2.2 Pre-Processing for GMMs

For our dummy GMM model we arbitrarily set the number of components to 4. The starting

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

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

As for our Gaussian Classifiers, Whitening and Normalization do not look promising. However, as the difference between the Raw and Normalized does not look significant, we will further test both models. This first test shows us that high dimensionality is actually preferred.

2.3 Pre-Processing for Polynomial Kernel SVMs

The polynomial kernel function is defined as:

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

We proceed to set our parameters arbitrarily for this dummy execution:

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

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 while normalization greatly improves our performance. The RBF Kernel works significantly better when working with all or most components.

We are not yet discussing performance 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 preprocessing 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

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

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

3 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:

$$f(s) = \alpha s + \beta \quad (1)$$

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.