Wine Project Report

Ruggero Nocera (SXXXXX1)
Quarta Matteo (SXXXXXX)

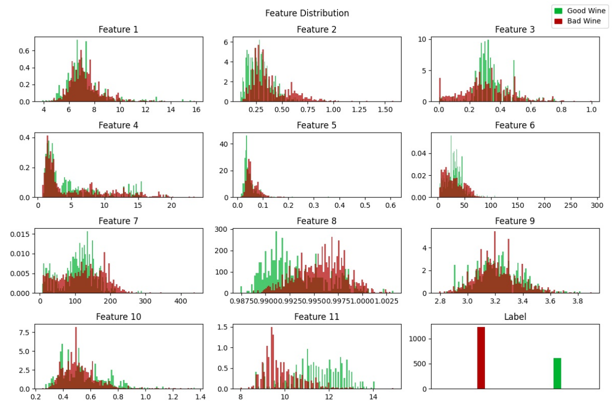
Contents

1 Preliminary Data Analysis

1.1 Feature Distribution

Before discussing the varius models and techniques that can give us robust result, we shortly discuss how the feature of our dataset are distributed.

Figure 1: Histogram of Class' Features



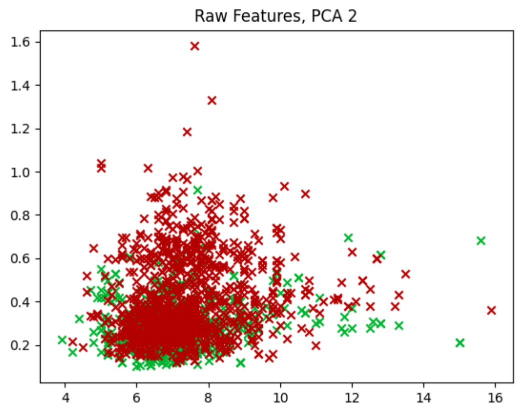
We start by noticing that the order of magnitude of some features can differ much from one another: for example, if we look at feature 5, it seems to span from 0.0 to as much as 0.6, but feature 7 instead spans two whole orders, occasionally being over 200, and with some outliers even double as much. An interesting observation is that some features do look gaussian distributed or regular enough to be described by a combination of gaussian distributions.

This may hint that some dimensionality reduction will be strongly biased towards features like 7, and that normalization can be taken as a valid pre-processing technique.

We also notice that our dataset is unbalanced: the quantity of bad wine is roughly twice the quantity of good one. This has to be taken into account when developing models, but we will be trating them as if the dataset is balanced, that is considering and equal probability of good wine and bad wine, occasionally rebalancing scores.

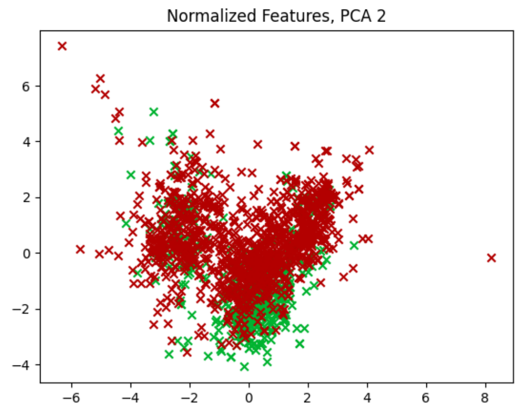
Let's apply a 2D projection of our data to further discuss it.

Figure 2: 2D-PCA Projection



The features look highly correlated and fairly regularly distributed: this may be a hint that some gassuain models may perform well enough, like Full-Covariang MVG Classifiers or Tied-Covariance ones, but some other with linear decision boundaries, like linear SVMs, may perform poorly.

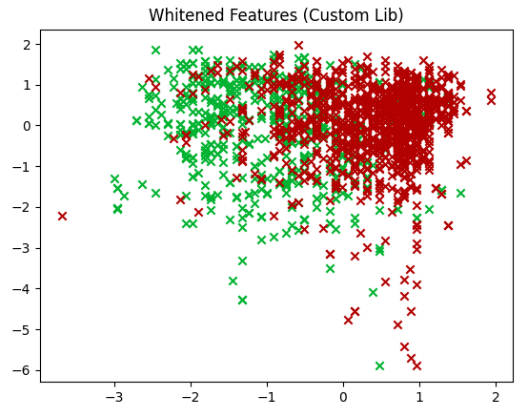
Figure 3: 2D-PCA Projection, Normalized Data



After normalization (Z-Normalization, mean $\mu = 0$ and St.D. $\sigma = 1$) our points do get apart but not significantly: we could expect linear models to still perform poorly and gaussian classifiers to get slightly worse. On the other hand, Gaussian Mixture Models look like a more interesting choice.

Lastly we take a look at normalized and whitened data.

Figure 4: 2D-PCA Projection, Whitened Data



Here the results are a bit more interesting: points do get separeted just like after normalization, but same class points stay close to each other while different class get more apart. In this scenario, it looks like even linear model might have a chance to achieve a decent performance.

2 Pre-Processing Analysis

2.1 Pre-Processing for Gaussian Models

We will shortly decide, by running some non-optimized models, which pre-processing techniques are useful and which are not.

We will be reporting the best results for each classifiers, extracted from a 3 or 5 Fold Validation approach. The decision rule, since this is a binary problem, will be set to $t = \log(\frac{\pi_T}{1-\pi_T}) \lesseqgtr 0$, where π_T is assumed $= 0.5$ and it is the prior probability of class Good.

Let's start with Gaussian Classifiers: Full-Covariance, Tied-Covariance, and Naive-Bayes.

PCA	DCF	minDCF
9	0.401	0.362
8	0.403	0.373
7	0.426	0.406
5	0.427	0.413
No	0.465	0.389

Table 1: Full-Covariance MVG - PCA

PCA	DCF	minDCF
7	0.375	0.366
8	0.384	0.366
9	0.387	0.377
10	0.388	0.377
No	0.402	0.401

Table 2: Tied-Covariance MVG - PCA

PCA	DCF	minDCF
7	0.383	0.366
8	0.393	0.387
6	0.401	0.393
9	0.412	0.391
No	0.442	0.404

Table 3: Naive-Bayes MVG - PCA

We find out that a small number of features can be removed, with modest gains.

Let's combine these results with normalization and whitening.

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

Table 4: Full-Covariance MVG - Best Results

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

Table 5: 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 6: Naive-Bayes MVG - Best Results

Now we can finally say something on MVG Classifiers:

- As expected, the Tied Covariance model works fairly well
- Unexpectedly, the Naive-Bayes model outperforms the Full-Covariance one, even without whitening
- Normalization and whitening seem to hurt performance
- PCA can be used, but gains are not particularly high

2.2 Pre-Processing for GM Models

Staying of the subject of Gaussian Classifiers, we make similar operations for GMMs. This is a dummy 4-Component model.

PCA	DCF	minDCF
No	0.410	0.394
10	0.418	0.410
9	0.431	0.425
7	0.436	0.420
No	0.410	0.394

Table 7: GMM - PCA

Results seem on par, not significantly lower than MVG Classifiers, but GMM actually performs best when keeping as much dimensions as possible.

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

Table 8: GMM - Best Results

Also here, normalization and whitening seem to do more bad than good. Results do not get much worse, sometimes even get better but not significantly.

Our best option is to work with raw features on GMMs. Weirdly enough, Whitened GMM can discard many components and still perform fairly well compared to the raw counterpart.

In the next two subsections we will discuss preprocessing on two types of kernel SVMs: Polynomial and RBF ones. Given the non-probabilistic nature of the scores, using a threshold of 0 is somewhat an arbitrary choice: to account for that, we will be giving a greater importance to minimum DCFs rather than plain DCFs.

2.3 Pre-Processing for Polynomial Kernel SVMs

As usual we start by looking at the effect of PCA on a dummy Polynomial Kernel. The dummy kernel used is the following:

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

With $\phi(\cdot)$ being:

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

[1]

Here are the results:

PCA	DCF	minDCF
No	0.648	0.642
10	0.666	0.663
2	0.666	0.664
6	0.666	0.666
No	0.648	0.642

Table 9: Polynomial Kernel SVM - PCA

Generally speaking, results look far from good (the DCFs are much higher than Gaussian Models) and PCA seems to have no effect whatsoever.

Type	PCA	DCF	minDCF
Raw	11	0.648	0.642
Normalized	10	0.666	0.653
Whitened	11	0.666	0.659

Table 10: Polynomail Kernel SVM - Best Results

Here normalization and whiteining seem to have little to no effect: our DCFs are the same, even thought our scores vary significantly.

Moreover, since we are enforcing $\pi_T = 0.5$, this means that we are achieving $\approx 33\%$ error rate, which is the same we would get by guessing always Bad (on our Training/Validation set, that is). This may hint that polynomial kernel SVMs are just bad suited for this task.

2.4 Pre-Processing for RBF Kernel SVMs

Just like we did for the polynomial one, here we try again a dummy kernel to see how we should pre-

[1] Above we freely set c as it is the final derivation of this mapping, but the original kernel as desccribed here would require it to be 1

process our data. The dummy kernel function used is:

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

Small values of γ and b were picked to avoid numerical issues.

PCA	DCF	minDCF
10	0.585	0.579
No	0.585	0.579
5	0.585	0.579
4	0.613	0.606
3	0.655	0.647
2	0.682	0.651

Table 11: RBF Kernel SVM - PCA

The RBF seems to be robust while working with all features or almost, and start behaving worse when the number of dimensions gets much lower.

Given the non-optimed hyperparameters, trying various combination of hyperparameters with high dimensionalities is needed to draw any conclusions.

Type	PCA	DCF	minDCF
Raw	10	0.585	0.579
Normalized	No	**0.469**	**0.431**
Whitened	No	**0.469**	**0.431**

Table 12: RBF Kernel SVM - Best Results

For once normalization seems to help: the DCFs get significantly lower. Whitened features perform on par with normalized features, but given that before whiteining features are also normalized, we can say that also in this case whiteining has no effect whatsoever.

λ	Error Rate
0	16.026%
1e-09	16.081%
1e-06	16.081%
0.001	15.587%
0.1	18.057%

Table 13: Logistic Regression

λ	PCA	Error Rate
0.001	10	15.587%
0	8	15.862%
1e-09	8	15.862%
1e-06	8	15.862%
0.001	8	15.917%
0	10	16.081%
1e-09	10	16.081%
1e-06	10	16.136%
0	6	16.630%
1e-09	6	16.630%

Table 14: Logistic Regression With PCA

λ	PCA	Error Rate
0.001	10	15.038%
0	10	15.258%
1e-09	10	15.258%
1e-06	10	15.258%
0	8	15.532%
1e-09	8	15.532%
1e-06	8	15.532%
0.001	8	15.532%
0.001	6	16.246%
0	6	16.301%

Table 15: Logistic Regression With PCA and Z-Normalization

Bias	C	Error Rate
0	0.1	17.728%
0	1	16.520%
1	0.1	35.565%
1	1	38.090%
5.0	0.1	15.862%
5.0	1	47.805%
10.0	0.1	15.642%
10.0	1	16.081%

Table 16: Linear SVM

Bias	C	PCA	Error Rate
10	0.1	10	15.313%
5	0.1	10	15.642%
5	0.1	6	15.971%
10	0.1	8	16.301%
5	0.1	8	16.356%
10	0.1	6	16.575%
0.1	0.1	8	18.057%
0.1	0.1	6	18.222%
0.1	0.1	10	18.551%
0	0.1	8	18.716%

Table 17: Linear SVM With PCA

Bias	C	PCA	Error Rate
1	0.1	10	14.709%
5.0	0.1	10	14.929%
10.0	0.1	10	14.929%
1	1	10	15.038%
5.0	1	10	15.038%
10.0	1	10	15.038%
1	1	8	15.258%
5.0	0.1	8	15.258%
5.0	1	8	15.258%
10.0	0.1	8	15.258%

Table 18: Linear SVM With PCA and Z-Normalization