

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

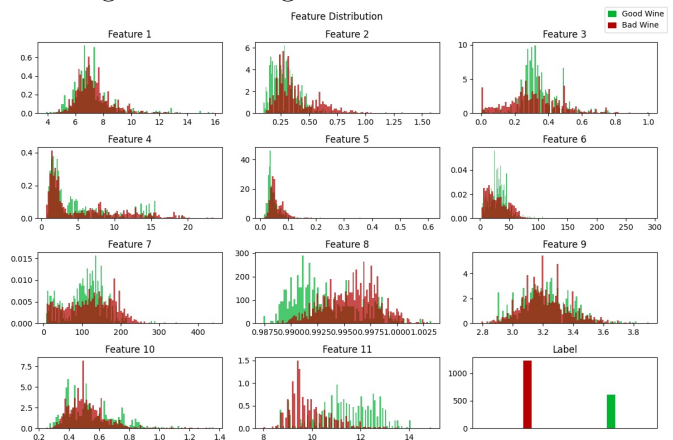
| | |
|--|----------|
| 1 Preliminary Data Analysis | 1 |
| 1.1 Feature Distribution | 1 |
| 2 Pre-Processing Analysis | 3 |
| 2.1 Pre-Processing MVG Classifiers . . | 3 |
| 2.2 Pre-Processing for GMMs | 3 |
| 2.3 Pre-Processing for Kernel SVMs . . | 4 |
| 2.3.1 Pre-Processing for Polynomial Kernel SVMs | 4 |
| 2.4 Pre-Processing for RBF Kernel SVMs | 4 |
| 2.5 Pre-Processing for Logistic Regression Model | 5 |
| 2.5.1 Linear Logistic Regression . . | 5 |
| 2.5.2 Quadratic Logistic Regression | 5 |
| 2.6 Pre-Processing for Linear SVM . . | 5 |
| 3 Optimizing Models | 5 |
| 3.1 Optimizing Logistic Regression . . | 5 |
| 3.2 Optimizing Gaussian Mixture Models | 6 |
| 3.3 Optimizing Kernel SVMs | 6 |
| 3.3.1 Optimizing Polynomial SVMs . . | 6 |
| 3.3.2 Optimizing RBF SVMs . . . | 6 |
| 4 Experimental Results | 6 |

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.

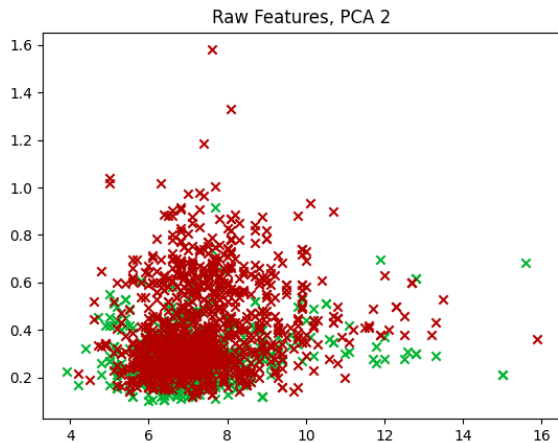


Figure 2: 2D-PCA Projection

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.

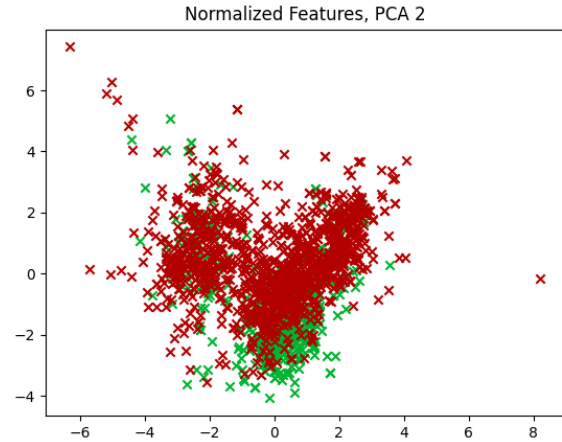


Figure 3: 2D-PCA Projection, Normalized Data

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.

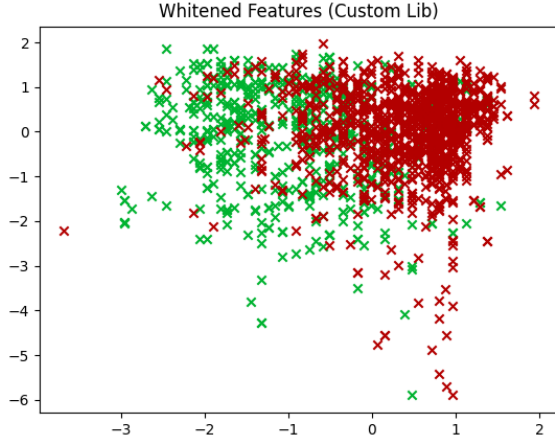


Figure 4: 2D-PCA Projection, Whiten Data

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 Kernel 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 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: Polynomial 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. For the whitened features, best results are obtained without applying PCA. 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 ($\lambda = 0.0001$) | 10 | 0.369 | 0.342 |
| Normalized ($\lambda = 0.01$) | 9 | 0.369 | 0.336 |
| Whitened ($\lambda = 0.0001$) | 11 | 0.549 | 0.503 |

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 much better with normalization and whitening rather than raw features. Applying PCA was useful to obtain low values of minimum DCF in all cases, even if the model trained with raw features performed worse than models trained with preprocessed features.

| Type | K | C | PCA | DCF | minDCF |
|------------|-----|------|-----|-------|--------------|
| Raw | 0 | 10.0 | 6 | 0.911 | 0.568 |
| Normalized | 10 | 0.1 | 9 | 0.350 | 0.330 |
| Whitened | 1.0 | 1.0 | 9 | 0.364 | 0.331 |

Table 9: Linear SVM - Best Results

The high difference between minimum DCF and DCF suggests us that scores are miscalibrated, but we will try to optimize it in next chapters.

3 Optimizing Models

3.1 Optimizing Logistic Regression

Now we will try to find the λ that gives us the best (lower) minimum DCF in the validation set, extracted from the training set through a 5-fold cross validation protocol. Several values of λ have been tried, in a range between $[10^{-4}, 10^3]$ for both linear and quadratic model.

| Model | Type | λ | PCA | DCF | minDCF |
|-----------|------------|-----------|-----|-------|--------|
| Linear | Normalized | 0.01 | 9 | 0.369 | 0.336 |
| Quadratic | Normalized | 0.001 | 10 | 0.324 | 0.307 |

Table 10: Best λ values for Logistic Regression

We tried to lower the difference between the DCF and the minimum one, in both linear and quadratic model. To reduce the DCF, we tried to calibrate the scores:

| 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

As reported in table 11, calibration helped in the quadratic case, but DCF got worse in linear one.

3.2 Optimizing Gaussian Mixture Models

For GMMs we decided inspect both raw and normalized optimization.

| 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 both perform similarly, so both model could be of use, depending on the application.

3.3 Optimizing Kernel SVMs

Differently from other models, for SVMs we will choose the best parameter based on only the minimum DCF values, given the non probabilistic nature of the scores generated.

3.3.1 Optimizing Polynomial SVMs

| Type | DCF _{min} | PCA |
|-------|--------------------|-----|
| Raw | 0.380 | 9 |
| Norm. | 0.367 | No |
| Whit. | 0.381 | No |

Table 13: Polynomial K. Optimization Results

As opposed to table 5, we find out that the normalized version is actually better than the raw features one.

However, our best performing polynomial model is of degree $c = 1$, in other words a linear model. This is not surprising given the effectiveness of the Linear Logist Regression model, but we won't further discuss it.

3.3.2 Optimizing RBF SVMs

| Type | DCF _{min} | PCA |
|-------|--------------------|-----|
| Norm. | 0.288 | 10 |
| Whit | 0.00 | X |

Table 14: RBF K. Optimization Results

Here we determine that whitening has no more benefits than plain normalization, so we ultimately discard it.

4 Experimental Results

After having run our models in order 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 spans from 3 (heavier models, e.g. Kernel SVMs) up to 10 (lighter models, e.g. Gaussian Classifiers).

| 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 |

As mentioned earlier, up to now all models have been evaluated considering a balanced application ($\pi_T = 0.5$). We will now examine how these models perform under different applications, where the class distribution is unbalanced, and how they can be tweaked to perform better under these circumstances.

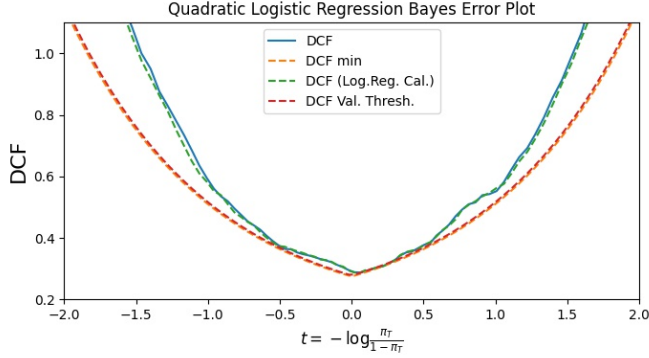


Figure 5: Quadratic LR BEP

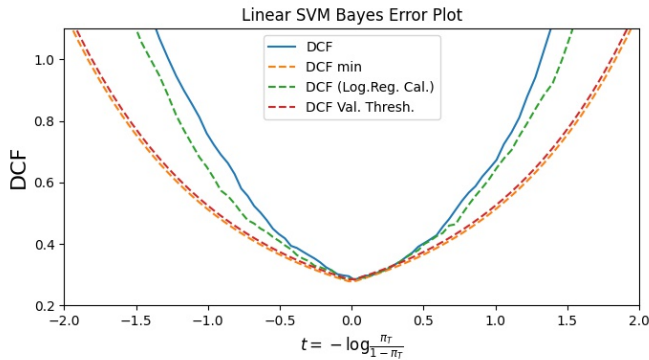


Figure 6: Linear SVM BEP

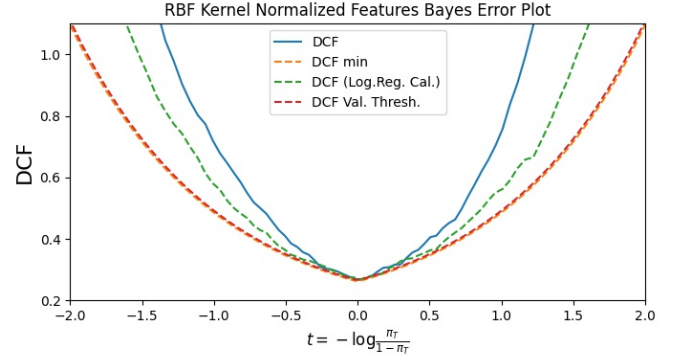


Figure 7: RBF Kernel BEP