Politecnico di Torino

Wine Quality Detection

Giulio Corallo, s282380

September 15, 2021

[1]

1 Introduction

The Wine Quality Data Set(1) is taken from the UCI Machine Learning Repository.

The goal is to model wine quality based on physicochemical tests. The original dataset consists of 10 classes (quality 1 to 10), but in this project, the dataset has been binarized, collecting low quality wines (lower than 6) into class 0, and good quality wines (greater than 6) into class 1. Wines with quality 6 have been discarded. The dataset contains both red and white vinho verde wine samples from the north of Portugal.

There are 11 features, that represent physical properties of the wine :

- Fixed Acidity
- Volatile Acidity
- Citric Acid
- Residual Sugar
- Chlorides
- Free Sulfur Dioxide
- Total Sulfur Dioxide
- Density
- pH
- Sulphates
- Alcohol

The training set contains 1839 samples, 1226 of bad quality wines and 613 samples of good quality wines.

The evaluation set contains 1822 samples, 1158 of bad quality wines and 664 samples of good quality wines.

[1]P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

2 Dataset Statistics and Data Exploration

The first step has been an analysis on the dataset to understand the kind of features, their ranges and their distributions.

2.1 Statistics

As reported in the following table, for each feature I extracted the number of elements, the number of null elements, the min and max values, the mean and the standard deviation.

Dataset Statistics

Feature	Count	Null Count	Min	Max	Mean	Std Dev
Fixed Acidity	1839	0	3.9	15.9	7.25881	1.35412
Volatile Acidity	1839	0	0.1	1.58	0.354905	0.170232
Citric Acid	1839	4	0	1	0.316259	0.148248
Residual Sugar	1839	0	0.7	23.5	5.4422	4.74465
Chlorides	1839	0	0.013	0.611	0.0575742	0.0370892
Free Sulfure Dioxide	1839	0	2	289	30.146	19.2233
Total Sulfure Dioxide	1839	0	7	440	116.35	58.0895
Density	1839	0	0.98742	1.0032	0.994863	0.00294148
pH	1839	0	2.79	3.9	3.21395	0.159383
Sulphates	1839	0	0.25	1.36	0.528434	0.142898
Alcohol	1839	0	8	14.9	10.3798	1.22512

2.2 Distributions

We want to visualize the distribution of the different features for the classes "Bad Quality" represented in the dataset as value 0 and "Good Quality" represented as value 1.

For each feature, firstly I extract from the data matrix the parts corresponding to the two classes and then I plot the corresponding histogram for each class using the function matplotlib.pyplot.hist.

I obtained the following figures:

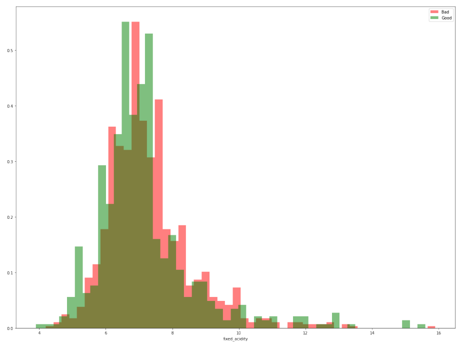


Figure 1: Fixed acidity

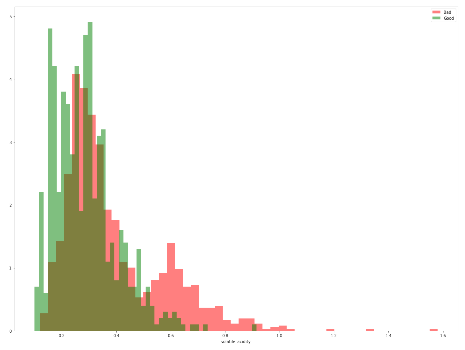


Figure 2: Volatile acidity

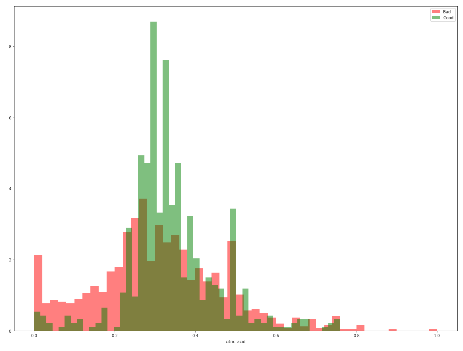


Figure 3: Citric acid

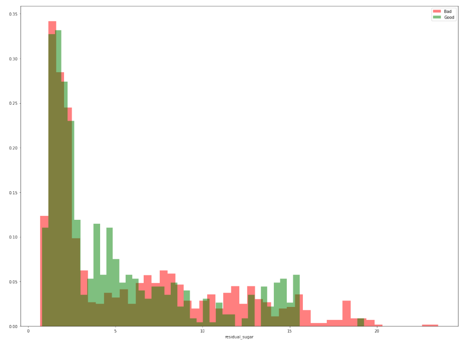


Figure 4: Residual sugar

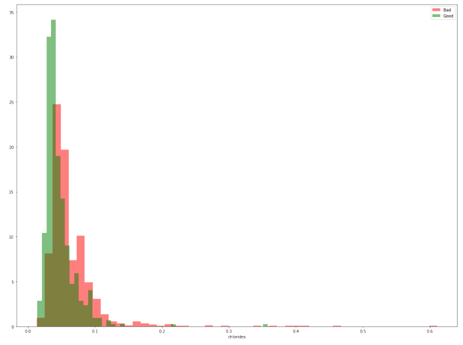


Figure 5: Chlorides

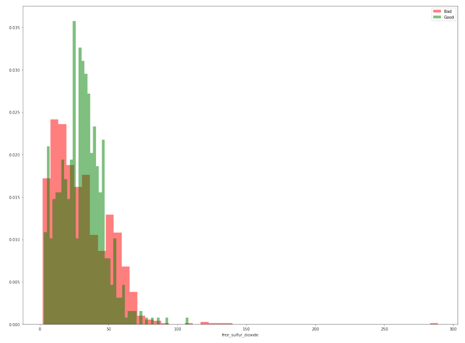


Figure 6: Free sulfur dioxide

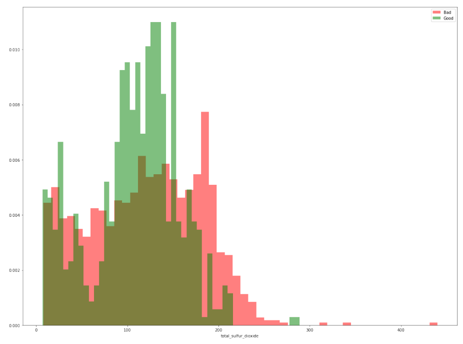


Figure 7: Total sulfur dioxide

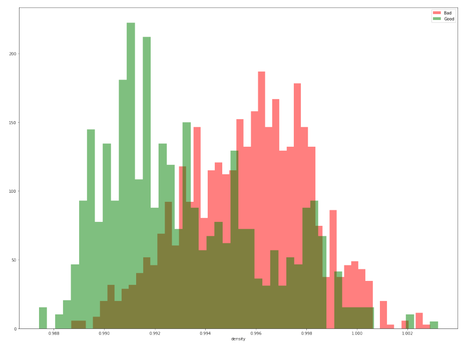


Figure 8: Density

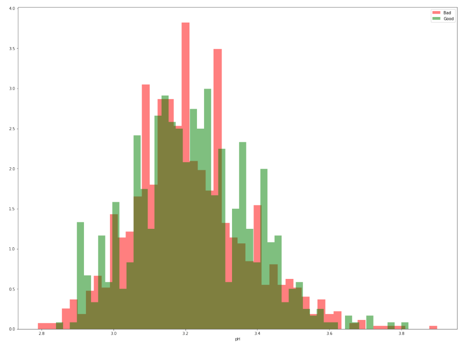


Figure 9: Ph

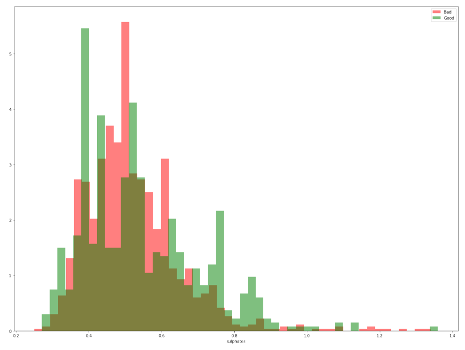


Figure 10: Sulphates

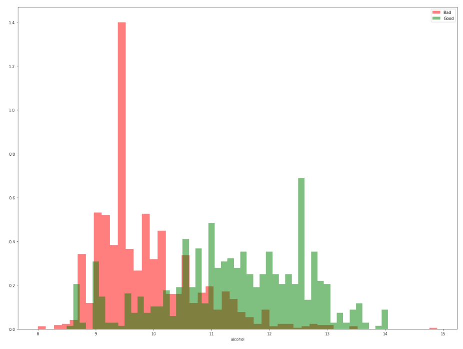
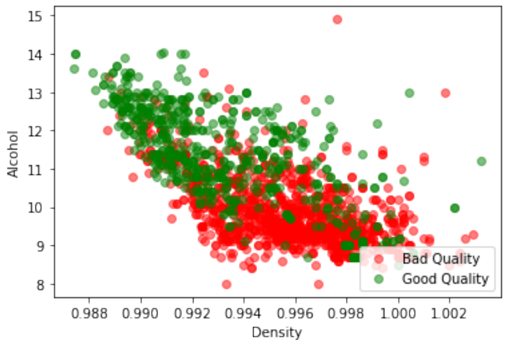


Figure 11: Alcohol

As we can see, for each of the features there is a large overlap for the two classes. Only for the attributes of Alcohol and Density we obtain a slight separation between the two classes, we can therefore conclude that these labels represent two of the most discriminating attributes for our analysis.

Hence, we can visualize the scatter plot between this two most discriminating attributes.



We can observe that for large values of alcohol and small values of density is more likely to find good quality of wine, instead for large values of density and small values of alcohol is more likely to find bad quality of wine.

Another way to show how different the feature ranges are is through boxplots.

In particular I extracted two boxplots: one that shows all the features and one without Free and Total Sulfure Dioxide. Removing these two features, that are the ones with the widest ranges of values, we can show better the other feature values ranges.

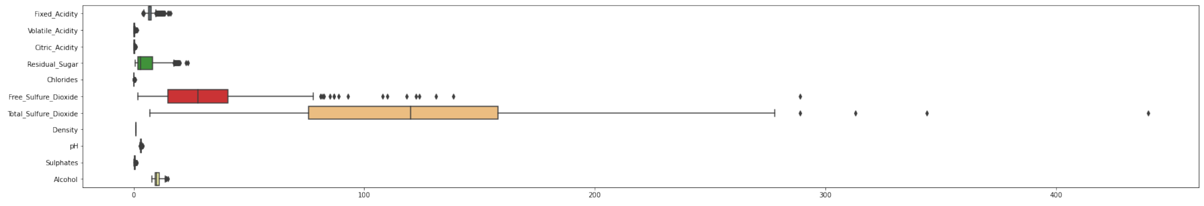


Figure 12: Full Boxplot

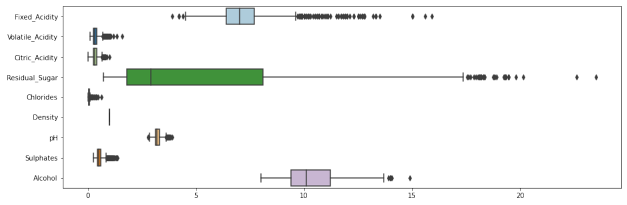


Figure 13: Boxplot without Free and Total Sulfure Dioxide

2.3 Correlations

To show how correlated the features are and to see if we can try to remove some of them in the dimensionality reduction phase, I computed the correlations matrices for the whole dataset but also splitting it in good/bad quality.

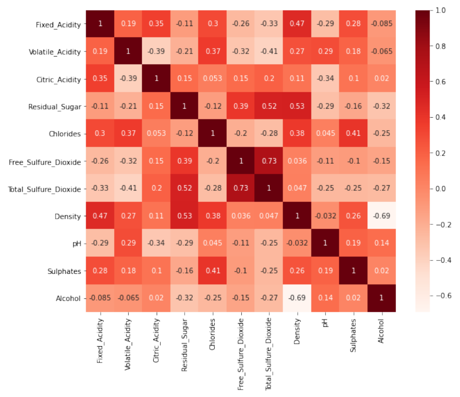


Figure 14: Covariance Heatmap of the whole dataset

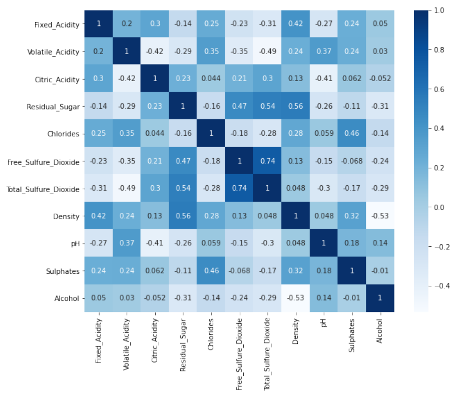


Figure 15: Covariance Heatmap of the Bad Quality samples

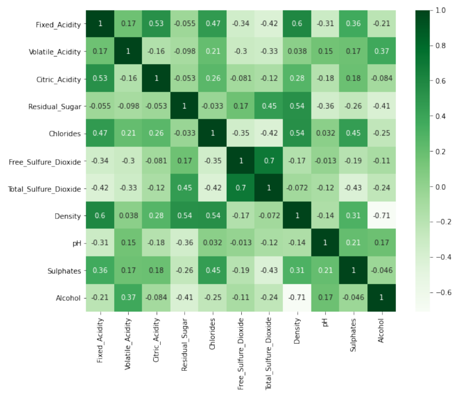


Figure 16: Covariance Heatmap of the Good Quality samples

As shown from the heatmaps, there are two couples of features whose correlation is more than 0.65.

The labels 'Free Sulfure Dioxide' and 'Total Sulfure Dioxide' have a correlation of 0.73 for the whole dataset and the correlation stays strong for both good (0.7) and bad quality (0.74).

The labels 'Density' and 'Alcohol' have a negative correlation of -0.69 for the whole dataset, but it decreases for bad quality wines (-0.53), while it is stronger for good quality wines (-0.71).

3 Gaussianization

As we can see analysing the train dataset, the raw feaures don't show Gaussian Normal distributions. For this reason but also for the presence of outliers it is easy to expect poor results for all the classification approaches and in particular for the Gaussian-based ones.

After all of these considerations, I hence decided to apply a pre-processing method to 'Gausssianize' our dataset. With this approach we aim to map our features and obtain a new set of features whose empirical cumulative distribution function is well approximated by a Gaussian cumulative distribution function.

The features will be mapped to a uniform distribution and then transformed through the inverse of Gaussian cumulative distribution function. The first function I apply is the Rank Function:

$$r(x) = \frac{\sum_{i=1}^{N} I[x < x_i] + 1}{N + 2} \tag{1}$$

and then I transform the features with the inverse of the cumulative distribution function (percent point function) of the standard normal distribution:

$$y = \Phi^{-1}(r(x)) \tag{2}$$

3.1 Distributions

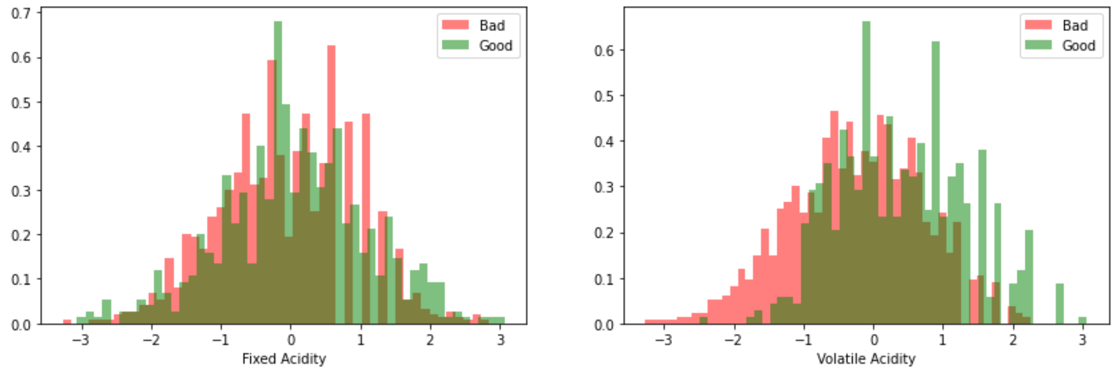


Figure 17: Fixed Acidity and Volatile Acidity distributions

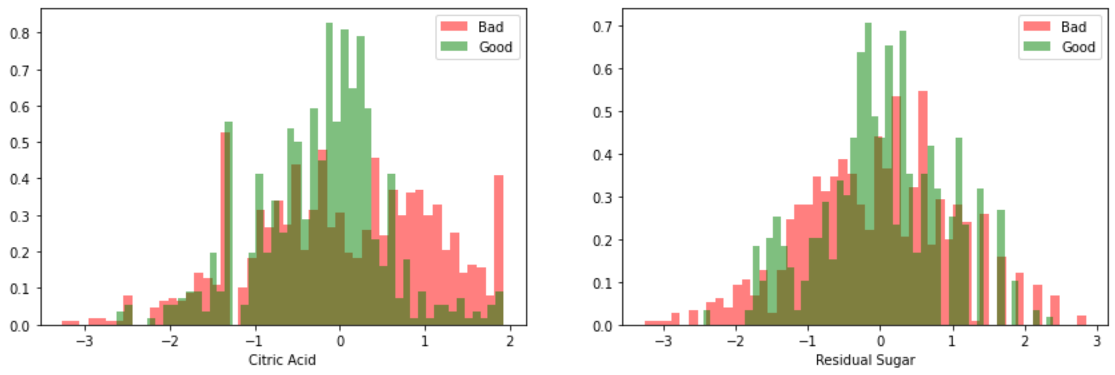


Figure 18: Citric Acid and Residual Sugar distributions

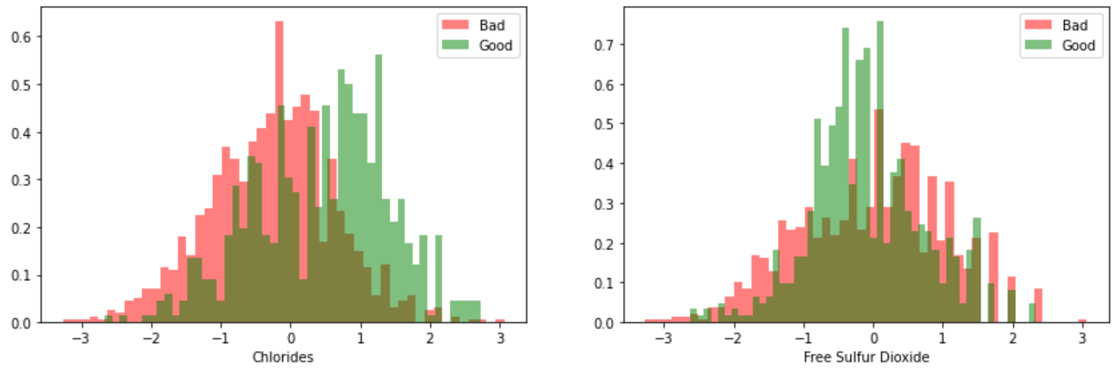


Figure 19: Chlorides and Free Sulfure Dioxide distributions

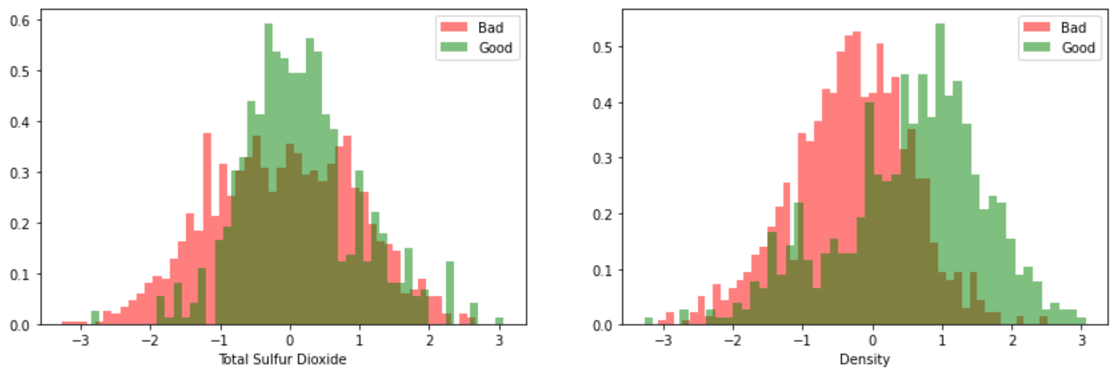


Figure 20: Total Sulfure Dioxide and Density distributions

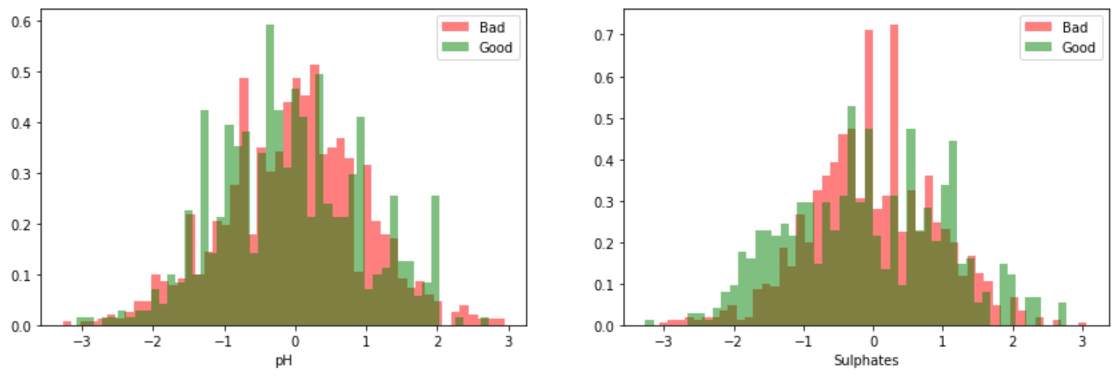


Figure 21: pH and Sulphates distributions

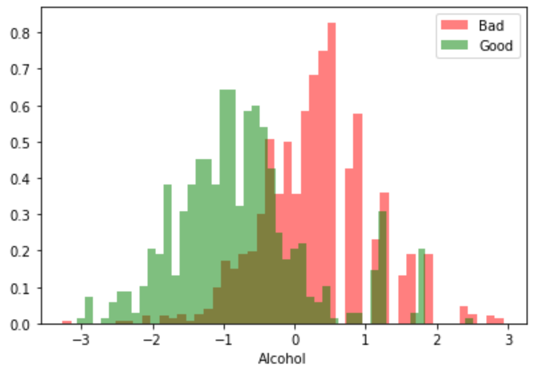


Figure 22: Alcohol distribution

The histrograms in most of the cases show a Gaussian Normal Distribution, but in some cases the features are still not normal distributed.

3.2 Correlations

I recomputed the correlation matrices and obtained the new heatmaps:

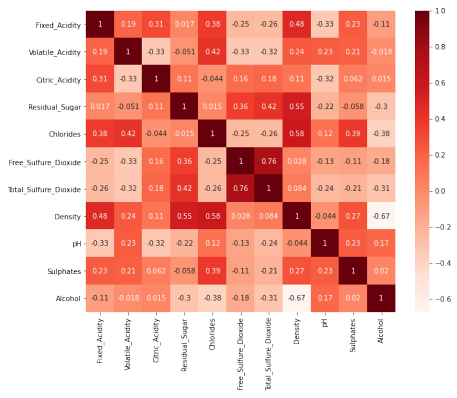


Figure 23: Covariance Heatmap of the whole gaussianized dataset

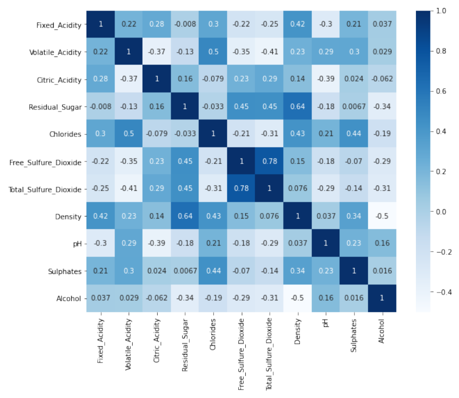


Figure 24: Covariance Heatmap of the Bad Quality samples

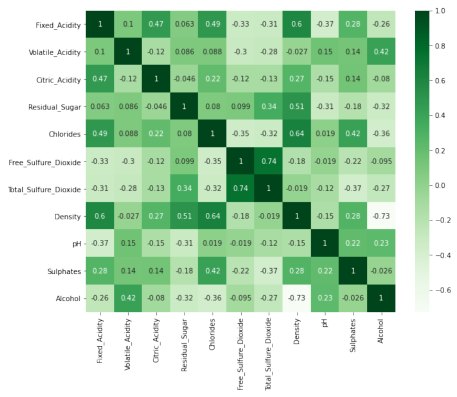


Figure 25: Covariance Heatmap of the Good Quality samples

There are two couples of features whose correlation is more than 0.65.

The labels 'Free Sulfure Dioxide' and 'Total Sulfure Dioxide' have a correlation of 0.76 for the whole dataset and the correlation stays strong for both good (0.74) and bad quality (0.78).

The labels 'Density' and 'Alcohol' have a negative correlation of -0.67 for the whole dataset, but it decreases for bad quality wines (-0.5), while it is stronger for good quality wines (-0.73).

4 Dimensionality Reduction Base

4.1 PCA

We would like to reduce the dimensionality of our dataset preserving most of the information. I exploit PCA that can be interpreted as a linear mapping that preserves the directions with highest variance; having computed the covariance matrix and so his eigen-decomposition, I projected the data in the subspace spanned by the m columns of U corresponding to the m highest eigenvalues where m=2 for reasons of plotting. Since for LDA the number of non zero eigenvalues is at most C-1, it allows estimating at most C-1 directions and this for a binary task is useless. I obtained the following plot:

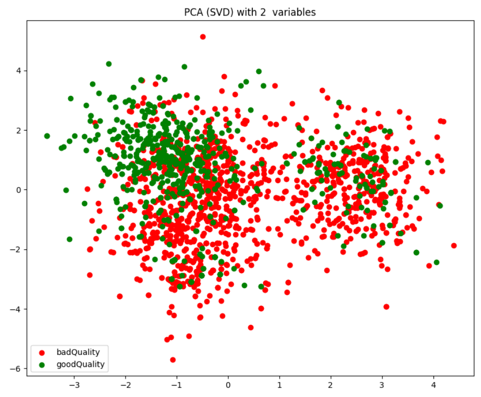


Figure 26: PCA applied on Gaussianized Dataset with two variables

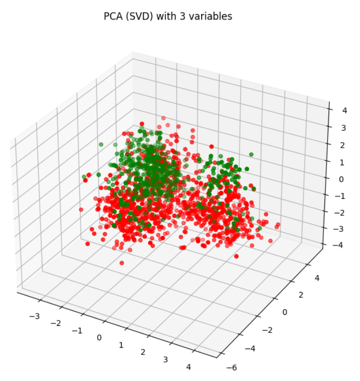


Figure 27: PCA applied on Gaussianized Dataset with three variables

5 MVG Classifier

Our main application will be a uniform prior one: $(\widetilde{\pi}, C_f p, C_f n) = (0.5, 1, 1)$ But I also considered unbalanced applications.

I measured performance in terms of normalized minimum detection costs, measuring the cost we would pay if we made optimal decisions for the validation set using the recognizer scores.

Single Fold			
Gaussianized features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.255	0.760	0.729
Diag-Cov	0.460	0.841	0.866
Tied Full-Cov	0.344	0.726	0.797
Tied Diag-Cov	0.441	0.834	0.928
Gaussianized features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.242	0.719	0.660
Diag-Cov	0.388	0.723	0.781
Tied Full-Cov	0.359	0.709	0.698
Tied Diag-Cov	0.362	0.718	0.714
Gaussianized features PCA(m=9)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.256	0.731	0.698
Diag-Cov	0.392	0.766	0.773
Tied Full-Cov	0.361	0.728	0.730
Tied Diag-Cov	0.362	0.725	0.728
Raw features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.264	0.702	0.802
Diag-Cov	0.409	0.822	0.941
Tied Full-Cov	0.327	0.754	0.706
Tied Diag-Cov	0.391	0.821	0.908

We can see, using a Single Fold protocol, that MVG(Full-Cov) gives better result using Gaussianized Features both and without PCA.

I also used a 5-fold cross validation to measure min DCF: every time a fold is used as a validation set and K-1 folds are used as training set.

5-Fold			
Gaussianized features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.302	0.799	0.768
Diag-Cov	0.339	0.848	0.873
Tied Full-Cov	0.346	0.790	0.860
Tied Diag-Cov	0.443	0.874	0.937
Gaussianized features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.292	0.794	0.719
Diag-Cov	0.398	0.818	0.863
Tied Full-Cov	0.351	0.792	0.809
Tied Diag-Cov	0.353	0.784	0.831
Gaussianized features PCA(m=9)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.304	0.813	0.721
Diag-Cov	0.403	0.818	0.810
Tied Full-Cov	0.352	0.811	0.792
Tied Diag-Cov	0.351	0.807	0.821
Raw features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.313	0.779	0.843
Diag-Cov	0.421	0.846	0.921
Tied Full-Cov	0.333	0.811	0.749
Tied Diag-Cov	0.403	0.866	0.931

The Full-Cov model obtains the best performance in K-fold cross validation protocol both with and without PCA. although single fold would seem to give better results, it is not possible to compare them with those obtained from 5-fold.

PCA is effective in improving our estimate and PCA with m=10 give the best performance.

The Tied models perform in general worse than Full-Cov.

Gaussianization significantly improves performance over raw features

The results between single-fold and K-fold are consistent, suggesting that the amount of data is enough for validation and model training.

As the data are very correlated for some features Naive Bayes assumption perform poorly

Overall, the best candidate is currently the MVG model with Full Covariance matrices (I choose the K-fold version, with PCA10), suggesting that we have enough data to reliably estimate the covariance matrices.

6 Discriminative approaches

Given the effectiveness of PCA on MVG Classifiers I also considered it on our analysis of Discriminative approaches.

6.1 Logistic Regression

we can re-balance the costs of the different classes, using the balanced version of objective function that we want to minimize

$$J(w,b) = \frac{\lambda}{2}||w||^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n} \log(1+e^{-z_i(w^T x_i+b)}) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n} \log(1+e^{-z_i(w^T x_i+b)}) \tag{3}$$

I used the K-fold version: the evaluation results are more reliable and the final model should prove more robust due to an increased number of training samples.

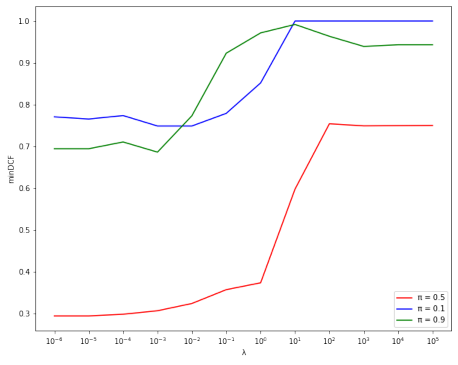


Figure 28: Single Fold Raw

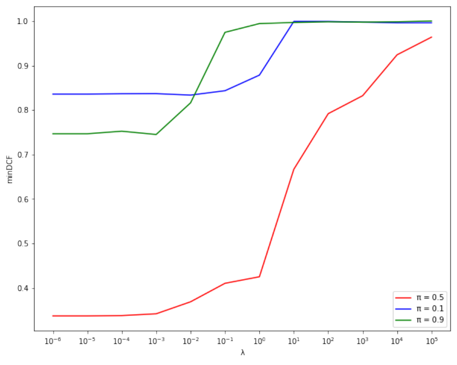


Figure 29: 5Fold Raw

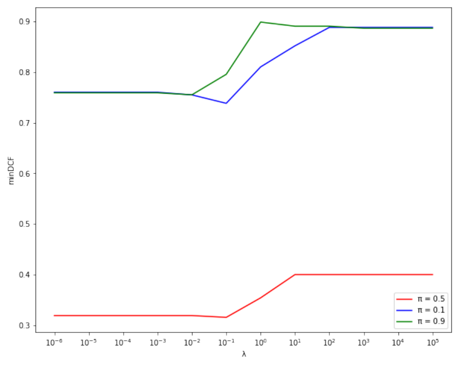


Figure 30: Single Fold Gaussianized

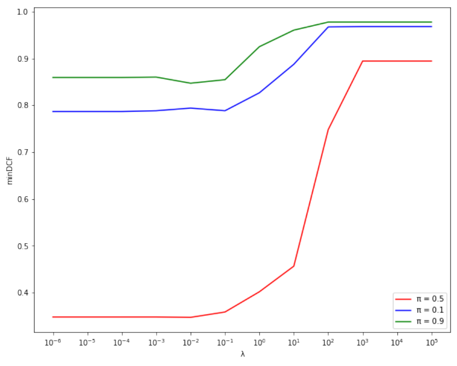


Figure 31: 5Fold Gaussianized

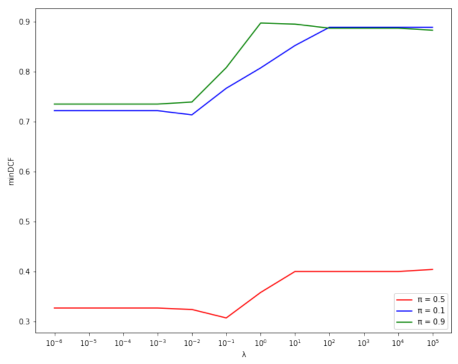


Figure 32: Single Fold PCA

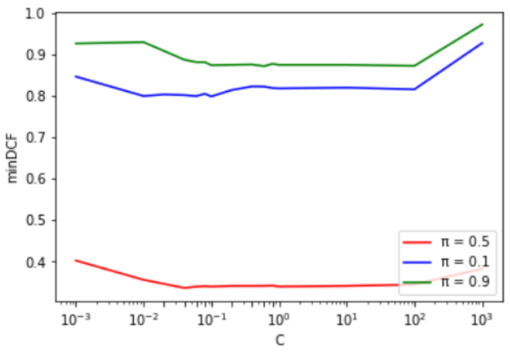


Figure 33: 5Fold PCA

Best results are obtained with small values of λ.
I chose $\lambda = 10^{-5}$.

5-Fold			
Gaussianized features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Full-Cov	0.292	0.794	0.719
Tied Full-Cov	0.351	0.792	0.809
Raw Features			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Log Reg($\lambda = 10^{-5}, \pi_T = 0.5$)	0.337	0.836	0.747
Log Reg($\lambda = 10^{-5}, \pi_T = 0.1$)	0.999	1.000	0.999
Log Reg($\lambda = 10^{-5}, \pi_T = 0.9$)	0.340	0.829	0.673
Gaussianized Features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Log Reg($\lambda = 10^{-5}, \pi_T = 0.5$)	0.348	0.787	0.860
Log Reg($\lambda = 10^{-5}, \pi_T = 0.1$)	1.000	1.000	1.000
Log Reg($\lambda = 10^{-5}, \pi_T = 0.9$)	0.351	0.800	0.817
Gaussianized Features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Log Reg($\lambda = 10^{-5}, \pi_T = 0.5$)	0.351	0.794	0.814
Log Reg($\lambda = 10^{-5}, \pi_T = 0.1$)	0.999	1.000	0.999
Log Reg($\lambda = 10^{-5}, \pi_T = 0.9$)	0.354	0.818	0.777

Overall, the MVG model with full covariances perform better.

Logistic regression provides very small improvement over the MVG model with linear classification rules (Tied Full-Cov)

Using different values for π_T does not improve too much the Logistic Regression models for the other two applications. On the contrary, it can be seen that using a $\pi_T = 0.1$ there is a noticeable deterioration

Gaussianization seems much less relevant with respect to the MVG Models.

Indeed, logistic regression does not require assumptions on the data distribution.

6.2 Logistic Regression Quadratic

I performed the Quadratic Logistic Regression, applying Logistic regression in the expanded dataset:

$$\phi(x) = \begin{bmatrix} vec(xx^T) \\ x \end{bmatrix} \tag{4}$$

This correspond to a Quadratic form in the original feature space, we are actually estimating quadratic separation surfaces in the original space.

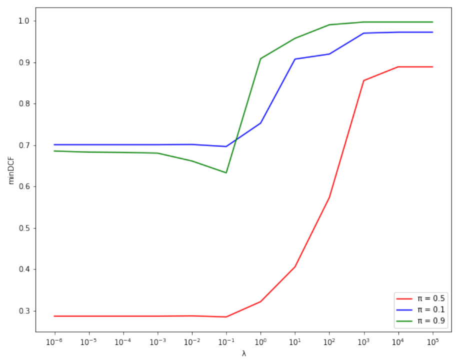


Figure 34: 5Fold Gau piT=0.5

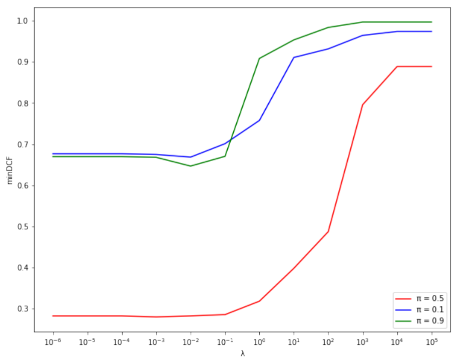


Figure 35: 5Fold Gau PCA piT=0.9

5-Fold			
Gaussianized Features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.302	0.799	0.768
Quad Log Reg($\lambda = 0.1, \pi_T = 0.5$)	0.285	0.696	0.633
Quad Log Reg($\lambda = 0.1, \pi_T = 0.1$)	1.000	1.000	1.000
Quad Log Reg($\lambda = 0.1, \pi_T = 0.9$)	0.288	0.686	0.647
Gaussianized Features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.292	0.794	0.719
Quad Log Reg($\lambda = 0.01, \pi_T = 0.5$)	0.282	0.696	0.681
Quad Log Reg($\lambda = 0.01, \pi_T = 0.1$)	1.000	1.000	1.000
Quad Log Reg($\lambda = 0.001, \pi_T = 0.9$)	0.280	0.675	0.668

The RAW Features results are not reported here because with Quadratic Logistic Regression gives very bad results (around 0.8-0.9). This confirm that pre-processing (Gaussinization) is significantly more helpful in this case.

We can see that for Gaussianized Features with PCA with m=10 there is an improvement using $\pi_T = 0.9$ however, the results for $\pi_T = 0.1$ are still poor.

In general the Quadratic Logistic Regression performs better than MVG(Full-Cov) both with PCA and no-PCA. This suggest that the Gaussian assumption may not be sufficiently accurate for our features. The best candidate is so the Quadratic Logistic Regression with $\lambda = 0.001$ and $\pi_T = 0.9$

6.3 SVM LINEAR

For linear SVM, we need to tune the hyper-parameter C.

I consider in the following the SVM model that does not balance the two classes because the balanced version and the unbalanced version gives me similar results (quite identical)

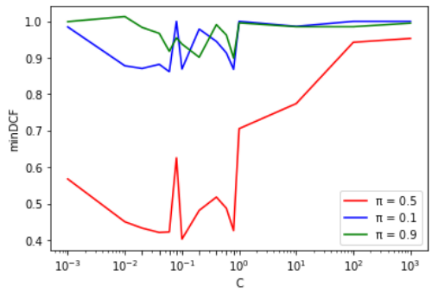


Figure 36: 5Fold Raw

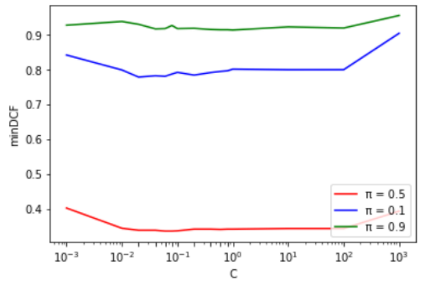


Figure 37: 5Fold Gau

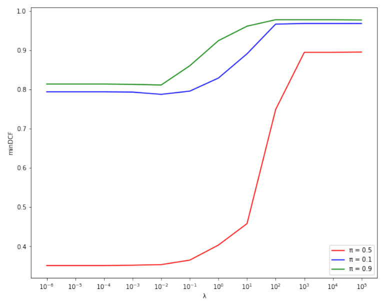


Figure 38: 5Fold Gau PCA

We can compare linear models in terms of min DCF:

5-Fold			
Raw Features			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Tied Full-Cov	0.333	0.811	0.749
Log Reg($\lambda = 10^-5, \pi_T = 0.5$)	0.337	0.836	0.747
Linear SVM(C=0.8)	0.403	0.870	0.938
Gaussianized Features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Tied Full-Cov	0.346	0.790	0.860
Log Reg($\lambda = 10^-5, \pi_T = 0.5$)	0.348	0.787	0.860
Linear SVM(C=0.06)	0.336	0.780	0.917
Gaussianized Features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
Tied Full-Cov	0.351	0.792	0.809
Log Reg($\lambda = 10^-5, \pi_T = 0.5$)	0.351	0.794	0.814
Linear SVM(C=0.04)	0.336	0.802	0.888

Linear SVM performs better with respect to other linear approaches with dataset with Gaussianized features both for no PCA and PCA m=10.

Instead considering RAW Features, the Tied Full-Cov performs better than the Linear SVM. This suggest that the covariances of the classes should be very similar.

6.4 SVM Quadratic

Since non-linear models perform better on this dataset, I considered two non-linear SVM formulations. To perform the Quadratic SVM I used a kernel function that allows training a SVM in a large dimensional Hilbert space, without requiring to explicity compute the mapping. In this case we are computing a linear separation surface in the expanded space, which correspond to a non-linear separation surface in the original feature space.

6.4.1 SVM Quadratic Polynomial

In this case i used the Kernel function:

$$k(x_1, x_2) = (x_1^T x_2 + c)^2 \tag{5}$$

In this case i tuned the Hyperparameters c and C.

We can compare quadratic models in terms of min DCF:

5-Fold			
Gaussianized Features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.302	0.799	0.768
Quad Log Reg($\lambda = 0.1, \pi_T = 0.5$)	0.285	0.696	0.633
Quadratic SVM(C=0.01,c=100.0,d=2)	0.286	0.721	0.644
Gaussianized Features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.292	0.794	0.719
Quad Log Reg($\lambda = 0.001, \pi_T = 0.9$)	0.280	0.675	0.668
Quadratic SVM(C=0.01,c=10.0,d=2)	0.284	0.718	0.638

Quadratic kernel polynomial SVM provides slightly worse results than the Quadratic Logistic regression.

6.4.2 SVM Quadratic RBF Kernel

In this case i used the Kernel function:

$$k(x_1, x_2) = e^{-\gamma ||x_1 - x_2||^2} \tag{6}$$

I used a grid search to jointly optimize C and

We calculated the minDCF for different values of C and γ

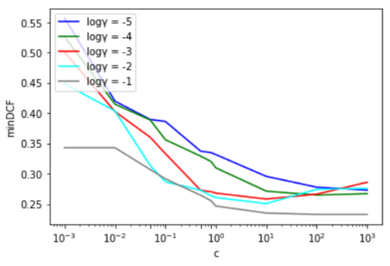


Figure 39: 5Fold Gau

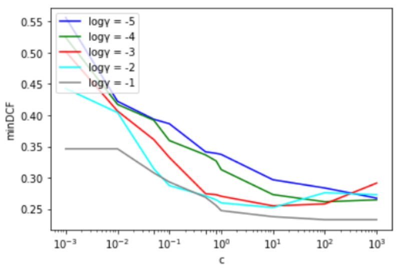


Figure 40: 5Fold Gau PCA

Best and more robust results are obtained using $\log \gamma = 1$ and C $= 100$

5-Fold			
Gaussianized Features no PCA			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.302	0.799	0.768
Quad Log Reg($\lambda = 0.1, \pi_T = 0.5$)	0.285	0.696	0.633
Quadratic SVM(C=0.01,c=100.0,d=2)	0.286	0.721	0.644
Kernel RBF SVM(C=100.0,log$\gamma = -1$)	0.233	0.496	0.656
Gaussianized Features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.292	0.794	0.719
Quad Log Reg($\lambda = 0.001, \pi_T = 0.9$)	0.280	0.675	0.668
Quadratic SVM(C=0.01,c=10.0,d=2)	0.284	0.718	0.638
Kernel RBF SVM(C=100,log$\gamma = -1$)	0.233	0.478	0.659

As for Quadratic Logistic regression the RAW Features results are not reported here because gives very bad results (around 0.8-0.9). This confirm, using a quadratic model, that pre-processing (Gaussinization) is significantly more helpful.

We can see that the Quadratic SVM based on RBF Kernel gives better results than the polynomial one and also the Quadratic Logistic Regression.

Till now, the Kernel RBF SVM is the best candidate both for Gaussianzed Features with PCA and with no PCA.

7 Gaussian Mixture Models

The last model I considered is a generative approach based on training a GMM over the data of each class. GMMs can approximate generic distributions, so we expect to obtain better results than with the Gaussian model.

5-Fold			
Raw Features			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.313	0.779	0.843
GMM(Full-Cov), 4 Gau	0.289	0.727	0.708
GMM(Full-Cov), 16 Gau	0.302	0.907	0.679
GMM(Tied Full-Cov), 4 Gau	0.318	0.816	0.838
GMM(Tied Full-Cov), 16 Gau	0.353	0.794	0.802
Gaussianized Features			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.302	0.799	0.768
Quad Log Reg($\lambda = 0.1, \pi_T = 0.5$)	0.285	0.696	0.633
Quadratic SVM(C=0.01,c=100.0,d=2)	0.286	0.721	0.644
Kernel RBF SVM(C=100.0,log$\gamma = -1$)	0.233	0.496	0.656
GMM(Full-Cov), 4 Gau	0.297	0.713	0.814
GMM(Full-Cov), 16 Gau	0.320	0.642	0.914
GMM(Tied Full-Cov), 4 Gau	0.302	0.799	0.768
GMM(Tied Full-Cov), 16 Gau	0.307	0.748	0.851
Gaussianized Features PCA(m=10)			
Classifier type	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.9$
MVG(Full-Cov)	0.292	0.794	0.719
Quad Log Reg($\lambda = 0.001, \pi_T = 0.9$)	0.280	0.675	0.668
Quadratic SVM(C=0.01,c=10.0,d=2)	0.284	0.718	0.638
Kernel RBF SVM(C=100,log$\gamma = -1$)	0.233	0.478	0.659
GMM(Full-Cov), 2 Gau	0.279	0.702	0.654
GMM(Full-Cov), 4 Gau	0.301	0.691	0.749
GMM(Tied Full-Cov), 4 Gau	0.292	0.794	0.719
GMM(Tied Full-Cov), 16 Gau	0.310	0.731	0.823

As expected GMM obtain better results than with the Gaussian model.

From the table above it can be seen that the most performing models are: GMM(Full-Cov) with 2 componens, SVM Kernel RBF and Quadratic Logistic Regression.

This in general is true for Gaussianized features with and without PCA, instead for RAW Features, the best model is GMM(Full-Cov) with 4 components.

It can be seen that RBF KERNEL performs better on the Gaussianized dataset, this is confirmed by the DET graph where the three best models mentioned above were compared.

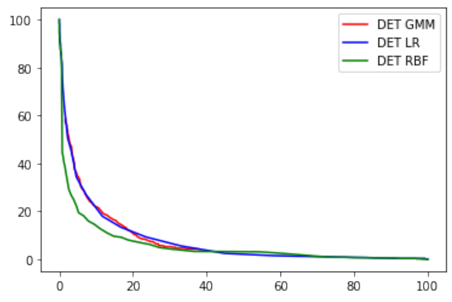


Figure 41: DET Gauss

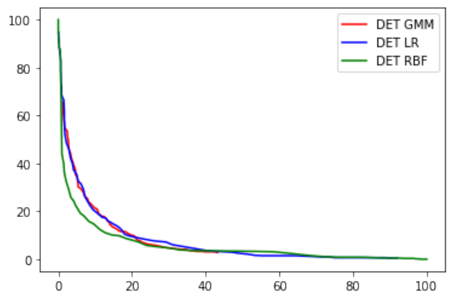


Figure 42: DET Gauss PCA 10

8 Actual DCF

Up to now I have considered only minimum DCF metrics. The cost that we actually pay, however, depends on the goodness of the decisions we make using those scores. So I calculated the actual DCF for the best three models mentioned above.

I evaluated the actual DCF using the theoretical threshold.

$$t = -\log \frac{\widetilde{\pi}}{1-\widetilde{\pi}} \tag{7}$$

Kernel RBF SVM(C=100,log$\gamma = -1$)			
	min DCF($\widetilde{\pi} = 0.5$)	min DCF($\widetilde{\pi} = 0.1$)	min DCF($\widetilde{\pi} = 0.9$)
	0.233	0.478	0.659
	act DCF($\widetilde{\pi} = 0.5$)	act DCF($\widetilde{\pi} = 0.1$)	act DCF($\widetilde{\pi} = 0.9$)
	0.247	0.971	0.993

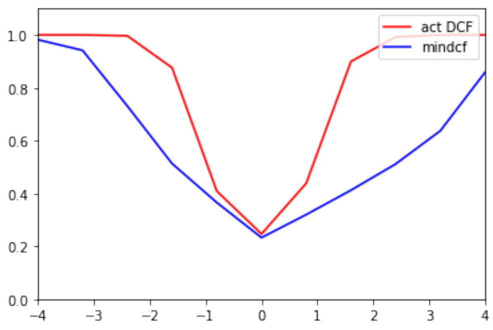


Figure 43: Bayes Error SVM

We can see that the calibration of SVM Kernel RBF is quite poor: this is due to the fact that the model lacks a probabilistic interpretation.

I tried to re-calibrate the scores using the prior-weighted Logistic Regression computing the following function to recover the calibrated scores:

$$f(s) = \alpha s + \beta' - \log \frac{\widetilde{\pi}}{1 - \widetilde{\pi}} \tag{8}$$

The regularization in this case is useless and has caused a worsening of the results, since there is a norm of w that is too low which again produces uncalibrated scores.

Instead we can observe that the GMM and Quadratic Logistic Regression provides scores that are almost calibrated as we can see from the Bayes Error graph.

GMM(Full-cov), 2 Gau			
	min DCF($\widetilde{\pi} = 0.5$)	min DCF($\widetilde{\pi} = 0.1$)	min DCF($\widetilde{\pi} = 0.9$)
	0.279	0.702	0.654
	act DCF($\widetilde{\pi} = 0.5$)	act DCF($\widetilde{\pi} = 0.1$)	act DCF($\widetilde{\pi} = 0.9$)
	0.281	0.776	0.731

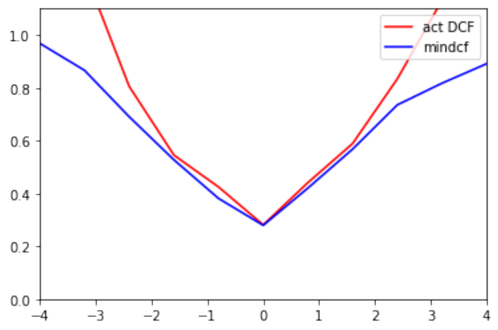


Figure 44: Bayes Error GMM

Logistic Regression Quadratic(L=0.001,piT=0.9)			
	min DCF($\widetilde{\pi} = 0.5$)	min DCF($\widetilde{\pi} = 0.1$)	min DCF($\widetilde{\pi} = 0.9$)
	0.280	0.675	0.668
	act DCF($\widetilde{\pi} = 0.5$)	act DCF($\widetilde{\pi} = 0.1$)	act DCF($\widetilde{\pi} = 0.9$)
	0.324	0.823	0.691

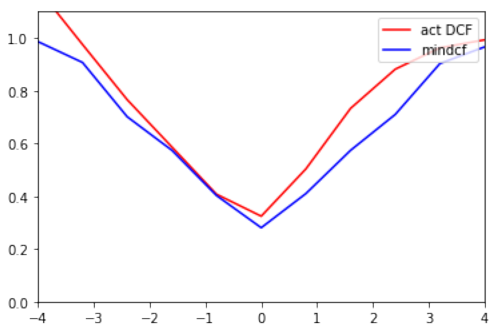


Figure 45: Bayes Error LR

9 Model Evaluation

Since according to the analysis we obtained better results on GMM 2 Gau, Kernel RBF SVM and Quadratic Logistic Regression, I computed the evaluation in these three models using the optimal Hyperparameters found in the Learn stage.

Below are the results of Accuracy and Error:

All Data		
Gaussianized Features PCA(m = 10)		
Classifier type	Accuracy	Error
GMM(Full-cov), 2 Gau	78.92%	21.08%
Kernel RBF SVM	76.13%	23.87%
Logistic Regression Quad	77.44%	22.56%
Gaussianized Features		
Classifier type	Accuracy	Error
GMM(Full-cov), 2 Gau	83.92%	16.08%
Kernel RBF SVM	83.86%	16.14%
Logistic Regression Quad	85.13%	14.87%
RAW Features		
Classifier type	Accuracy	Error
GMM(Full-cov), 2 Gau	83.97%	16.03%
Kernel RBF SVM	73.44%	26.56%
Logistic Regression Quad	63.56%	36.44%

Since we observed that GMM and Quadratic Logistic Regression provides well-calibrated scores with respect to SVM Kernel RBF, the results obtained are consistent with our expectations.

The Gaussianized Features without PCA, however, perform slightly better than we expected with respect to Gaussinized Features with PCA.

We can see that, as we expected, the Quadratic Logistic Regression over the RAW Features is harmulf since a Dummy Model that always predicts "Bad

Quality" would achieve an accuracy of $\approx 66\%$

10 Some Comments

During the Learn stage I performed wrongly the Gaussianization and PCA: I performed them on the initial train set, when instead it would be more correct to perform the Gaussianization and PCA on the train set and validation set at each iteration of the kfold.