Report of the project for the exam of the *Machine learning and pattern recognition* course.

Candidate 1: Diego Gasco s296762

Candidate 2: Giovanni Genna s304684

Abstract

We have chosen to work on the *Wine quality detection* task. The request of the problem is to discriminate between good and bad quality wines. The first goal of our work is to study and analyse the provided problem, in particular the kind of features, their ranges and their distribution. The second part consists in developing the most appropriate classification algorithms and discarding models that are not proper for the considered task, by means of the training data. Finally, the different approaches chosen are evaluated on the test set.

The dataset

The dataset is taken from the UCI repository. The original dataset consists of 10 classes (quality 1 to 10). For this project, the dataset has been binarized, collecting all wines with low quality (score lower than 6) into class 0, and good quality (score greater than 6) into class 1. Wines with quality 6 have been discarded to simplify the task. The dataset contains both red and white wines (originally separated, they have been merged). There are 11 features, that represent physical properties of the wine:

1. Fixed acidity
2. Volatile acidity
3. Citric acid
4. Residual sugar
5. Chlorides
6. Free sulfur dioxide
7. Total sulfur dioxide

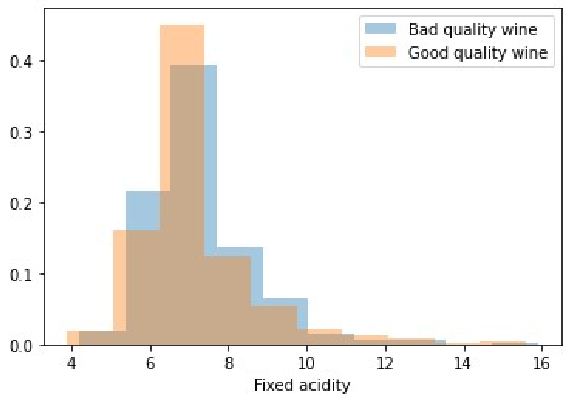
8. Density
9. pH
10. Sulphates
11. Alcohol

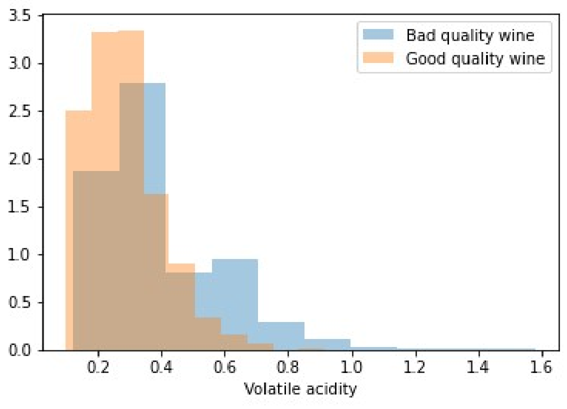
All the features can be considered continuous.

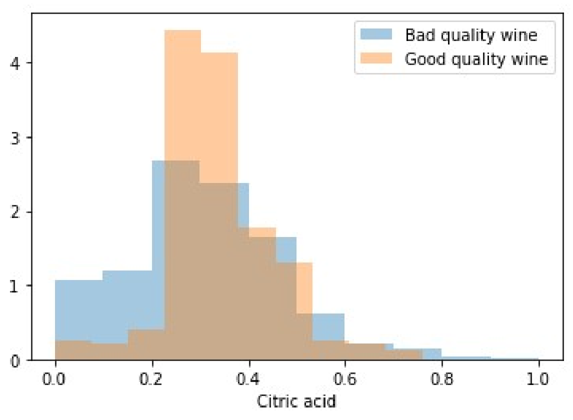
In the training set, there are 1839 samples, with their own features and another field with the class to which they belong (0 or 1). In particular, in this set, 1226 samples belong to the bad quality wine class and 613 to the good quality wine class. We modified the initial 'Train.txt' file to extract a matrix with on each column a sample (the numerical values of the eleven features: 11 rows, 1839 columns) and an array whose each element is the class label of the considered sample (we did these operations also for the 'Text.txt' file). In the test set, there are 1822 samples, 1158 belonging to the bad quality wine class and 664 belonging to the good quality wine class.

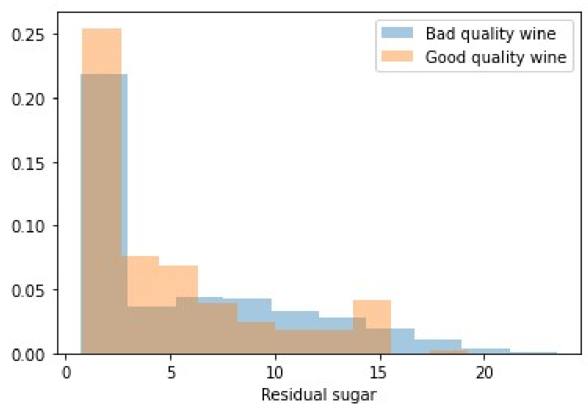
From this point, we will consider only the data of the training set. Later, when we will start working also on the test set (evaluation set), we will specify it.

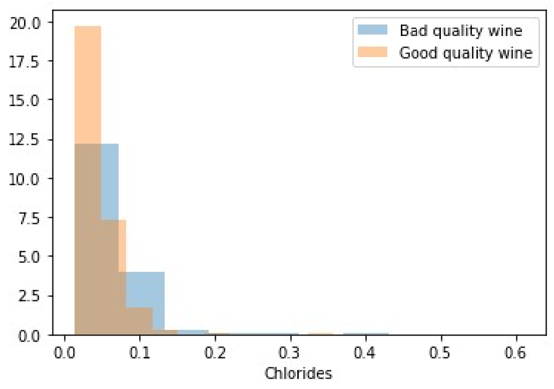
To understand how the several features of the two classes are distributed, for each of them, it is possible to plot the corresponding histogram:

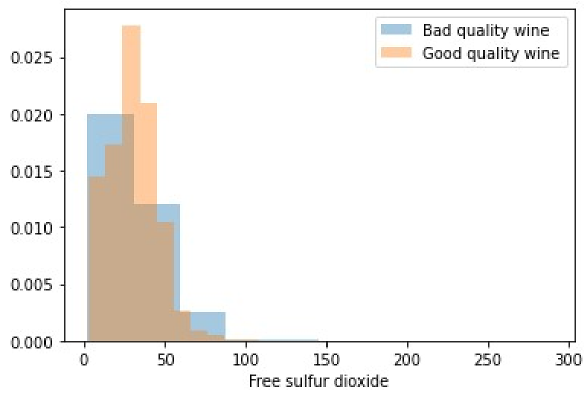


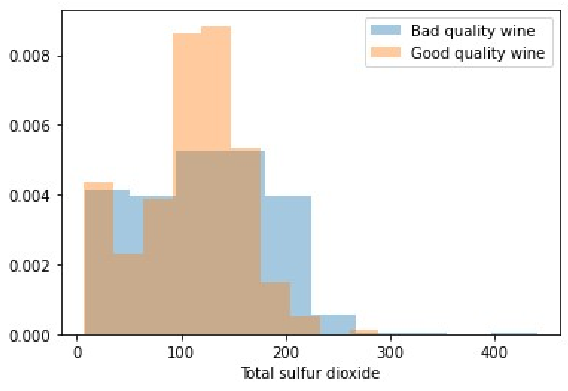


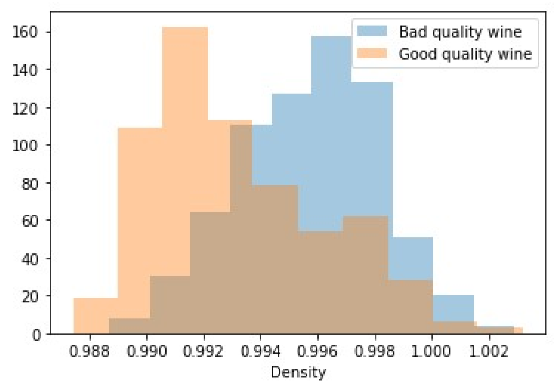

Bad quality wine
Good quality wine
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
Citric acid

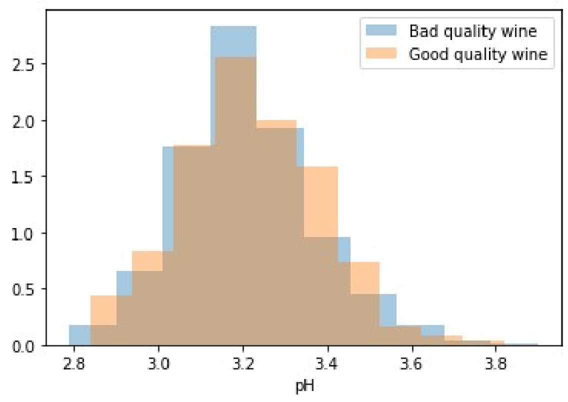

Bad quality wine
Good quality wine
0.00
0.05
0.10
0.15
0.20
0.25
0
5
10
15
20
Residual sugar

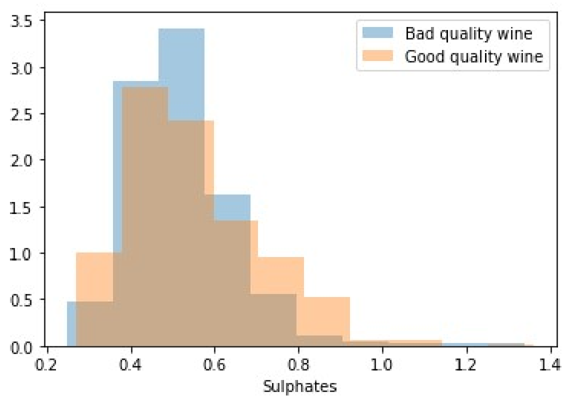

Bad quality wine
Good quality wine
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Chlorides

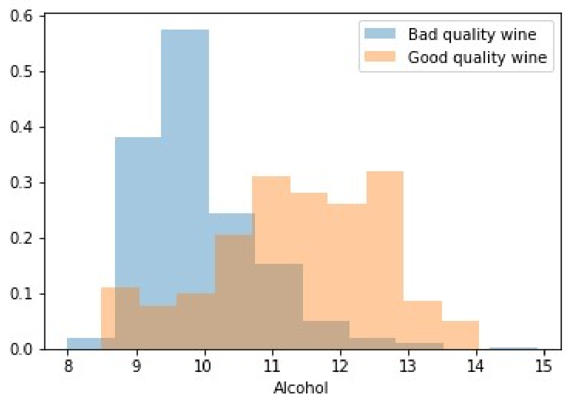

Bad quality wine
Good quality wine
0.000
0.005
0.010
0.015
0.020
0.025
0
50
100
150
200
250
300
Free sulfur dioxide


Bad quality wine
Good quality wine
0.000
0.002
0.004
0.006
0.008
0
100
200
300
400
Total sulfur dioxide


Bad quality wine
Good quality wine
0
20
40
60
80
100
120
140
160
0.988
0.990
0.992
0.994
0.996
0.998
1.000
1.002
Density


Bad quality wine
Good quality wine
0.0
0.5
1.0
1.5
2.0
2.5
2.8
3.0
3.2
3.4
3.6
3.8
pH


Bad quality wine
Good quality wine
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Sulphates



We can observe that, in general, for all features, there is no very well separation, vice versa, these is a large overlap and, sometimes, as in the case of *fixed acidity, residual sugar, pH* and *sulphates* the data is almost totally overlapping. For *density* and *alcohol*, it is possible to notice quite clearly that they are the features more separate. We can note that the data of the two classes for most of the features are distributed, more or less, like a Gaussian distribution, particularly for *fixed acidity, citric acid, density, pH, sulphates* and *alcohol*. In some classes, there is also the presence of outliers, especially for *chlorides* and *free sulfur dioxide.*

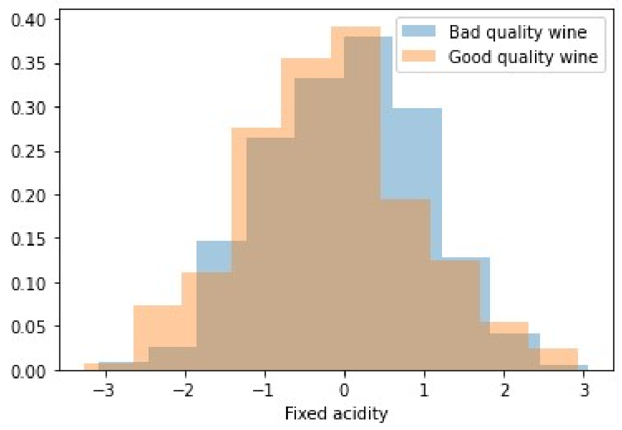
Even if most of the features have distribution like a Gaussian, given the presence of some features distributions less similar to a Gaussian and the presence of outliers, we try "Gaussianizing" the features. "Gaussianization" is a procedure that maps set of features to values whose empirical cumulative distribution function is well approximated by a Gaussian cumulative distribution function. The processing consists in mapping the features to a uniform distribution and then transforming the mapped features through the inverse of Gaussian cumulative distribution function. Let *x* be the feature to transform: the first step is to compute its rank *r(x)* over the training dataset:

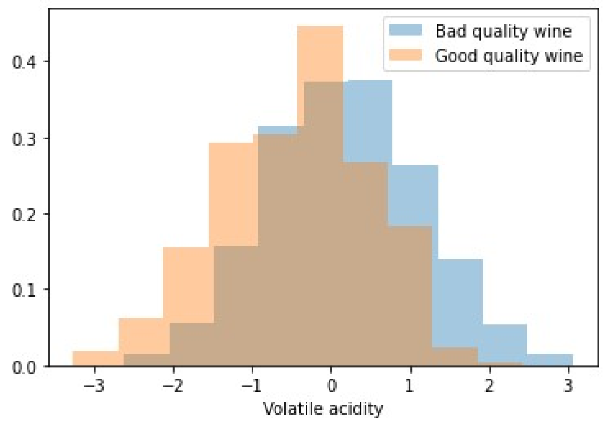
$$r(x) = \frac{\sum_{i=1}^{N} \mathbb{I}[x_i < x] \ + 1}{N + 2}$$

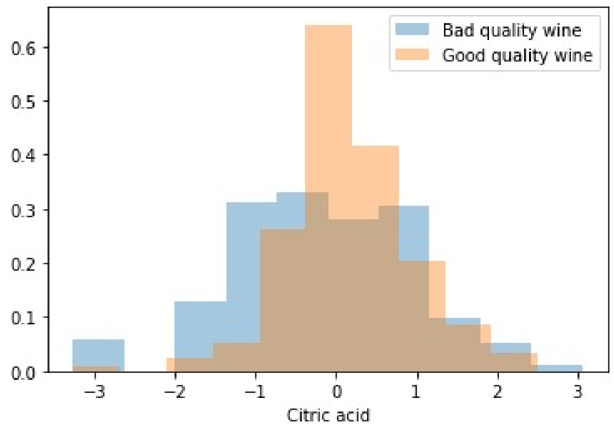
where x_i is the value of the considered feature for the ith training sample. 1 to the numerator and 2 to the denominator are added to avoid numerical issues in the next stage (i.e., to assume the existence of a feature smaller than all the others and a feature larger than all the others). Finally, we compute the transformed feature as $y = \Phi^{-1}\ (r(x))$, where Φ is the inverse of the cumulative distribution function of the standard normal distribution.

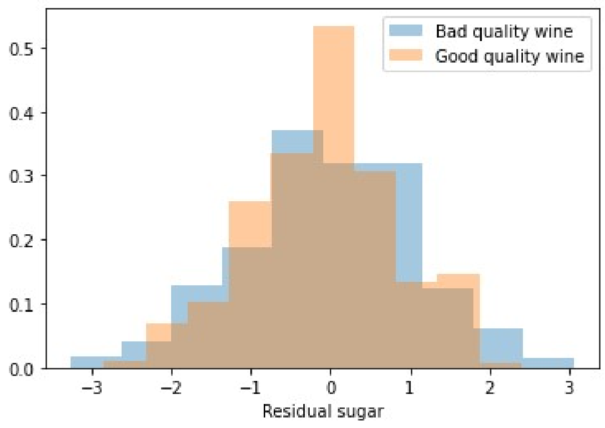
When we will work on the test set (evaluation set), we will apply the transformation both to training and evaluation samples, but the ranking should always be computed using the training data.

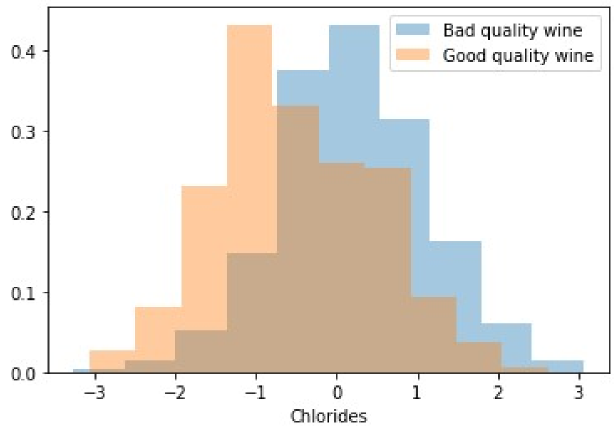
After the "Gaussianization", the several features of the two classes are distributed as represented in the following histograms:

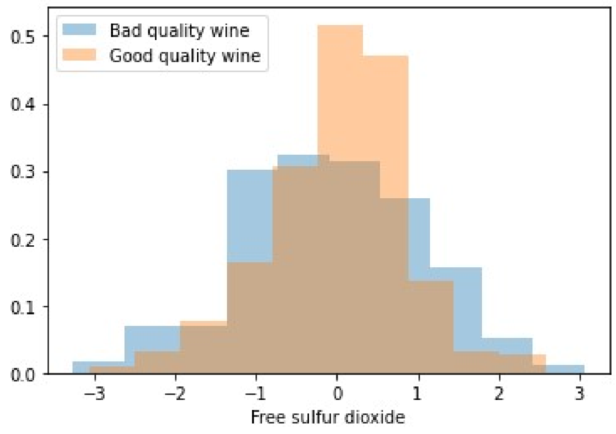


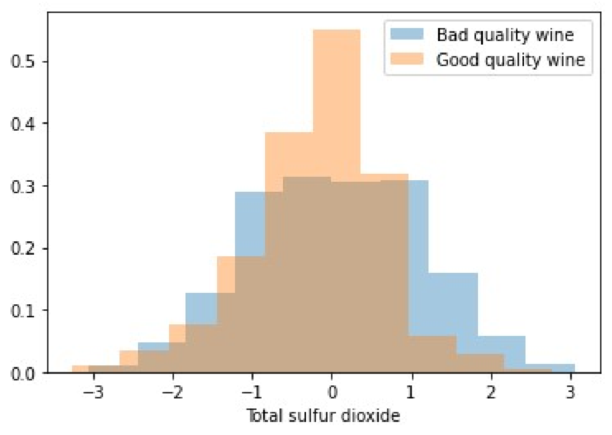


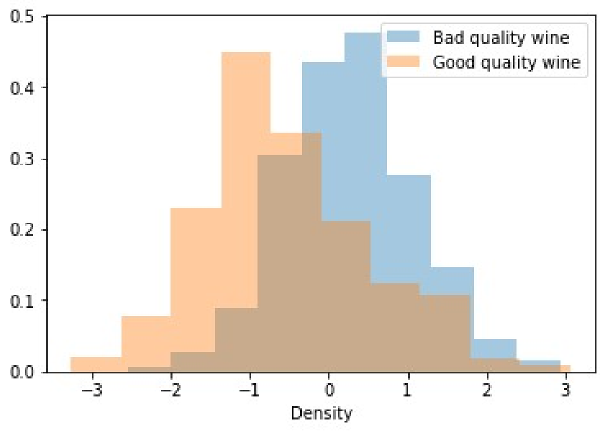


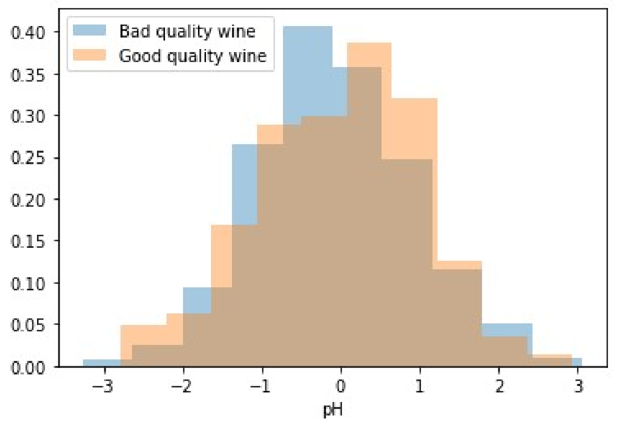


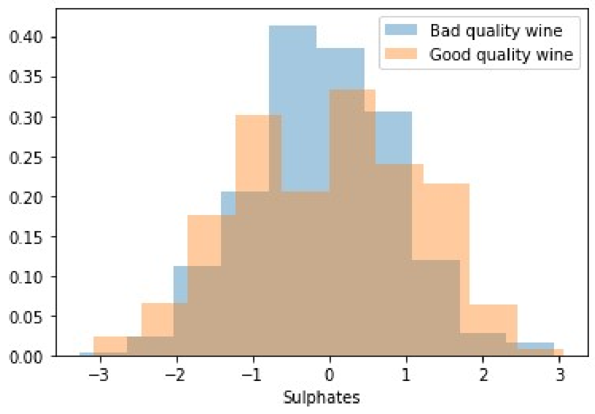


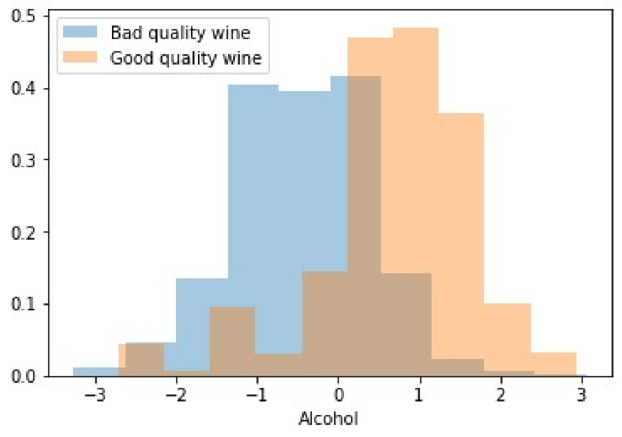






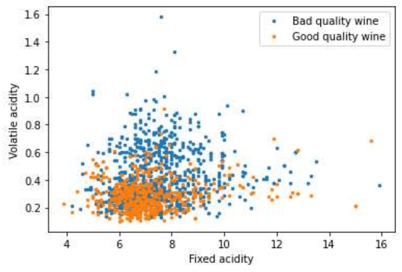
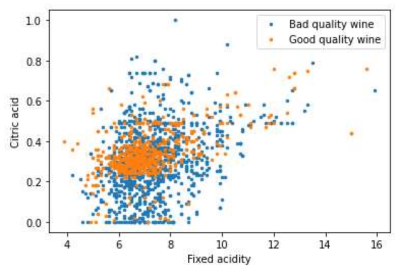
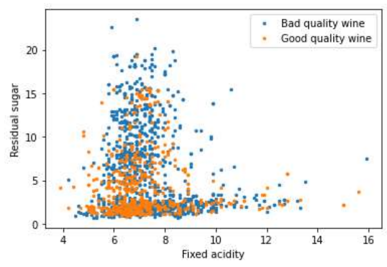
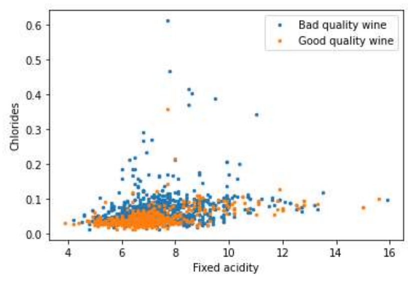
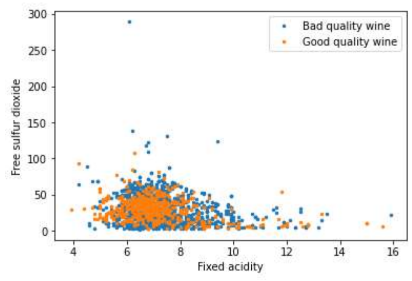
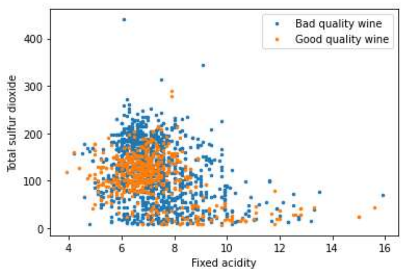
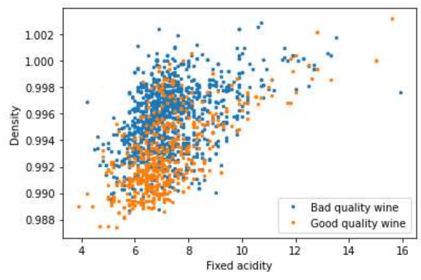
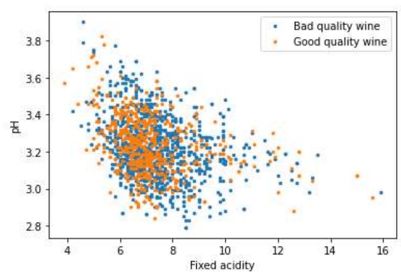
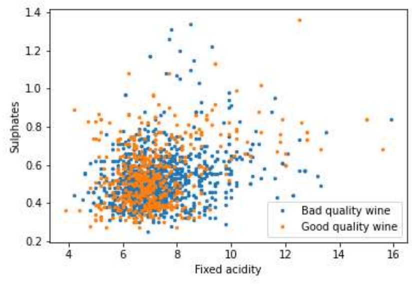
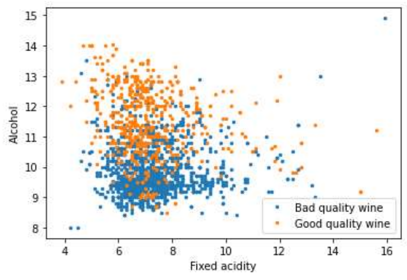
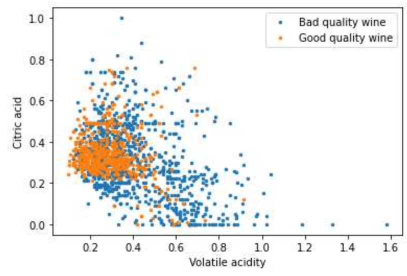
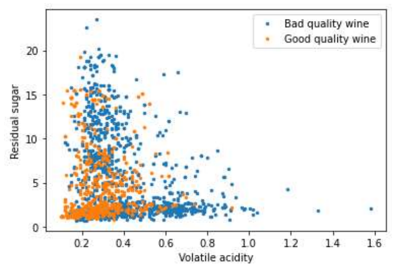
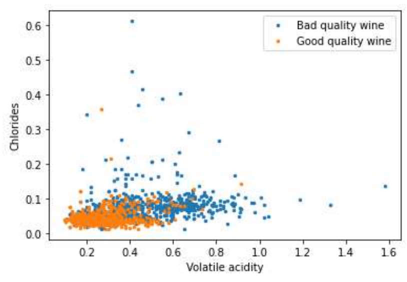
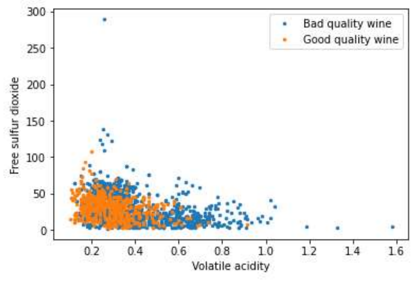
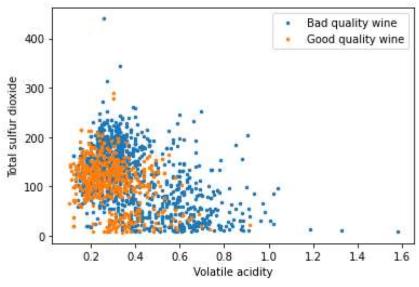
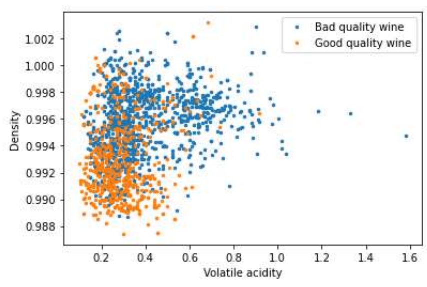
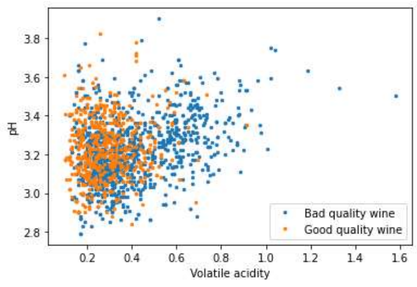
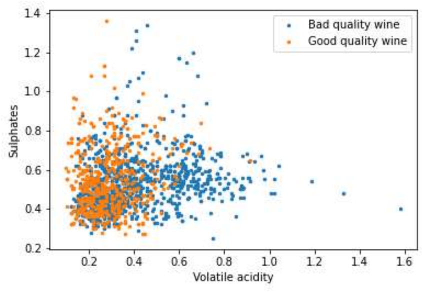


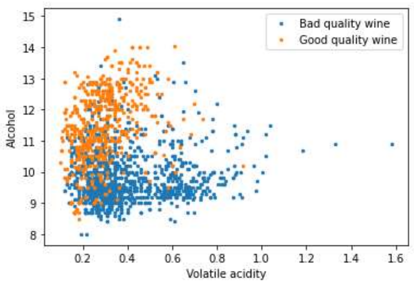


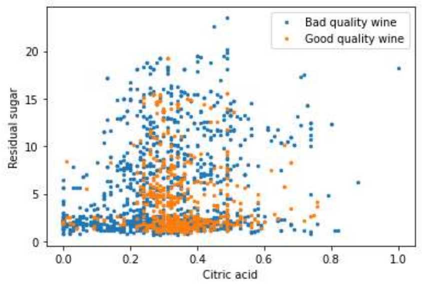


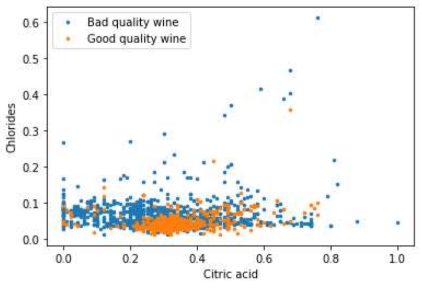
“Gaussianization” could be useful mainly for Gaussian classifier.

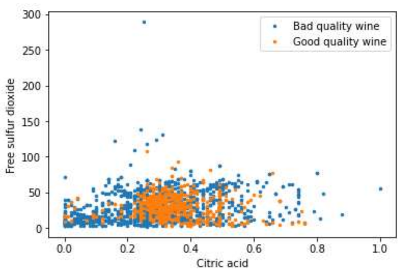
To have a view of the relationship between the different features for the different classes, we can visualize the scatter plots of the different features pairs for each class:

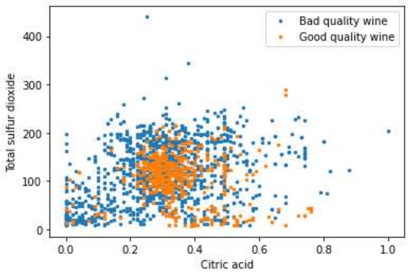

Volatile acidity
Fixed acidity
Bad quality wine
Good quality wine

Citric acid
Fixed acidity
Bad quality wine
Good quality wine

Residual sugar
Fixed acidity
Bad quality wine
Good quality wine

Chlorides
Fixed acidity
Bad quality wine
Good quality wine

Free sulfur dioxide
Fixed acidity
Bad quality wine
Good quality wine

Total sulfur dioxide
Fixed acidity
Bad quality wine
Good quality wine

Density
Fixed acidity
Bad quality wine
Good quality wine

pH
Fixed acidity
Bad quality wine
Good quality wine

Sulphates
Fixed acidity
Bad quality wine
Good quality wine

Alcohol
Fixed acidity
Bad quality wine
Good quality wine

Citric acid
Volatile acidity
Bad quality wine
Good quality wine

Residual sugar
Volatile acidity
Bad quality wine
Good quality wine

Chlorides
Volatile acidity
Bad quality wine
Good quality wine

Free sulfur dioxide
Volatile acidity
Bad quality wine
Good quality wine

Total sulfur dioxide
Volatile acidity
Bad quality wine
Good quality wine

Density
Volatile acidity
Bad quality wine
Good quality wine

pH
Volatile acidity
Bad quality wine
Good quality wine

Sulphates
Volatile acidity
Bad quality wine
Good quality wine

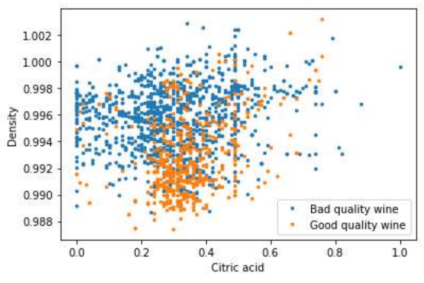

Bad quality wine
Good quality wine
Alcohol
Volatile acidity

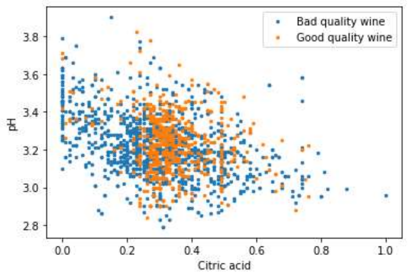

Bad quality wine
Good quality wine
Residual sugar
Citric acid

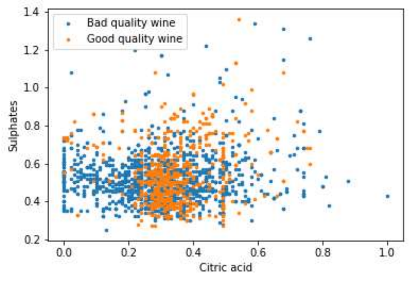

Bad quality wine
Good quality wine
Chlorides
Citric acid

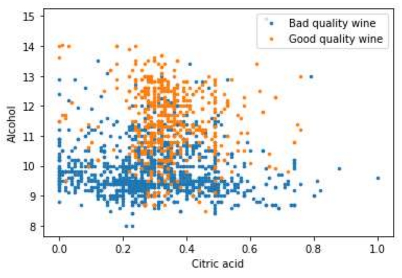

Bad quality wine
Good quality wine
Free sulfur dioxide
Citric acid

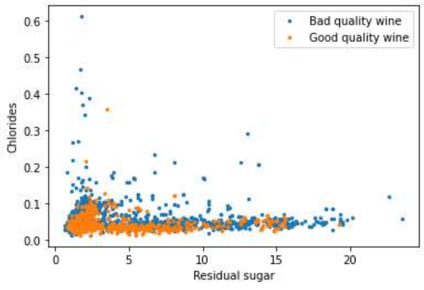

Bad quality wine
Good quality wine
Total sulfur dioxide
Citric acid

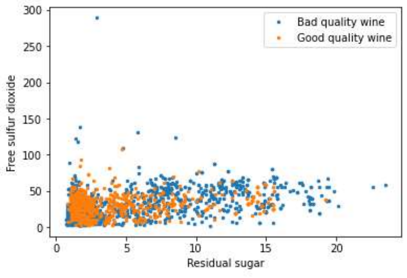

Bad quality wine
Good quality wine
Density
Citric acid

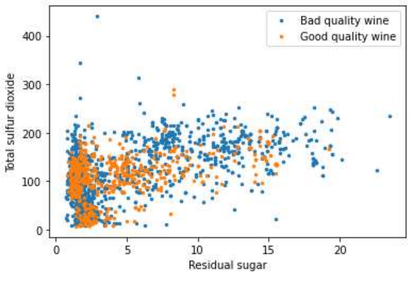

Bad quality wine
Good quality wine
pH
Citric acid

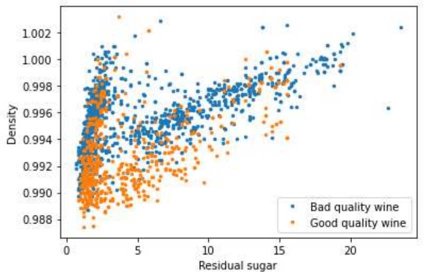

Bad quality wine
Good quality wine
Sulphates
Citric acid

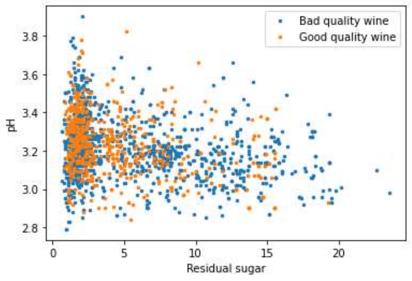

Bad quality wine
Good quality wine
Alcohol
Citric acid

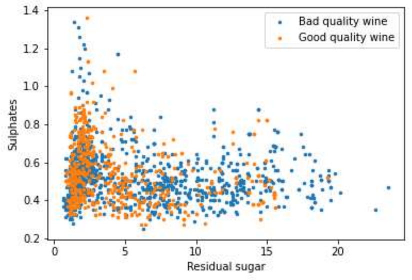

Bad quality wine
Good quality wine
Chlorides
Residual sugar

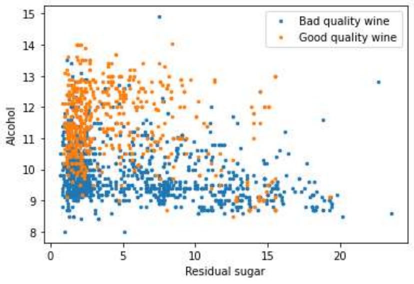

Bad quality wine
Good quality wine
Free sulfur dioxide
Residual sugar

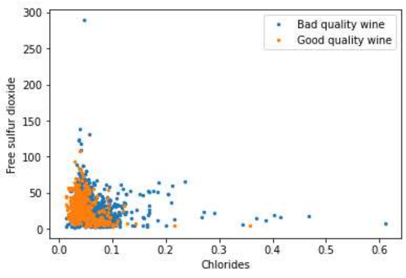

Bad quality wine
Good quality wine
Total sulfur dioxide
Residual sugar

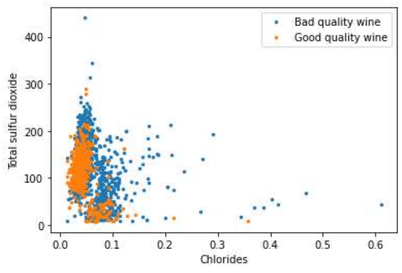

Bad quality wine
Good quality wine
Density
Residual sugar

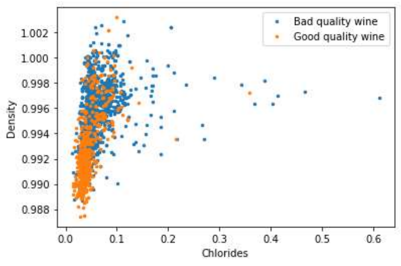

Bad quality wine
Good quality wine
pH
Residual sugar

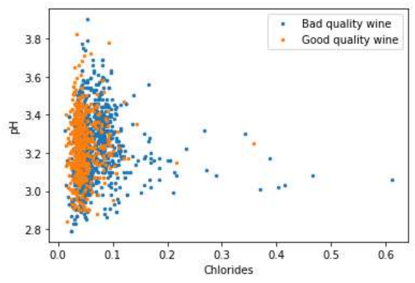

Bad quality wine
Good quality wine
Sulphates
Residual sugar

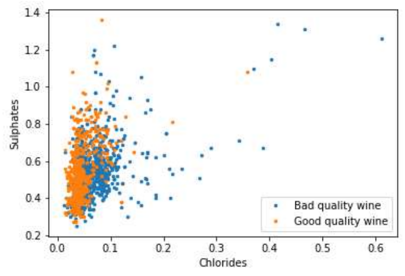

Bad quality wine
Good quality wine
Alcohol
Residual sugar

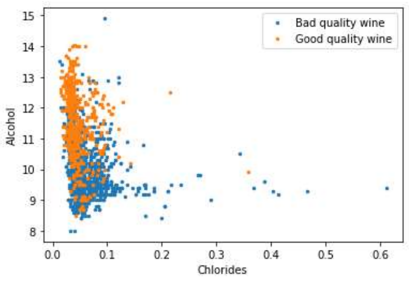

Bad quality wine
Good quality wine
Free sulfur dioxide
Chlorides

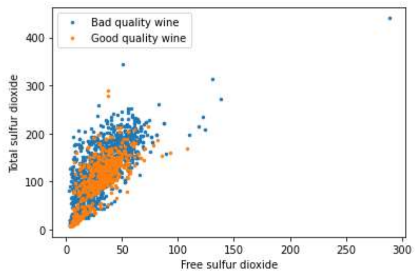

Bad quality wine
Good quality wine
Total sulfur dioxide
Chlorides

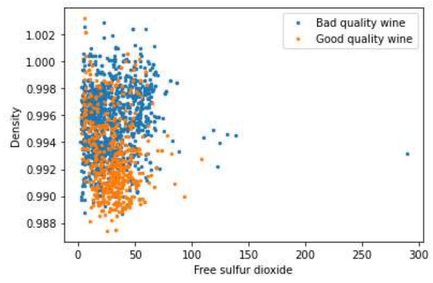

Density
Chlorides
Bad quality wine
Good quality wine

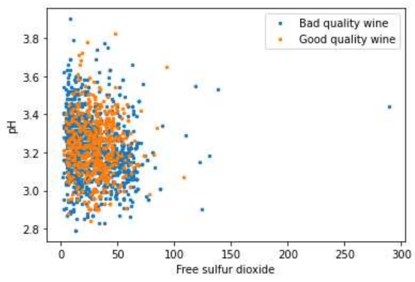

pH
Chlorides
Bad quality wine
Good quality wine

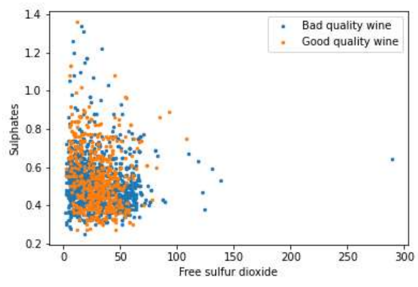

Sulphates
Chlorides
Bad quality wine
Good quality wine

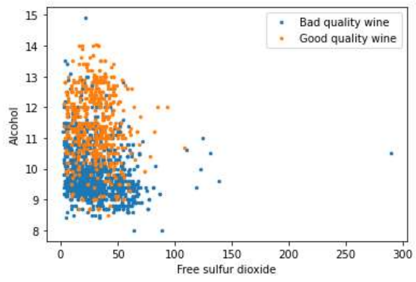

Alcohol
Chlorides
Bad quality wine
Good quality wine

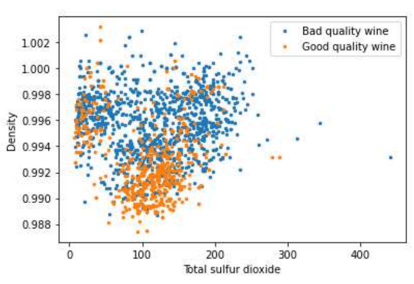

Total sulfur dioxide
Free sulfur dioxide
Bad quality wine
Good quality wine

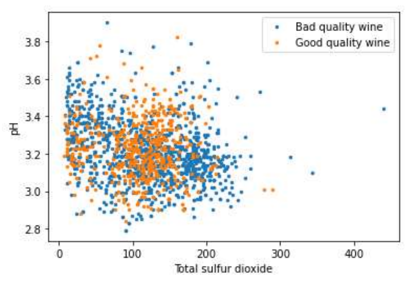

Density
Free sulfur dioxide
Bad quality wine
Good quality wine

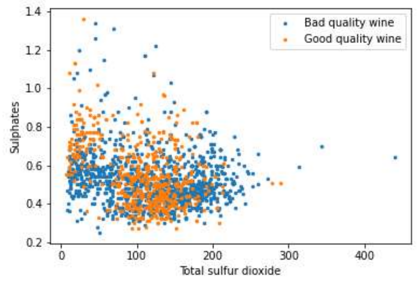

pH
Free sulfur dioxide
Bad quality wine
Good quality wine

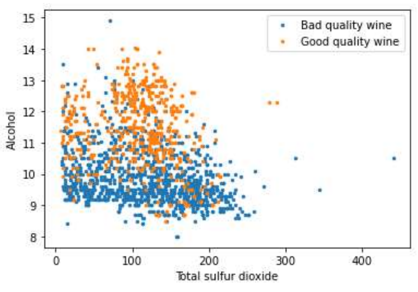

Sulphates
Free sulfur dioxide
Bad quality wine
Good quality wine

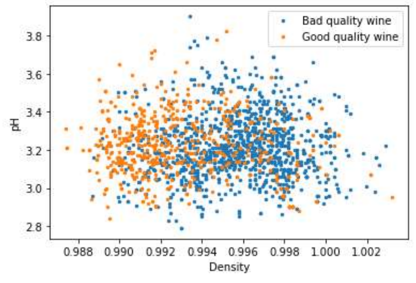

Alcohol
Free sulfur dioxide
Bad quality wine
Good quality wine

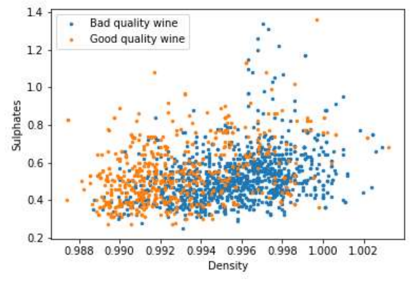

Density
Total sulfur dioxide
Bad quality wine
Good quality wine

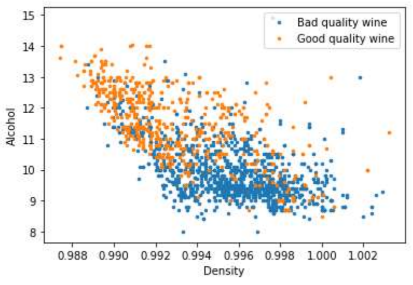

pH
Total sulfur dioxide
Bad quality wine
Good quality wine

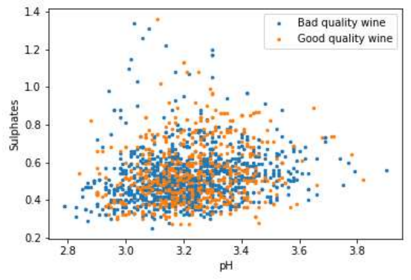

Sulphates
Total sulfur dioxide
Bad quality wine
Good quality wine

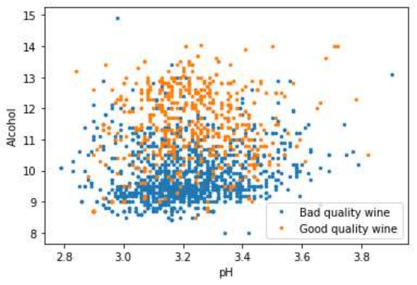

Alcohol
Total sulfur dioxide
Bad quality wine
Good quality wine

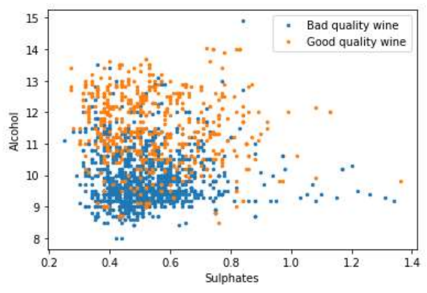

pH
Density
Bad quality wine
Good quality wine


Sulphates
Density
Bad quality wine
Good quality wine


Alcohol
Density
Bad quality wine
Good quality wine


Sulphates
pH
Bad quality wine
Good quality wine


Alcohol
pH
Bad quality wine
Good quality wine



Dimensionality Reduction

Dimensionality reduction techniques compute a mapping from the n-dimensional original features space (in this case, n = 11), to a m-dimensional space, with m<n. The goal of these methods are several: compress information (also to reduce the computational time), remove unwanted variability, simplify classification, data visualization (in case of m = 2 and m =3).

11 features are not few, so it may be reasonable to apply a dimensionality reduction algorithm, to compress the information, preserving the most useful for the classification. The method we adopted for this problem is the *PCA (Principal Component Analysis)*, a linear unsupervised technique that finds a subspace of $\mathbb{R}^n$. *PCA* projects the data over the principal components. These can be computed from the eigenvectors of the data covariance matrix *C* corresponding to the largest eigenvalues.

$$C = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

Where *N* is the number of samples, x_i is the *i*-th sample and μ is the dataset mean:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

C can be decomposed as:

$$C = U \Sigma U^T$$

Where *U* is a matrix, whose columns are the eigenvectors of *C* and *Σ* is a diagonal matrix, containing the eigenvalues in descending order.

$$U = [u_1 \dots u_m, u_{m+1} \dots u_n]$$

$$P = [u_1 \dots u_m]$$

P corresponds to the first m columns of *U*.

Finally, we can apply the projection to the initial n-dimensional matrix of samples, to obtain the dimensionality reduction and have a m-dimensional matrix.

$$Y = P^T X$$

From the scatter plots of the previous paragraph, it is possible to observe that there are some features correlated, particularly:

- *fixed acidity* and *density;*
- *residual sugar* and *density*;
- *alcohol* and *density*;
- *free sulfur dioxide* and *total sulfur dioxide*.

This information suggests that we may benefit from using *PCA* to map data to reduce the number of parameters to estimate. Given that there is the presence of four evident correlations, we will use m in the range [7, 10].

It is not possible to use *LDA (Linear Discriminant Analysis)* as a linear dimensionality reduction method, because it allows estimating at most C-1 directions, where C is the number of classes, but in this case, C is equal to 2.

Classifications

Classification consists in, given the feature vector representing an object, associate a label to the object based on properties of the representation; so, perform a mapping from the m-dimensional feature space to the space of labels. The mapping is also called decision function.

To evaluate and compare the different models that we will develop, we chosen to adopt the *K-fold cross validation.* This method can be employed to split the dataset in *K*, non-overlapping, subsets. We will then iteratively consider one subset as validation, and the remaining $K - 1$ subsets as training set. We will implement the *K-fold cross validation* with *K=5.*

We know that the original dataset has 10 classes, scores from 1 to 10, but wines with quality 6 have been discarded. Therefore, the dataset has been binarized, from the 9 (10 original classes minus the class of wine with quality 6) classes to all wines with low quality (scores from 1 to 5, i.e., the merge of five classes) into class 0, and good quality (scores from 7 to 10, i.e., the merge of four classes) into class 1. For this reason, we chosen a prior probability $\tilde{\pi} = 4/9$, as the prior probability that a wine belongs to the good quality wine class and $1-\tilde{\pi} = 5/9$, as the prior probability that a wine belongs to the bad quality wine. So, for the main application the prior is slightly biased towards the bad quality wine.

For what concerns the costs of miss-classified samples, we have decided to adopt *$C_{fp}=1$* and *$C_{fn}=1$*, so our application will be: *$(\tilde{\pi}, C_{fp}, C_{fn})= (4/9, 1, 1)$*.

We will also consider other two unbalanced applications: *$(\tilde{\pi}, C_{fp}, C_{fn})= (1/5, 1, 1)$*, *$(\tilde{\pi}, C_{fp}, C_{fn})= (4/5, 1, 1)$* where the prior highly is biased towards one of the two classes.

Our goal, for the moment, is to choose the most promising classification model. It may be useful knowing how good the model would perform if we had selected the best possible threshold, so it is possible to compute the (normalized) DCF (Detection Cost Function) using all possible thresholds and select its minimum value.

$$DCF(\tilde{\pi}, C_{fp}, C_{fn}) = \frac{DCF_u}{B_{dummy}} = \frac{\tilde{\pi}C_{fn}FNR + (1-\tilde{\pi})C_{fp}FPR}{min(\tilde{\pi}C_{fn}, (1-\tilde{\pi})C_{fp})}$$

To compute the minimum cost, we will consider a set of thresholds corresponding to $(-\infty, s_1 \dots s_n, +\infty)$, where $s_1 \dots s_n$ are the scores, sorted in increasing order. For each threshold t, we will compute the confusion matrix if scores were thresholded at t and the corresponding normalized DCF. The minimum DCF is the minimum of the obtained values.

Min DCF measures the cost we would pay if we made optimal decisions for test set (in this case, the validation set) using the recognizer scores: the cost we would pay if we knew before-hand the optimal threshold for the evaluation (in this case, validation); so, this value can be seen as a measure of the quality of the classifier.

The classification algorithm we will consider are:

- Multivariate Gaussian Classifier (MVG)
- Naïve Bayes Gaussian Classifier
- Tied Covariance Gaussian Classifier
- Naïve Tied Gaussian Classifier

- Logistic Regression (LR)

- Linear Support Vector Machines (Linear SVM)
- Polynomial Kernel Support Vector Machines (Poly SVM)
- Gaussian Radial Basis Function Kernel Support Vector Machines (RBF SVM)

- Gaussian Mixture Model (GMM)
- Diagonal Covariance Gaussian Mixture Model (Diag GMM)
- Tied Covariance Gaussian Mixture Model (Tied GMM)

Multivariate Gaussian Classifier (MVG)

The first model we will implement is the Multivariate Gaussian Classifier (MVG). This classifier is a generative model and assumes that samples of each class c, in our case ∈ {0, 1}, can be modelled as samples of a multivariate Gaussian distribution with class-dependent mean and covariance matrices.

$$f_{X|C}(x|c) = N(x|\mu_c, \Sigma_c)$$

where *N* is a normal distribution.

The ML solution for the parameters is given by the empirical mean and covariance matrix of each class:

$$\mu_c^* = \frac{1}{N_c}\sum_i x_{c,i}$$

$$\Sigma_c^* = \frac{1}{N_c}\sum_i (x_{c,i} - \mu_c^*)(x_{c,i} - \mu_c^*)^T$$

where $x_{c,i}$ is the i-th sample of the class c.

To carry out the computations useful for the calculation of the DCF, we use the log-likelihood ratio (llr), i.e., the logarithm of the ratio between the likelihood of observing the sample given that it belongs to class h_1 or to class h_0 (we denote the classes sometimes with *h* and sometimes with *c*):

$$llr(x_t) = log\frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)}$$

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
MVG, no PCA	0.445	0.685	0.811
MVG, with PCA (m=10)	0.406	0.693	0.741
MVG, with PCA (m=9)	0.399	0.695	0.684
MVG, with PCA (m=8)	0.398	0.684	0.716
MVG, with PCA (m=7)	0.418	0.685	0.749

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
MVG, “Gaussianization”, no PCA	0.465	0.726	0.782
MVG, “Gaussianization”, with PCA (m=10)	0.451	0.724	0.781
MVG, “Gaussianization”, with PCA (m=9)	0.461	0.759	0.802
MVG, “Gaussianization”, with PCA (m=8)	0.462	0.752	0.785
MVG, “Gaussianization”, with PCA (m=7)	0.447	0.701	0.804

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9,* $\tilde{\pi}$ = *1/5,* $\tilde{\pi}$ = *4/5).*

In general, the pre-processing of data through "Gaussianization" always results in worse results than those without.

Naïve Bayes Gaussian Classifier

The Naïve Bayes Gaussian Classifier is a “simplification” of the MVG because assumes that, for each class, the different components are independent. In practice, it is simply a Gaussian classifier where the covariance matrices are diagonal. The ML solution for the mean parameters is the same, whereas the ML solution for the covariance matrices is:

$$diag(\Sigma_c^*) = diag[\frac{1}{N_c}\sum_i (x_{c,i} - \mu_c^*)(x_{c,i} - \mu_c^*)^T]$$

i.e., the diagonal of the ML solution for the MVG model

As for the MVG, to carry out the computations useful for the calculation of the DCF, we use the log-likelihood ratio (llr).

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
Naïve Bayes, no PCA	0.494	0.742	0.875

Naïve Bayes, with PCA (m=10)	0.476	0.775	0.818
Naïve Bayes, with PCA (m=9)	0.462	0.767	0.833
Naïve Bayes, with PCA (m=8)	0.456	0.767	0.889
Naïve Bayes, with PCA (m=7)	0.454	0.772	0.883

	minimum DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*
Naïve Bayes, “Gaussianization”, no PCA	0.532	0.739	0.984
Naïve Bayes, “Gaussianization”, with PCA (m=10)	0.551	0.770	0.970
Naïve Bayes, “Gaussianization”, with PCA (m=9)	0.549	0.785	0.975
Naïve Bayes, “Gaussianization”, with PCA (m=8)	0.555	0.773	0.977
Naïve Bayes, “Gaussianization”, with PCA (m=7)	0.557	0.796	0.972

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9*, $\tilde{\pi}$ = *1/5*, $\tilde{\pi}$ = *4/5).*

In general, the pre-processing of data through "Gaussianization" always results in worse results than those without.

The performances, overall, are worse than the full-covariance MVG, and this could be related to the fact that the data are highly correlated, and, instead, the Naïve Bayes assumes that, for each class, the different components are independent.

Tied Covariance Gaussian Classifier

The Tied Gaussian Classifier is another version of the MVG and assumes that each class has its own mean, but the covariance matrix is the same for all classes. The ML solution for the covariance matrix is given by the empirical within-class covariance matrix:

$$\Sigma^* = \frac{1}{N}\sum_{c}\sum_{i|c_i=c}(x_{c,i} - \mu_c^*)(x_{c,i} - \mu_c^*)^T$$

where N is the number of samples.

As for the MVG, to carry out the computations useful for the calculation of the DCF, we use the log-likelihood ratio (llr).

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
Tied, no PCA	0.387	0.687	0.746
Tied, with PCA (m=10)	0.397	0.672	0.692
Tied, with PCA (m=9)	0.394	0.677	0.678
Tied, with PCA (m=8)	0.419	0.677	0.783
Tied, with PCA (m=7)	0.422	0.669	0.789

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
Tied, “Gaussianization”, no PCA	0.440	0.703	0.708
Tied, “Gaussianization”, with PCA (m=10)	0.445	0.708	0.698
Tied, “Gaussianization”, with PCA (m=9)	0.448	0.726	0.710
Tied, “Gaussianization”, with PCA (m=8)	0.470	0.729	0.724
Tied, “Gaussianization”, with PCA (m=7)	0.480	0.762	0.736

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = 4/9, $\tilde{\pi}$ = 1/5, $\tilde{\pi}$ = 4/5).

In general, the pre-processing of data through "Gaussianization" always results in worse results than those without.

The results, overall, are better than the previous and this may be linked to the fact that in the dataset there are great correlations and Tied Covariance model can capture these correlations. In general, we have seen that the classes have, more or less, similar distributions, so covariances should be similar: the model should provide a more dependable estimate.

Naïve Tied Gaussian Classifier

The Naïve Tied Gaussian Classifier put the two methods explained above together: it assumes that each class has its own mean, but the covariance matrix is the same for all classes and it is diagonal. The ML solution for the covariance matrix is given by:

$$diag(\Sigma^*) = diag[\frac{1}{N}\sum_c \sum_{i|c_i=c} (x_{c,i} - \mu_c^*)(x_{c,i} - \mu_c^*)^T]$$

i.e., the diagonal of the ML solution for the Tied Covariance Gaussian Classifier model.

As for the MVG, to carry out the computations useful for the calculation of the DCF, we use the log-likelihood ratio (llr).

We report in the tables below the obtained computational results:

	minimum DCF (*$\tilde{\pi}$ = 4/9)*	minimum DCF (*$\tilde{\pi}$ = 1/5)*	minimum DCF (*$\tilde{\pi}$ = 4/5)*
Naïve Tied, no PCA	0.459	0.739	0.922
Naïve Tied, with PCA (m=10)	0.421	0.679	0.684
Naïve Tied, with PCA (m=9)	0.424	0.679	0.692
Naïve Tied, with PCA (m=8)	0.448	0.684	0.735
Naïve Tied, with PCA (m=7)	0.452	0.675	0.739

	minimum DCF (*$\tilde{\pi}$ = 4/9)*	minimum DCF (*$\tilde{\pi}$ = 1/5)*	minimum DCF (*$\tilde{\pi}$ = 4/5)*
Naïve Tied, “Gaussianization”, no PCA	0.521	0.749	0.940
Naïve Tied, “Gaussianization”, with PCA (m=10)	0.445	0.731	0.733
Naïve Tied, “Gaussianization”, with PCA (m=9)	0.447	0.747	0.727
Naïve Tied, “Gaussianization”, with PCA (m=8)	0.475	0.721	0.749
Naïve Tied, “Gaussianization”, with PCA (m=7)	0.474	0.749	0.748

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for *$\tilde{\pi}$ = 4/9, $\tilde{\pi}$ = 1/5, $\tilde{\pi}$ = 4/5).*

In general, the pre-processing of data through "Gaussianization" always results in worse results than those without.

These performances, in general, are better than the best performance of the Naïve Bayes, but it is worse than the best of the full-covariance MVG and than the best performance of the Tied Covariance.

As already mentioned, pre-processing data through "Gaussianization" always results in worse results than those without. This could be related to the fact that most of the features are already more or less distributed as Gaussian and only few classes have outliers. Given that the "Gaussianization" did not give the desired results for the classifiers that should benefit most from it, this pre-processing step will not be used later for other classifiers that should not be too affected by the type of initial distribution of the data.

Logistic Regression

The Logistic regression is a discriminative approach for classification.

The regularized Logistic Regression objective can be written as follows:

$$J(w,b) = -l(w,b) = \frac{\lambda}{2}\|w\|^{2} + \frac{\pi_T}{n_T}\sum_{i|z_i=1}^{n} log\left(1+e^{-z_i(w^T x_i+b)}\right) + \frac{1-\pi_T}{n_T}\sum_{i|z_i=-1}^{n} log\left(1+e^{-z_i(w^T x_i+b)}\right),$$

$where\ z_i = 2c_i - 1$

The ML solution is the solution that minimize *J(w,b)*.

This is a prior-weighted version of the model; in fact, it considers the empirical class prior of the training data π_T. We will consider $\pi_T = 4/9$, $\pi_T = 1/5$ and $\pi_T = 4/5$.

For Logistic Regression models closed for expressions are not available for the ML solution. Nevertheless, we turn to numerical optimization, particularly, we will use the L-BFGS algorithm.

The scores *s* obtained computing w^* and b^* are equal to posterior-llr:

$$log\frac{P(C=h_1|x)}{P(C=h_0|x)} = log\frac{f_{X|C}(x|h_1)}{f_{X|C}(x|h_0)} + log\frac{\pi}{1-\pi} = w^T x + b = s$$

But, for the calculation of the actual DCF, we will use an optimal threshold for the llr and not for the posterior-llr, so we consider score s' equal to:

$$s' = s - log\frac{\pi}{1-\pi}$$

We will consider different values of λ from 1e-5 to until the DCF values worse again.

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
LR (λ=1e-5, π_T=4/9), no PCA	0.417	0.719	0.597
LR (λ=1e-5, π_T=4/9), with PCA (m=10)	0.417	0.715	0.604
LR (λ=1e-5, π_T=4/9), with PCA (m=9)	0.417	0.726	0.590
LR (λ=1e-5, π_T=4/9), with PCA (m=8)	0.420	0.685	0.678
LR (λ=1e-5, π_T=4/9), with PCA (m=7)	0.437	0.682	0.702
LR (λ=1e-4, π_T=4/9), no PCA	0.410	0.719	0.607
LR (λ=1e-4, π_T=4/9), with PCA (m=10)	0.411	0.716	0.600
LR (λ=1e-4, π_T=4/9), with PCA (m=9)	0.412	0.724	0.604
LR (λ=1e-4, π_T=4/9), with PCA (m=8)	0.416	0.682	0.684
LR (λ=1e-4, π_T=4/9), with PCA (m=7)	0.435	0.682	0.701
LR (λ=1e-3, π_T=4/9), no PCA	0.407	0.697	0.666
LR (λ=1e-3, π_T=4/9), with PCA (m=10)	0.406	0.697	0.665
LR (λ=1e-3, π_T=4/9), with PCA (m=9)	0.406	0.697	0.665
LR (λ=1e-3, π_T=4/9), with PCA (m=8)	0.420	0.695	0.697
LR (λ=1e-3, π_T=4/9), with PCA (m=7)	0.436	0.690	0.743
LR (λ=1e-2, π_T=4/9), no PCA	0.465	0.697	0.780
LR (λ=1e-2, π_T=4/9), with PCA (m=10)	0.465	0.700	0.780
LR (λ=1e-2, π_T=4/9), with PCA (m=9)	0.465	0.697	0.779
LR (λ=1e-2, π_T=4/9), with PCA (m=8)	0.471	0.698	0.797
LR (λ=1e-2, π_T=4/9), with PCA (m=7)	0.475	0.682	0.805

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
LR (λ=*1e-5*, π_T=*1/5*), no PCA	0.404	0.710	0.622
LR (λ=*1e-5*, π_T=*1/5*), with PCA (m=10)	0.407	0.718	0.613
LR (λ=*1e-5*, π_T=*1/5*), with PCA (m=9)	0.402	0.719	0.625
LR (λ=*1e-5*, π_T=*1/5*), with PCA (m=8)	0.409	0.692	0.744
LR (λ=*1e-5*, π_T=*1/5*), with PCA (m=7)	0.432	0.695	0.738
LR (λ=*1e-4*, π_T=*1/5*), no PCA	0.394	0.703	0.636
LR (λ=*1e-4*, π_T=*1/5*), with PCA (m=10)	0.393	0.716	0.631
LR (λ=*1e-4*, π_T=*1/5*), with PCA (m=9)	0.391	0.718	0.632
LR (λ=*1e-4*, π_T=*1/5*), with PCA (m=8)	0.414	0.692	0.759
LR (λ=*1e-4*, π_T=*1/5*), with PCA (m=7)	0.433	0.692	0.744
LR (λ=*1e-3*, π_T=*1/5*), no PCA	0.407	0.685	0.719
LR (λ=*1e-3*, π_T=*1/5*), with PCA (m=10)	0.405	0.688	0.718
LR (λ=*1e-3*, π_T=*1/5*), with PCA (m=9)	0.402	0.687	0.719
LR (λ=*1e-3*, π_T=*1/5*), with PCA (m=8)	0.424	0.684	0.782
LR (λ=*1e-3*, π_T=*1/5*), with PCA (m=7)	0.426	0.674	0.792

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
LR (λ=*1e-5*, π_T=*4/5*), no PCA	0.432	0.729	0.563
LR (λ=*1e-5*, π_T=*4/5*), with PCA (m=10)	0.433	0.734	0.569
LR (λ=*1e-5*, π_T=*4/5*), with PCA (m=9)	0.432	0.739	0.545
LR (λ=*1e-5*, π_T=*4/5*), with PCA (m=8)	0.440	0.711	0.639
LR (λ=*1e-5*, π_T=*4/5*), with PCA (m=7)	0.456	0.690	0.664
LR (λ=*1e-4*, π_T=*4/5*), no PCA	0.433	0.734	0.560
LR (λ=*1e-4*, π_T=*4/5*), with PCA (m=10)	0.431	0.728	0.554
LR (λ=*1e-4*, π_T=*4/5*), with PCA (m=9)	0.433	0.729	0.547

LR (λ=*1e-4*, π_T=*4/5)*, with PCA (m=8)	0.438	0.710	0.644
LR (λ=*1e-4*, π_T=*4/5)*, with PCA (m=7)	0.456	0.688	0.669
LR (λ=*1e-3*, π_T=*4/5)*, no PCA	0.428	0.700	0.605
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=10)	0.430	0.698	0.604
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=9)	0.426	0.695	0.603
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=8)	0.439	0.706	0.695
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=7)	0.449	0.701	0.697
LR (λ=*1e-2*, π_T=*4/5)*, no PCA	0.485	0.728	0.785
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=10)	0.485	0.728	0.785
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=9)	0.483	0.728	0.786
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=8)	0.493	0.732	0.805
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=7)	0.489	0.731	0.806

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9,* $\tilde{\pi}$ = *1/5,* $\tilde{\pi}$ = *4/5).*

Up to now, it is possible to observe that the models that in general perform better (Tied MVG and Linear LR) are **linear**.

Linear Support Vector Machines

Linear Support Vector Machines are linear classifiers that look for maximum margin separation hyperplanes. The (primal) SVM objective consists in minimizing:

$$J(w,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} max(0,\ \ 1 - z_i(w^T x + b))$$

where n is the number of training samples, C is a hyper-parameter and z_i is the class label for the ith sample, encoded as:

$$z_i = \begin{cases} +1, & if\ c_i = 1 \\ -1, & if\ c_i = 0 \end{cases}$$

To solve the SVM problem, it is possible to consider the dual formulation:

$$J^D(\alpha) = -\frac{1}{2}\,\alpha^T H\,\alpha + \alpha^T 1$$

$$s.t.\ 0 \le \alpha_i \le C, \forall\, i\, \in \{1...n\}, \qquad \sum_{i=0}^{n} \alpha_i z_i = 0$$

where **1** is a n-dimensional vector of ones, and H is a matrix whose elements are:

$$H_{i,j} = z_i z_j x_i^T x_j$$

The SVM dual solution is the maximizer of $J^D(\alpha)$. The dual and primal solutions α^* and w^* are related through:

$$w^* = \sum_{i=1}^{n} \alpha_i^* z_i x_i$$

and the optimal bias b^* can be computed considering a sample x_i that lies on the margin:

$$z_i({w^*}^T x_i + b^*) = 1$$

and then solving for b^*.

It is possible to slightly modify the SVM problem to employ L-BFGS-B to solve the dual problem:

$$\hat{J}(\hat{w}) = \frac{1}{2}\,\|\hat{w}\|^2 + C\sum_{i=1}^{n} max(0, 1 - z_i(\hat{w}^T \hat{x}_i))$$

where

$$\hat{x}_i = \frac{x_i}{k}, \quad \hat{w} = \frac{w}{b}$$

where k is a constant that contrasts the effect of regularizing b

The dual objective of the modified primal SVM becomes the maximization of:

$$J^D(\alpha) = -\frac{1}{2}\,\alpha^T \widehat{H}\,\alpha + \alpha^T 1$$

$$s.t.\ 0 \le \alpha_i \le C, \forall\, i\, \in \{1...n\}$$

where $\widehat{H}$ is a matrix whose elements are:

$$\widehat{H}_{i,j} = z_i z_j \hat{x}_i^T \hat{x}_j$$

To carry out the computations useful for the calculation of the DCF, we use the scores:

$$s(x_t) = \widehat{w}^{*T} \hat{x}_t$$

For our computations, we will fix the value of k to 1 and we will test the model with different value of C (0.001, 0.01, 0.1, 1, 10).

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
SVM (C= 0.001, k=1), no PCA	0.799	0.997	1.013
SVM (C= 0.001, k=1), with PCA (m=10)	0.809	0.997	1.013
SVM (C= 0.001, k=1), with PCA (m=9)	0.798	0.997	1.013
SVM (C= 0.001, k=1), with PCA (m=8)	0.809	0.997	1.013
SVM (C= 0.001, k=1), with PCA (m=7)	0.836	0.997	0.999
SVM (C= 0.01, k=1), no PCA	0.519	0.763	0.954
SVM (C= 0.01, k=1), with PCA (m=10)	0.524	0.775	0.962
SVM (C= 0.01, k=1), with PCA (m=9)	0.520	0.770	0.964
SVM (C= 0.01, k=1), with PCA (m=8)	0.529	0.778	0.959
SVM (C= 0.01, k=1), with PCA (m=7)	0.545	0.811	0.961
SVM (C= 0.1, k=1), no PCA	0.460	0.746	0.763
SVM (C= 0.1, k=1), with PCA (m=10)	0.512	0.768	0.909
SVM (C= 0.1, k=1), with PCA (m=9)	0.476	0.742	0.749
SVM (C= 0.1, k=1), with PCA (m=8)	0.481	0.773	0.834
SVM (C= 1, k=1), with PCA (m=7)	0.488	0.790	0.874
SVM (C= 1, k=1), no PCA	0.602	0.834	0.914
SVM (C= 1, k=1), with PCA (m=10)	0.630	0.837	0.966
SVM (C= 1, k=1), with PCA (m=9)	0.538	0.824	0.928
SVM (C= 1, k=1), with PCA (m=8)	0.499	0.757	0.783
SVM (C= 1, k=1), with PCA (m=7)	0.647	0.992	0.931
SVM (C= 10, k=1), no PCA	0.847	0.982	0.977

SVM (C= 10, k=1), with PCA (m=10)	0.920	0.976	0.966
SVM (C= 10, k=1), with PCA (m=9)	1	1	0.99
SVM (C= 10, k=1), with PCA (m=8)	0.943	1	0.972
SVM (C= 10, k=1), with PCA (m=7)	0.878	1	0.968

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for *$\tilde{\pi}$ = 4/9, $\tilde{\pi}$ = 1/5, $\tilde{\pi}$ = 4/5).*

The best performance for the Linear SVM, for the considered hyperparameter and constant, is obtained for C= 0.1, k=1.

Even if the Linear SVM is a liner model, it does not performs well as Tied MVG and Logistic Regression.

Kernel Support Vector Machines

A kernel functions allows training a SVM in a large dimensional Hilbert space, without requiring to explicitly compute the mapping. The complexity of the primal problem may be too large, but the complexity of the dual problem depends only on the number of training points. In practice, it is possible to compute a linear separation in the expanded space, which corresponds to a non-linear separation surface in the original feature space.

Practically, for Kernel SVM, $\widehat{H}$ is a matrix whose elements are:

$$\widehat{H}_{i,j} = z_i z_j k(x_i, x_j)$$

As stated above, we cannot compute the primal solution, so, we cannot compute the scores in the same way as for the Linear SVM. However, it is possible to compute the score in the following way:

$$s(x_t) = \sum_{i=1}^{n} \alpha_i^* z_i k(x_i, x_t)$$

We will use these scores to carry out the computations useful for the calculation of the DCF.

Polynomial Kernel Support Vector Machines

For the Polynomial Kernel Support Vector Machines, the kernel is:

$$k(x_1, x_2) = (x_1^T x_2 + c)^d$$

To add a regularized bias, we add a constant value k to the kernel function:

$$\hat{k}(x_1, x_2) = k(x_1, x_2) + k$$

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
Poly (C= 0.001, d=2, c=1, k=1), no PCA	0.887	1	0.958
Poly (C= 0.001, d=2, k=1), with PCA (m=10)	0.939	1	0.995
Poly (C= 0.001, d=2, k=1), with PCA (m=9)	0.908	0.992	0.961
Poly (C= 0.001, d=2, k=1), with PCA (m=8)	0.859	1	0.988
Poly (C= 0.001, d=2, c=1, k=1), with PCA (m=7)	0.826	0.993	0.992
Poly (C= 0.01, d=2, c=1, k=1), no PCA	0.928	1	0.985
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=10)	0.997	0.997	0.995
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=9)	0.993	1	0.985
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=8)	0.994	1	0.965
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=7)	1	1	0.996
Poly (C= 0.1, d=2, c=1, k=1), no PCA	0.896	0.992	0.940
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=10)	0.955	1	0.954
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=9)	1	1	0.923
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=8)	0.923	1	0.985
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=7)	1	1	1
Poly (C= 1, d=2, c=1, k=1), no PCA	0.982	0.994	1
Poly (C= 1, d=2, c=1, k=1), with PCA (m=10)	0.927	0.887	1

Poly (C= 1, d=2, c=1, k=1), with PCA (m=9)	0.967	1	1
Poly (C= 1, d=2, c=1, k=1), with PCA (m=8)	0.980	0.978	0.982
Poly (C= 1, d=2, c=1, k=1), with PCA (m=7)	0.875	1	0.986
Poly (C= 10, d=2, c=1, k=1), no PCA	0.988	1	0.989
Poly (C= 10, d=2, c=1, c=1, k=1), with PCA (m=10)	1	1	0.963
Poly (C= 10, d=2, c=1, k=1), with PCA (m=9)	0.892	0.959	0.982
Poly (C= 10, d=2, c=1, k=1), with PCA (m=8)	0.996	0.997	0.983
Poly (C= 10, d=2, c=1, k=1), with PCA (m=7)	0.866	0.985	0.982

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

Gaussian Radial Basis Function Kernel Support Vector Machines

For the Gaussian Radial Basis Function Kernel Support Vector Machines, the kernel is:

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

To add a regularized bias, we add a constant value k to the kernel function:

$$\hat{k}(x_1, x_2) = k(x_1, x_2) + k$$

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
RBF (C= 0.001, $\gamma=e^{-3}$, k=1), no PCA	1	1	0.997
RBF (C= 0.001, $\gamma=e^{-3}$, k=1), with PCA (m=10)	1	1	0.997
RBF (C= 0.001, $\gamma=e^{-3}$, k=1), with PCA (m=9)	1	1	0.997
RBF (C= 0.001, $\gamma=e^{-3}$, k=1), with PCA (m=8)	1	1	0.997
RBF (C= 0.001, $\gamma=e^{-3}$, k=1), with PCA (m=7)	1	1	0.997

RBF (C= 0.01, $\gamma=e^{-3}$, k=1), no PCA	0.766	0.904	0.954
RBF (C= 0.1, $\gamma=e^{-3}$, k=1), no PCA	0.759	0.896	0.978
RBF (C= 1, $\gamma=e^{-3}$, k=1), no PCA	0.730	0.878	0.958
RBF (C= 10, $\gamma=e^{-3}$, k=1), no PCA	0.704	0.915	0.984
RBF (C= 0.001, $\gamma=e^{-2}$, k=1), no PCA	0.998	0.998	0.993
RBF (C= 0.01, $\gamma=e^{-2}$, k=1), no PCA	0.895	1	0.977
RBF (C= 0.1, $\gamma=e^{-2}$, k=1), no PCA	0.791	0.933	0.986
RBF (C= 1, $\gamma=e^{-2}$, k=1), no PCA	0.762	0.935	0.978
RBF (C= 10, $\gamma=e^{-2}$, k=1), no PCA	0.746	0.962	0.996
RBF (C= 0.001, $\gamma=e^{-1}$, k=1), no PCA	0.998	0.998	0.983
RBF (C= 0.01, $\gamma=e^{-1}$, k=1), no PCA	0.846	0.953	0.965
RBF (C= 0.1, $\gamma=e^{-1}$, k=1), no PCA	0.840	0.953	0.965
RBF (C= 1, $\gamma=e^{-1}$, k=1), no PCA	0.824	0.933	0.967
RBF (C= 10, $\gamma=e^{-1}$, k=1), no PCA	0.837	0.949	0.979
RBF (C= 0.001, $\gamma=1$, k=1), no PCA	1	1	0.987
RBF (C= 0.01, $\gamma=1$, k=1), no PCA	0.932	0.984	0.971
RBF (C= 0.1, $\gamma=1$, k=1), no PCA	0.935	0.985	0.974
0.974RBF (C= 1, $\gamma=1$, k=1), no PCA	0.900	0.969	0.974
RBF (C= 10, $\gamma=1$, k=1), no PCA	0.919	0.974	0.970
RBF (C= 0.001, $\gamma=e$, k=1), no PCA	1	1	0.987
RBF (C= 0.01, $\gamma=e$, k=1), no PCA	0.989	0.989	0.986
RBF (C= 0.1, $\gamma=e$, k=1), no PCA	0.989	0.989	0.986
RBF (C= 1, $\gamma=e$, k=1), no PCA	0.984	0.989	0.986
RBF (C= 10, $\gamma=e$, k=1), no PCA	0.98	0.989	0.986
RBF (C= 0.001, $\gamma=e^{2}$, k=1), no PC	1	1	0.988
RBF (C= 0.01, $\gamma=e^{2}$, k=1), no PCA	0.988	0.989	0.986
RBF (C= 0.1, $\gamma=e^{2}$, k=1), no PCA	0.988	0.989	0.987

RBF (C= 1, γ=e^2, k=1), no PCA	0.989	0.989	0.986
RBF (C= 10, γ=e^2, k=1), no PCA	0.976	0.987	0.986

At one point, we no longer used the PCA, because we realized that it did not significantly affect performances.

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

Gaussian Mixture Model

Gaussian Mixture Models are models obtained as a weighted combination of Gaussians; in this way, it is possible to approximate the density of a Random Variable X when the density of X is not known:

$$X \sim GMM(M, S, w) \implies f_X(x) = \sum_{g=1}^{M} w_g N(x|\mu_g, \Sigma_g)$$

where

$$M = [\mu_1 \dots \mu_M], \qquad S = [\Sigma_1 \dots \Sigma_M], \qquad w = [w_1 \dots w_M]$$

are the components means, covariance matrices and weights, respectively.

GMM can be interpreted as the marginal distribution obtained by marginalizing the joint density:

$$f_{X_i,G_i}(x_i, g) = w_g N(x_i|\mu_g, \Sigma_g), \qquad f_{X_i}(x_i) = \sum_{g=1}^{M} f_{X_i,G_i}(x_i, g)$$

G$_i$ is a discrete hidden random variable that represents the Gaussian component that was responsible for the generation of x$_i$. The joint density can be expressed as a product of the component (cluster) conditional distribution for X$_i$ and the prior distribution for G$_i$:

$$f_{X_i|G_i}(x_i|g) = N(x_i|\mu_g, \Sigma_g), \qquad P(G_i = g) = w_g, \qquad f_{X_i,G_i}(x_i, g) = f_{X_i|G_i}(x_i|g)P(G_i = g)$$

The EM (Expectation-Maximization) algorithm can be used to estimate the parameters of a GMM that maximize the likelihood for a training set X. The EM algorithm consists of two steps:

- E-step: compute the posterior probability for each component of the GMM for each sample, using an estimate (M$_t$, S$_t$, w$_t$) of the model parameters. These quantities are also called responsibilities:

$$\gamma_{g,i} = P(G_i = g|X_i = x_i, M_t, S_t, w_t)$$

- M-step: Update the model parameters. It is possible to use the statistics:

$$Z_g = \sum_{i=1}^{N} \gamma_{g,i}\,, \quad F_g = \sum_{i=1}^{N} \gamma_{g,i} x_i\,, \quad S_g = \sum_{i=1}^{N} \gamma_{g,i} x_i x_i^T$$

  to obtain the new parameters:

$$\mu_{g_{t+1}} = \frac{F_g}{Z_g}, \quad \Sigma_{g_{t+1}} = \frac{S_g}{Z_g} - \mu_{g_{t+1}} \mu_{g_{t+1}}^T\,, \quad w_{g_{t+1}} = \frac{Z_g}{\sum_{g'=1}^{M} Z_{g'}}$$

The EM algorithm requires an initial guess for the GMM parameters. The Linde-Buzo-Gray (LBG) algorithm allows to incrementally construct a GMM with 2G components from a GMM with G components. Starting with a single-component GMM (i.e., a Gaussian density), we can build a 2-component GMM and then use the EM algorithm to estimate a ML solution for the 2-components model. We can then split the 2 components to obtain a 4-components GMM, and re-apply the EM algorithm to estimate its parameters, and so on.

We will use the Maximum Likelihood solution for a Gaussian density as starting point: (w, μ, Σ) = (1, μ, Σ), with μ, Σ mean and covariance matrix of the samples of the class in question (obviously, we will apply the model to both classes separately).

The way we will adopt to split the GMM consists in replacing component (w_g, μ_g, Σ_g) with two components:

$$\left(\frac{w_g}{2}, \mu_g + d_g, \Sigma_g\right), \left(\frac{w_g}{2}, \mu_g - d_g, \Sigma_g\right)$$

We will compute the displacement vector by taking the leading eigenvector of Σ_g, scaled by the square root of the corresponding eigenvalue, multiplied by some factor α (we will adopt α=1).

The GMM log-likelihood is not bounded above for $M \geq 2$. Indeed, we can have arbitrarily high log-likelihood by centring one component at one of the N samples and letting the corresponding covariance matrix shrink towards zero. To avoid these kinds of degenerate solutions, it is possible to constrain the minimum values of the eigenvalues of the covariance matrices. A possible solution consists in constraining the eigenvalues of the covariance matrices to be larger or equal to a lower bound $\psi > 0$. We will use ψ=0.01.

As mentioned, we will train a *GMM(M_c, S_c, w_c)* for each class, and then use the GMM log-density function to compute class-conditional distributions:

$$f_{X_t,C_t}(x_t, c_t) = GMM(x_t | M_c, S_c, w_c)$$

for all classes.

To carry out the computations useful for the calculation of the DCF, we use the log-likelihood ratio (llr), i.e., the logarithm of the ratio between the likelihood of observing the sample given that it belongs to class h_1 or to class h_0 (again, we denote the classes sometimes with *h* and sometimes with *c*):

$$llr(x_t) = log\frac{f_{X|C}(x_t|h_1)}{f_{X|C}(x_t|h_0)}$$

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
GMM, 2 components, no PCA	0.781	1	0.999
GMM, 2 components, with PCA (m=10)	0.783	1	0.999
GMM, 2 components, with PCA (m=9)	0.782	1	0.999
GMM, 2 components, with PCA (m=8)	0.780	1	0.994
GMM, 2 components, with PCA (m=7)	0.814	1	0.997
GMM, 4 components, no PCA	0.768	0.938	0.998
GMM, 4 components, with PCA (m=10)	0.768	0.938	0.998
GMM, 4 components, with PCA (m=9)	0.767	0.941	0.998
GMM, 4 components, with PCA (m=8)	0.771	0.974	0.995
GMM, 4 components, with PCA (m=7)	0.821	1	0.992
GMM, 8 components, no PCA	0.787	0.974	1.054
GMM, 8 components, with PCA (m=10)	0.787	0.974	1.054
GMM, 8components, with PCA (m=9)	0.816	0.976	1.061
GMM, 8 components, with PCA (m=8)	0.955	1	1.042
GMM, 8 components, with PCA (m=7)	0.955	1	1.092

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9*, $\tilde{\pi}$ = *1/5*, $\tilde{\pi}$ = *4/5).*

Diagonal Covariance Gaussian Mixture Model

The Diagonal Covariance Gaussian Mixture Model consists of a model whose components all have diagonal covariance matrices; we will replace $\Sigma_{g_{t+1}}$ with:

$$diag(\Sigma_{g_{t+1}})$$

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
Diag GMM, 2 components, no PCA	0.653	1	0.905
Diag GMM, 2 components, with PCA (m=10)	0.735	0.922	0.940
Diag GMM, 2 components, with PCA (m=9)	0.737	0.922	0.940
Diag GMM, 2 components, with PCA (m=8)	0.744	0.936	0.934
Diag GMM, 2 components, with PCA (m=7)	0.743	0.920	0.936
Diag GMM, 4 components, no PCA	0.833	1	0.966
Diag GMM, 4 components, with PCA (m=10)	0.911	1	0.962
Diag GMM, 4 components, with PCA (m=9)	0.910	1	0.959
Diag GMM, 4 components, with PCA (m=8)	0.904	1	0.962
Diag GMM, 4 components, with PCA (m=7)	0.902	1	0.960
Diag GMM, 8 components, no PCA	0.933	1	0.979
Diag GMM, 8 components, with PCA (m=10)	0.918	1	1.386
Diag GMM, 8 components, with PCA (m=9)	0.919	1	1.391
Diag GMM, 8 components, with PCA (m=8)	0.829	1	1.389

Diag GMM, 8 components, with PCA (m=7)	0.882	1	1.385

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

Tied Covariance Gaussian Mixture Model

The Tied Covariance Gaussian Mixture Model consists of a model whose components has a covariance matrix $\Sigma_g = \Sigma$; we will replace $\Sigma_{g_{t+1}}$ with:

$$\frac{1}{N}\sum_{g=1}^{M} Z_g \Sigma_{g_{t+1}}$$

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Tied GMM, 2 components, no PCA	0.385	0.680	0.688
Tied GMM, 2 components, with PCA (m=10)	0.385	0.680	0.688
Tied GMM, 2 components, with PCA (m=9)	0.386	0.685	0.688
Tied GMM, 2 components, with PCA (m=8)	0.405	0.687	0.740
Tied GMM, 2 components, with PCA (m=7)	0.404	0.706	0.785
Tied GMM, 4 components, no PCA	0.471	0.881	0.883
Tied GMM, 4 components, with PCA (m=10)	0.471	0.881	0.883
Tied GMM, 4 components, with PCA (m=9)	0.472	0.883	0.883
Tied GMM, 4 components, with PCA (m=8)	0.496	0.927	0.992
Tied GMM, 4 components, with PCA (m=7)	0.499	0.869	0.902

Tied GMM, 8 components, no PCA	0.570	0.824	0.908
Tied GMM, 8 components, with PCA (m=10)	0.587	0.824	0.908
Tied GMM, 8 components, with PCA (m=9)	0.579	0.817	0.916
Tied GMM, 8 components, with PCA (m=8)	0.594	0.990	0.990
Tied GMM, 8 components, with PCA (m=7)	0.583	0.860	0.998

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ *= 4/9,* $\tilde{\pi}$ *= 1/5,* $\tilde{\pi}$ *= 4/5).*

Actual Detection Cost Function

After evaluating the considered models with the minimum DCF, we will calculate the actual Detection Cost Function for the classifiers that obtained the best performances in terms of minimum DCF (we have chosen the best models by observing the performances given by all the $\tilde{\pi}$ considered and giving slightly more weight to the performances given by $\tilde{\pi}$ *= 4/9*, i.e., the prior that we chose for our main application).

We obtained the best results for the Tied Covariance Multivariate Gaussian Classifier, for the Linear Logistic Regression and for the Tied Gaussian Mixture Model with 2 components (the specific values of m for PCA and the hyperparameter λ will be shown in the table below).

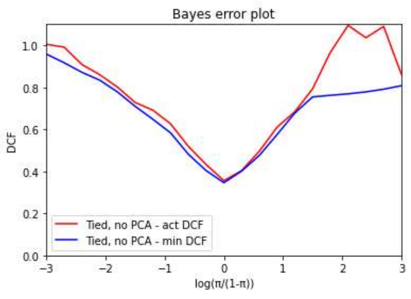
The actual DCF is calculated, if scores are well calibrated, using the optimal threshold that optimizes the Bayes risk:

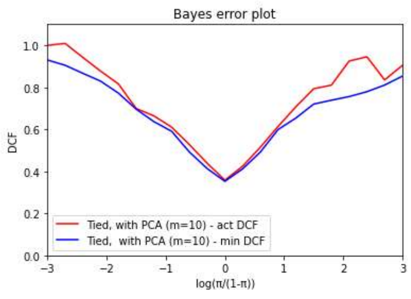
$$t = -\, log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

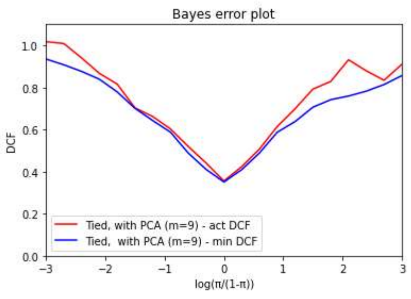
This threshold should be the optimal threshold if the considered scores are log-likelihood ratio: this should not be a problem for the Gaussian classifiers and the Gaussian Mixture Model, because we considered scores as llrs; furthermore, as mentioned, also for the Logistic Regression we used scores that are llrs (subtracting the log-odds); thus, the scores should be, more or less, well calibrated.

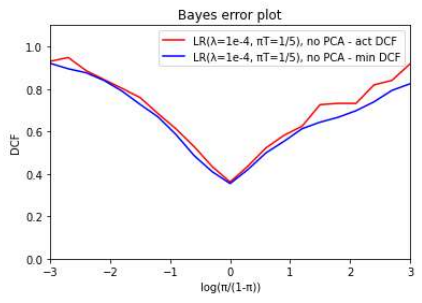
	minimum DCF ($\tilde{\pi}$ = *4/9*)	actual DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	actual DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)	actual DCF ($\tilde{\pi}$ = *4/5*)
Tied, no PCA	0.387	0.420	0.687	0.698	0.746	0.750
Tied, with PCA (m=10)	0.397	0.414	0.672	0.688	0.692	0.758
Tied, with PCA (m=9)	0.394	0.410	0.677	0.697	0.678	0.757
LR (λ=*1e-4*, π_T=*1/5)*, no PCA	0.394	0.403	0.703	0.706	0.636	0.679
LR (λ=*1e-4*, π_T=*1/5)*, with PCA (m=10)	0.393	0.404	0.716	0.721	0.631	0.704
LR (λ=*1e-4*, π_T=*1/5)*, with PCA (m=9)	0.391	0.399	0.718	0.728	0.632	0.678
Tied GMM, 2 components, no PCA	0.385	0.428	0.680	0.814	0.688	0.713
Tied GMM, 2 components, with PCA (m=10)	0.385	0.428	0.680	0.814	0.688	0.713
Tied GMM, 2 components, with PCA (m=9)	0.386	0.429	0.685	0.814	0.688	0.714

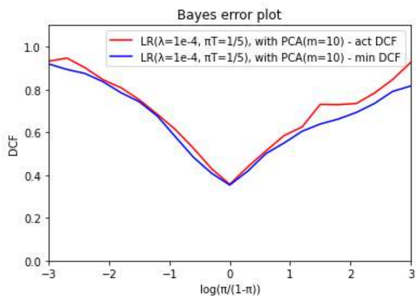
Thanks to the Bayes error plot, it is possible to report the actual and the minimum DCF for different applications: specifically, in the graphs below, we plot it for each of the models considered in the table above:

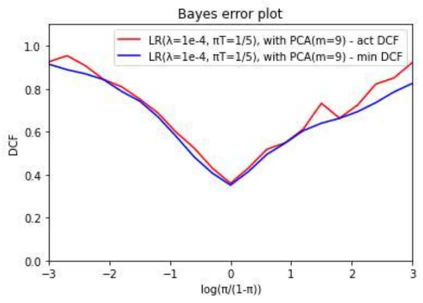


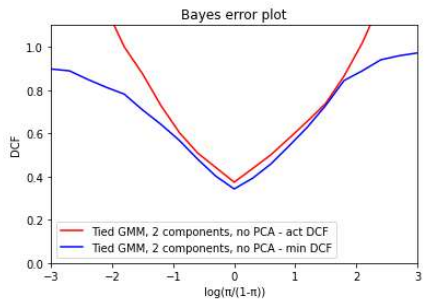


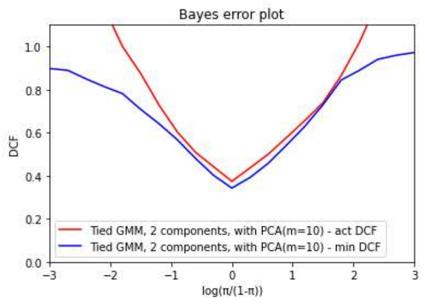


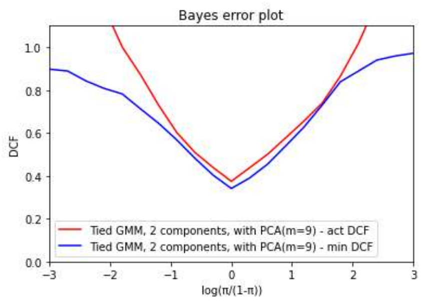












It is possible to notice that for Tied GMM with 2 components, scores are not perfectly calibrated for some applications, as it can be observed in the graphs above (the trends of actual and minimum DCF are not very similar, in some ranges).

Fusion

We have obtained the best performances for the aforementioned models; since we considered as scores for these classifiers the llrs and, furthermore, these scores are almost well calibrated, it is reasonable to think that the combination (fusion) of the decisions of these classifiers could lead to improved performances or, in any case, similar to those obtained by them.

An effective fusion approach consists in performing score-level fusion, assuming that the fused score is a function of the scores of different classifiers. If $s_{t,A}$, $s_{t,b}$, $s_{t,c}$ are the scores of classifier A, B and C for the sample x_t, the fused score for the sample x_t will be:

$$s_t = f(s_{t,A}, s_{t,B}, s_{t,C})$$

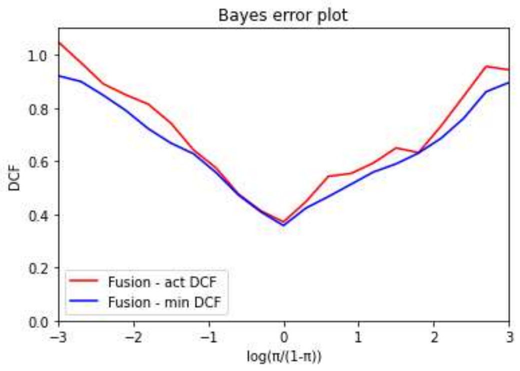
We will try to fuse the models in different ways.

The first score for the sample x_t will be computed as follows:

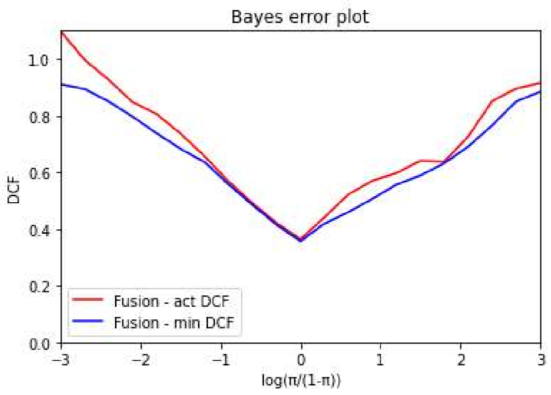
$$s_t = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \alpha_C s_{t,C}$$

Where A, B, C are, respectively, Tied MVG, LR (*λ=1e-4*, *π_T=1/5)* and Tied GMM (2 components); we will give to each classifier the same weight α =1/3, considering firstly the scores obtained from all the models without PCA, then with PCA (m=10) and finally with PCA (m=9):

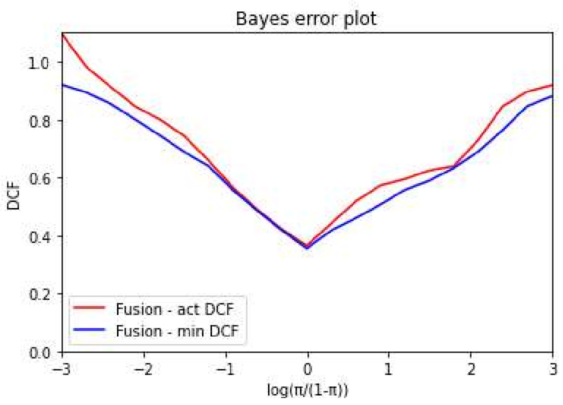
	minimum DCF (*$\tilde{\pi}$ = 4/9)*	actual DCF (*$\tilde{\pi}$ = 4/9)*	minimum DCF (*$\tilde{\pi}$ = 1/5)*	actual DCF (*$\tilde{\pi}$ = 1/5)*	minimum DCF (*$\tilde{\pi}$ = 4/5)*	actual DCF (*$\tilde{\pi}$ = 4/5)*
Fusion (3 models, no PCA)	**0.394**	0.411	**0.651**	0.703	**0.577**	0.631



	minimum DCF (*$\tilde{\pi}$ = 4/9)*	actual DCF (*$\tilde{\pi}$ = 4/9)*	minimum DCF (*$\tilde{\pi}$ = 1/5)*	actual DCF (*$\tilde{\pi}$ = 1/5)*	minimum DCF (*$\tilde{\pi}$ = 4/5)*	actual DCF (*$\tilde{\pi}$ = 4/5)*
Fusion (3 models, with PCA (m=10))	0.400	0.408	0.666	0.695	0.575	0.635

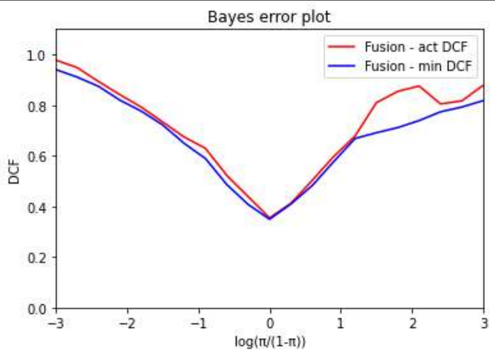


	minimum DCF (*$\tilde{\pi}$ = 4/9)*	actual DCF (*$\tilde{\pi}$ = 4/9)*	minimum DCF (*$\tilde{\pi}$ = 1/5)*	actual DCF (*$\tilde{\pi}$ = 1/5)*	minimum DCF (*$\tilde{\pi}$ = 4/5)*	actual DCF (*$\tilde{\pi}$ = 4/5)*
Fusion (3 models, with PCA (m=9))	0.399	0.404	0.669	0.705	0.556	0.629

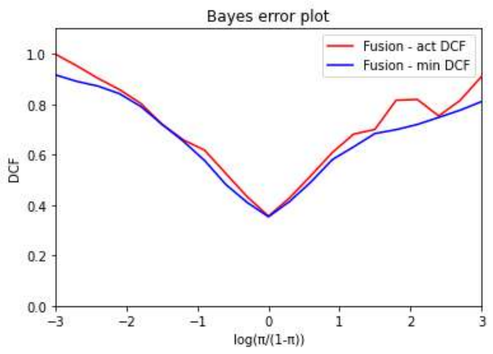


We will try also to combine the scores of the pairs obtained from the three classifiers considered, giving again a balanced weight (in this case, α =1/2 for both models) and considering again firstly the scores obtained from the models without PCA, then with PCA (m=10) and finally with PCA (m=9):

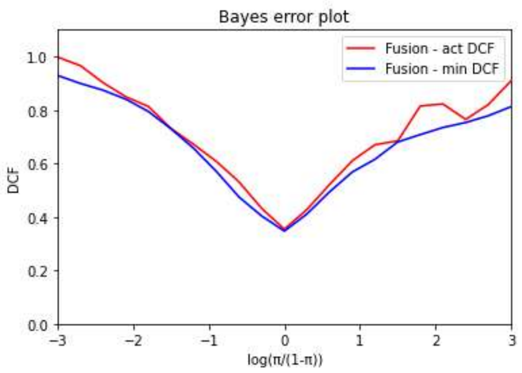
	minimum DCF (*$\tilde{\pi}$ = 4/9)*	actual DCF (*$\tilde{\pi}$ = 4/9)*	minimum DCF (*$\tilde{\pi}$ = 1/5)*	actual DCF (*$\tilde{\pi}$ = 1/5)*	minimum DCF (*$\tilde{\pi}$ = 4/5)*	actual DCF (*$\tilde{\pi}$ = 4/5)*
Fusion (MVG, LR), no PCA)	0.391	0.411	0.692	0.706	0.684	0.750



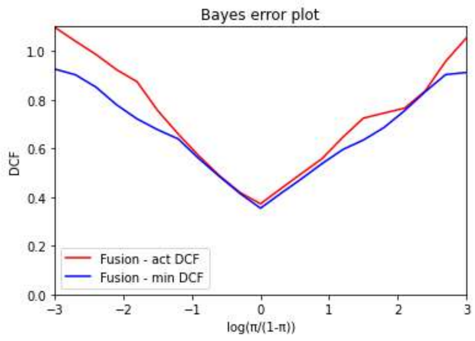
	minimum DCF ($\tilde{\pi}$ = *4/9)*	actual DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	actual DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*	actual DCF ($\tilde{\pi}$ = *4/5)*
Fusion (MVG, LR), with PCA (m=10))	0.394	0.398	0.695	0.698	0.668	0.712



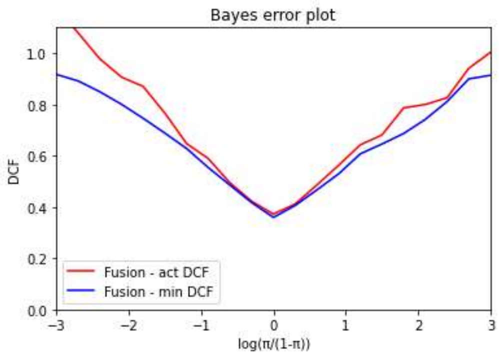
	minimum DCF ($\tilde{\pi}$ = *4/9)*	actual DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	actual DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*	actual DCF ($\tilde{\pi}$ = *4/5)*
Fusion (MVG, LR), with PCA (m=9))	0.387	0.402	0.703	0.713	0.653	0.670



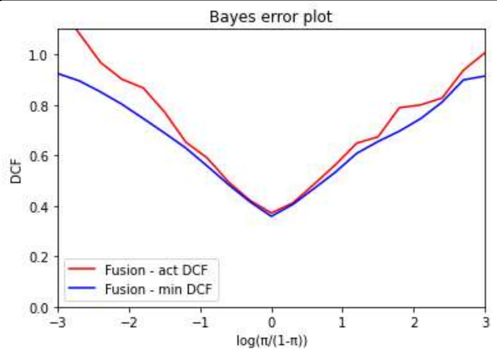
	minimum DCF ($\tilde{\pi}$ = *4/9)*	actual DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	actual DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*	actual DCF ($\tilde{\pi}$ = *4/5)*
Fusion (MVG, GMM), no PCA)	0.398	0.403	0.661	0.719	0.618	0.692



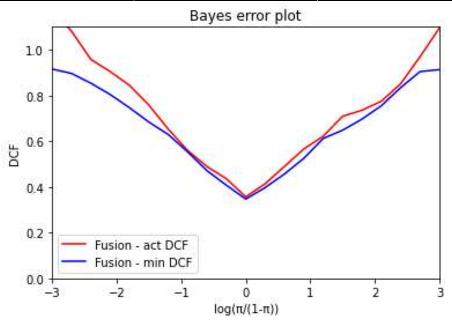
	minimum DCF ($\tilde{\pi}$ = *4/9)*	actual DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	actual DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*	actual DCF ($\tilde{\pi}$ = *4/5)*
Fusion (MVG, GMM), with PCA (m=10))	0.401	0.412	0.666	0.718	0.631	0.693



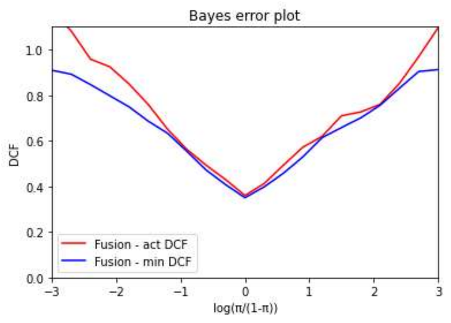
	minimum DCF ($\tilde{\pi}$ = *4/9*)	actual DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	actual DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)	actual DCF ($\tilde{\pi}$ = *4/5*)
Fusion (MVG, GMM), with PCA (m=9))	0.399	0.415	0.664	0.728	0.639	0.699



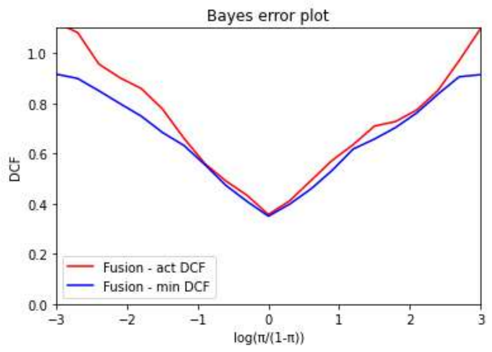
	minimum DCF ($\tilde{\pi}$ = *4/9*)	actual DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	actual DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)	actual DCF ($\tilde{\pi}$ = *4/5*)
Fusion (LR, GMM), no PCA)	0.393	0.421	0.664	0.713	0.633	0.673



	minimum DCF ($\tilde{\pi}$ = *4/9*)	actual DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	actual DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)	actual DCF ($\tilde{\pi}$ = *4/5*)
Fusion (LR, GMM), with PCA (m=10))	0.394	0.419	0.664	0.716	0.640	0.670



	minimum DCF ($\tilde{\pi}$ = *4/9)*	actual DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	actual DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*	actual DCF ($\tilde{\pi}$ = *4/5)*
Fusion (LR, GMM), with PCA (m=9))	0.393	0.415	0.662	0.718	0.643	0.674



Among the three models chosen (with and without PCA) and among the various fusion models considered, the one that seems to perform slightly better than the others is **the fusion of Tied MVG, LR (*λ=1e-4*, *π_T=1/5)* and Tied GMM (2 components), with no PCA**, because, for $\tilde{\pi}$ = *4/9,* it obtains a minimum DCF that it is not the best (0.394), but it is pretty close to that value (0.385), for $\tilde{\pi}$ =*1/5,* it obtains the best minimum DCF among the considered (0.651), and, finally, for $\tilde{\pi}$ = *4/5*, it obtains a minimum DCF equal to 0.577 that is pretty close to the best obtained (0.556).

The scores of the final model we have chosen are among well-calibrated.

Experimental results

From this point, the goal is to assess the quality of our model on held-out data (the evaluation set, contained in “Test.txt”).

We will verify the performance and analyse the choices we made, to see how they affected performance for unseen data.

We again will evaluate systems in terms of minimum DCFs, i.e., as mentioned, the cost we would have if we were able to select the optimal threshold for the evaluation set; this allows verifying whether the proposed solution is indeed the one that can achieve the best accuracy.

Since we have used a k-fold (5-fold) cross validation approach, we will train the model over the 100% of the training data (i.e., the whole dataset contained in “Train.txt”, that we have used so far).

We will start again from the Gaussian classifiers.

Multivariate Gaussian Classifier (MVG)

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
MVG, no PCA	0.375	0.545	0.506
MVG, with PCA (m=10)	0.376	0.586	0.516
MVG, with PCA (m=9)	0.381	0.572	0.534
MVG, with PCA (m=8)	0.372	0.596	0.595
MVG, with PCA (m=7)	0.385	0.615	0.628

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
MVG, “Gaussianization”, no PCA	0.365	0.558	0.580
MVG, “Gaussianization”, with PCA (m=10)	0.365	0.613	0.535
MVG, “Gaussianization”, with PCA (m=9)	0.363	0.613	0.577

MVG, “Gaussianization”, with PCA (m=8)	0.396	0.631	0.588
MVG, “Gaussianization”, with PCA (m=7)	0.396	0.626	0.601

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

In general, the performances seem to be better than those obtained with the k-fold cross validation of the training set.

Unlike the performances obtained on the validation set, for the evaluation set with the “Gaussianization” pre-processing step we have obtained, overall, similar performances to those obtained without using “Gaussianization” .

Naïve Bayes Gaussian Classifier

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Naïve Bayes, no PCA	0.440	0.640	0.766
Naïve Bayes, with PCA (m=10)	0.396	0.607	0.710
Naïve Bayes, with PCA (m=9)	0.395	0.623	0.727
Naïve Bayes, with PCA (m=8)	0.398	0.657	0.727
Naïve Bayes, with PCA (m=7)	0.401	0.664	0.765

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Naïve Bayes, “Gaussianization”, no PCA	0.427	0.632	0.850
Naïve Bayes, “Gaussianization”, with PCA (m=10)	0.394	0.610	0.650
Naïve Bayes, “Gaussianization”, with PCA (m=9)	0.394	0.625	0.662
Naïve Bayes, “Gaussianization”, with PCA (m=8)	0.409	0.671	0.683

Naïve Bayes, “Gaussianization”, with PCA (m=7)	0.413	0.693	0.716

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

In general, the performances seem to be better than those obtained with the k-fold cross validation of the training set.

Unlike the performances obtained on the validation set, for the evaluation set with the “Gaussianization” pre-processing step we have obtained, overall, similar performances to those obtained without using “Gaussianization” .

Tied Covariance Gaussian Classifier

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Tied, no PCA	0.338	0.545	0.625
Tied, with PCA (m=10)	0.363	0.566	0.627
Tied, with PCA (m=9)	0.360	0.570	0.615
Tied, with PCA (m=8)	0.357	0.575	0.676
Tied, with PCA (m=7)	0.366	0.592	0.704

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Tied, “Gaussianization”, no PCA	0.364	0.559	0.679
Tied, “Gaussianization”, with PCA (m=10)	0.356	0.559	0.672
Tied, “Gaussianization”, with PCA (m=9)	0.368	0.591	0.669
Tied, “Gaussianization”, with PCA (m=8)	0.376	0.630	0.645
Tied, “Gaussianization”, with PCA (m=7)	0.370	0.644	0.676

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

In general, the performances seem to be better than those obtained with the k-fold cross validation of the training set.

Unlike the performances obtained on the validation set, for the evaluation set with the “Gaussianization” pre-processing step we have obtained, overall, similar performances to those obtained without using “Gaussianization”.

Naïve Tied Gaussian Classifier

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*
Naïve Tied, no PCA	0.413	0.634	0.758
Naïve Tied, with PCA (m=10)	0.365	0.578	0.628
Naïve Tied, with PCA (m=9)	0.364	0.579	0.627
Naïve Tied, with PCA (m=8)	0.366	0.588	0.699
Naïve Tied, with PCA (m=7)	0.372	0.590	0.737

	minimum DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*
Naïve Tied, “Gaussianization”, no PCA	0.433	0.669	0.809
Naïve Tied, “Gaussianization”, with PCA (m=10)	0.354	0.545	0.647
Naïve Tied, “Gaussianization”, with PCA (m=9)	0.357	0.578	0.656
Naïve Tied, “Gaussianization”, with PCA (m=8)	0.368	0.633	0.656
Naïve Tied, “Gaussianization”, with PCA (m=7)	0.368	0.638	0.678

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9*, $\tilde{\pi}$ = *1/5*, $\tilde{\pi}$ = *4/5).*

In general, the performances seem to be better than those obtained with the k-fold cross validation of the training set.

Unlike the performances obtained on the validation set, for the evaluation set with the “Gaussianization” pre-processing step we have obtained, overall, similar performances to those obtained without using “Gaussianization”.

Linear Logistic Regression

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
LR (λ=*1e-5*, π_T=*4/9)*, no PCA	0.378	0.579	0.535
LR (λ=*1e-5*, π_T=*4/9)*, with PCA (m=10)	0.379	0.574	0.531
LR (λ=*1e-5*, π_T=*4/9)*, with PCA (m=9)	0.380	0.582	0.536
LR (λ=*1e-5*, π_T=*4/9)*, with PCA (m=8)	0.361	0.590	0.639
LR (λ=*1e-5*, π_T=*4/9)*, with PCA (m=7)	0.375	0.600	0.678
LR (λ=*1e-4*, π_T=*4/9)*, no PCA	0.375	0.578	0.539
LR (λ=*1e-4*, π_T=*4/9)*, with PCA (m=10)	0.375	0.583	0.530
LR (λ=*1e-4*, π_T=*4/9)*, with PCA (m=9)	0.377	0.582	0.538
LR (λ=*1e-4*, π_T=*4/9)*, with PCA (m=8)	0.358	0.590	0.664
LR (λ=*1e-4*, π_T=*4/9)*, with PCA (m=7)	0.375	0.599	0.674
LR (λ=*1e-3*, π_T=*4/9)*, no PCA	0.352	0.577	0.574
LR (λ=*1e-3*, π_T=*4/9)*, with PCA (m=10)	0.353	0.573	0.573
LR (λ=*1e-3*, π_T=*4/9)*, with PCA (m=9)	0.353	0.572	0.572
LR (λ=*1e-3*, π_T=*4/9)*, with PCA (m=8)	0.358	0.577	0.644
LR (λ=*1e-3*, π_T=*4/9)*, with PCA (m=7)	0.368	0.586	0.677
LR (λ=*1e-2*, π_T=*4/9)*, no PCA	0.378	0.588	0.711
LR (λ=*1e-2*, π_T=*4/9)*, with PCA (m=10)	0.376	0.586	0.711
LR (λ=*1e-2*, π_T=*4/9)*, with PCA (m=9)	0.377	0.586	0.709
LR (λ=*1e-2*, π_T=*4/9)*, with PCA (m=8)	0.380	0.587	0.738
LR (λ=*1e-2*, π_T=*4/9)*, with PCA (m=7)	0.384	0.586	0.740

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
LR (λ=*1e-5*, π_T=*1/5)*, no PCA	0.373	0.567	0.577
LR (λ=*1e-5*, π_T=*1/5)*, with PCA (m=10)	0.373	0.565	0.555
LR (λ=*1e-5*, π_T=*1/5)*, with PCA (m=9)	0.374	0.568	0.556
LR (λ=*1e-5*, π_T=*1/5)*, with PCA (m=8)	0.354	0.590	0.686
LR (λ=*1e-5*, π_T=*1/5)*, with PCA (m=7)	0.364	0.598	0.709
LR (λ=*1e-4*, π_T=*1/5)*, no PCA	0.367	0.572	0.567
LR (λ=*1e-4*, π_T=*1/5)*, with PCA (m=10)	0.362	0.568	0.575
LR (λ=*1e-4*, π_T=*1/5)*, with PCA (m=9)	0.364	0.564	0.561
LR (λ=*1e-4*, π_T=*1/5)*, with PCA (m=8)	0.353	0.595	0.701
LR (λ=*1e-4*, π_T=*1/5)*, with PCA (m=7)	0.364	0.595	0.701
LR (λ=*1e-3*, π_T=*1/5)*, no PCA	0.349	0.595	0.701
LR (λ=*1e-3*, π_T=*1/5)*, with PCA (m=10)	0.350	0.559	0.647
LR (λ=*1e-3*, π_T=*1/5)*, with PCA (m=9)	0.349	0.561	0.643
LR (λ=*1e-3*, π_T=*1/5)*, with PCA (m=8)	0.350	0.561	0.685
LR (λ=*1e-3*, π_T=*1/5)*, with PCA (m=7)	0.360	0.575	0.705
LR (λ=*1e-2*, π_T=*1/5)*, no PCA	0.373	0.594	0.740
LR (λ=*1e-2*, π_T=*1/5)*, with PCA (m=10)	0.375	0.594	0.738
LR (λ=*1e-2*, π_T=*1/5)*, with PCA (m=9)	0.375	0.594	0.737
LR (λ=*1e-2*, π_T=*1/5)*, with PCA (m=8)	0.377	0.594	0.739
LR (λ=*1e-2*, π_T=*1/5)*, with PCA (m=7)	0.377	0.597	0.743

In the table above, it is possible to notice that for the evaluation set, the best performances are obtained for (λ=*1e-3*, π_T=*1/5);* instead, for the validation set, we have seen that the best result are for (λ=*1e-4*, π_T=*1/5).* However, for the evaluation set the performances for (λ=*1e-4*, π_T=*1/5)* do not differ so much from the results for (λ=*1e-3*, π_T=*1/5).*

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
LR (λ=*1e-5*, π_T=*4/5)*, no PCA	0.382	0.603	0.524
LR (λ=*1e-5*, π_T=*4/5)*, with PCA (m=10)	0.380	0.603	0.527
LR (λ=*1e-5*, π_T=*4/5)*, with PCA (m=9)	0.387	0.598	0.532
LR (λ=*1e-5*, π_T=*4/5)*, with PCA (m=8)	0.375	0.602	0.580
LR (λ=*1e-5*, π_T=*4/5)*, with PCA (m=7)	0.393	0.605	0.615
LR (λ=*1e-4*, π_T=*4/5)*, no PCA	0.381	0.591	0.517
LR (λ=*1e-4*, π_T=*4/5)*, with PCA (m=10)	0.379	0.590	0.519
LR (λ=*1e-4*, π_T=*4/5)*, with PCA (m=9)	0.379	0.591	0.519
LR (λ=*1e-4*, π_T=*4/5)*, with PCA (m=8)	0.376	0.606	0.584
LR (λ=*1e-4*, π_T=*4/5)*, with PCA (m=7)	0.390	0.604	0.615
LR (λ=*1e-3*, π_T=*4/5)*, no PCA	0.361	0.597	0.553
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=10)	0.363	0.597	0.550
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=9)	0.363	0.597	0.550
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=8)	0.377	0.608	0.588
LR (λ=*1e-3*, π_T=*4/5)*, with PCA (m=7)	0.383	0.600	0.615
LR (λ=*1e-2*, π_T=*4/5)*, no PCA	0.394	0.608	0.675
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=10)	0.394	0.605	0.676
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=9)	0.394	0.608	0.674
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=8)	0.394	0.605	0.701
LR (λ=*1e-2*, π_T=*4/5)*, with PCA (m=7)	0.400	0.606	0.705

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9*, $\tilde{\pi}$ = *1/5*, $\tilde{\pi}$ = *4/5)*.

In general, the performances seem to be better than those obtained with the k-fold cross validation of the training set.

Linear Support Vector Machines

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
SVM (C= 0.001, k=1), no PCA	0.520	0.768	0.972
SVM (C= 0.001, k=1), with PCA (m=10)	0.520	0.773	0.972
SVM (C= 0.001, k=1), with PCA (m=9)	0.522	0.578	0.972
SVM (C= 0.001, k=1), with PCA (m=8)	0.551	0.818	0.967
SVM (C= 0.001, k=1), with PCA (m=7)	0.421	0.365	0.839
SVM (C= 0.01, k=1), no PCA	0.421	0.635	0.839
SVM (C= 0.01, k=1), with PCA (m=10)	0.421	0.636	0.832
SVM (C= 0.01, k=1), with PCA (m=9)	0.422	0.636	0.832
SVM (C= 0.01, k=1), with PCA (m=8)	0.427	0.637	0.849
SVM (C= 0.01, k=1), with PCA (m=7)	0.441	0.649	0.840
SVM (C= 0.1, k=1), no PCA	0.395	0.588	0.711
SVM (C= 0.1, k=1), with PCA (m=10)	0.399	0.594	0.717
SVM (C= 0.1, k=1), with PCA (m=9)	0.427	0.621	0.802
SVM (C= 0.1, k=1), with PCA (m=8)	0.403	0.611	0.730
SVM (C= 1, k=1), with PCA (m=7)	0.413	0.617	0.734
SVM (C= 1, k=1), no PCA	0.355	0.569	0.594
SVM (C= 1, k=1), with PCA (m=10)	0.443	0.614	0.667
SVM (C= 1, k=1), with PCA (m=9)	0.384	0.578	0.648
SVM (C= 1, k=1), with PCA (m=8)	0.878	0.979	0.972
SVM (C= 1, k=1), with PCA (m=7)	0.400	0.591	0.666
SVM (C= 10, k=1), no PCA	0.994	1	0.998
SVM (C= 10, k=1), with PCA (m=10)	0.969	1	0.952
SVM (C= 10, k=1), with PCA (m=9)	1	1	0.977

SVM (C= 10, k=1), with PCA (m=8)	0.810	0.971	0.983
SVM (C= 10, k=1), with PCA (m=7)	0.839	1	0.945

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for *$\tilde{\pi}$ = 4/9, $\tilde{\pi}$ = 1/5, $\tilde{\pi}$ = 4/5).*

In general, the performances seem to be better than those obtained with the k-fold cross validation of the training set; in particular, for C=1 and k=1, with no PCA, the performances are comparable to those of Tied MVG and Linear LR, something that did not happen for the validation set.

Polynomial Kernel Support Vector Machines

We report in the table below the obtained computational results:

	minimum DCF (*$\tilde{\pi}$ = 4/9*)	minimum DCF (*$\tilde{\pi}$ = 1/5*)	minimum DCF (*$\tilde{\pi}$ = 4/5*)
Poly (C= 0.001, d=2, c=1, k=1), no PCA	0.958	0.996	0.930
Poly (C= 0.001, d=2, k=1), with PCA (m=10)	0.941	0.985	0.971
Poly (C= 0.001, d=2, k=1), with PCA (m=9)	0.900	0.985	0960
Poly (C= 0.001, d=2, k=1), with PCA (m=8)	0.765	0.897	0.922
Poly (C= 0.001, d=2, c=1, k=1), with PCA (m=7)	0.754	1	0.901
Poly (C= 0.01, d=2, c=1, k=1), no PCA	0.987	1	0.998
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=10)	0.904	1	0.970
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=9)	1	1	0.973
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=8)	0.989	1	0.971
Poly (C= 0.01, d=2, c=1, k=1), with PCA (m=7)	0.943	0.989	0.974
Poly (C= 0.1, d=2, c=1, k=1), no PCA	0.917	1	0.977
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=10)	1	1	0.947
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=9)	0.674	0.849	0.977
Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=8)	0.587	0.988	0.936

Poly (C= 0.1, d=2, c=1, k=1), with PCA (m=7)	0.980	1	0.960
Poly (C= 1, d=2, c=1, k=1), no PCA	0.646	1	0.811
Poly (C= 1, d=2, c=1, k=1), with PCA (m=10)	0.893	0.984	0.952
Poly (C= 1, d=2, c=1, k=1), with PCA (m=9)	1	1	0.997
Poly (C= 1, d=2, c=1, k=1), with PCA (m=8)	0.991	1	0.986
Poly (C= 1, d=2, c=1, k=1), with PCA (m=7)	0.933	1	0.994
Poly (C= 10, d=2, c=1, k=1), no PCA	0.797	1	0.886
Poly (C= 10, d=2, c=1, c=1, k=1), with PCA (m=10)	0.987	1	0.964
Poly (C= 10, d=2, c=1, k=1), with PCA (m=9)	0.962	0.999	0.959
Poly (C= 10, d=2, c=1, k=1), with PCA (m=8)	0.985	0.998	0.960
Poly (C= 10, d=2, c=1, k=1), with PCA (m=7)	1	1	0.966

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

Gaussian Radial Basis Function Kernel Support Vector Machines

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
RBF (C= 0.001, $\gamma=e^{-3}$, k=1), no PCA	0.663	0.876	0.949
RBF (C= 0.01, $\gamma=e^{-3}$, k=1), no PCA	0.592	0.826	0.875
RBF (C= 0.1, $\gamma=e^{-3}$, k=1), no PCA	0.595	0.825	0.891
RBF (C= 1, $\gamma=e^{-3}$, k=1), no PCA	0.575	0.773	0.921
RBF (C= 10, $\gamma=e^{-3}$, k=1), no PCA	0.583	0.828	0.941
RBF (C= 0.001, $\gamma=e^{-2}$, k=1), no PCA	0.637	0.880	0.933

RBF (C= 0.01, $\gamma=e^{-2}$, k=1), no PCA	0.616	0.881	0.917
RBF (C= 0.1, $\gamma=e^{-2}$, k=1), no PCA	0.614	0.888	0.925
RBF (C= 1, $\gamma=e^{-2}$, k=1), no PCA	0.620	0.847	0.923
RBF (C= 10, $\gamma=e^{-2}$, k=1), no PCA	0.638	0.881	0.945
RBF (C= 0.001, $\gamma=e^{-1}$, k=1), no PCA	0.637	0.888	0.900
RBF (C= 0.01, $\gamma=e^{-1}$, k=1), no PCA	0.638	0.899	0.901
RBF (C= 0.1, $\gamma=e^{-1}$, k=1), no PCA	0.649	0.916	0.903
RBF (C= 1, $\gamma=e^{-1}$, k=1), no PCA	0.642	0.901	0.906
RBF (C= 10, $\gamma=e^{-1}$, k=1), no PCA	0.638	0.906	0.910
RBF (C= 0.001, $\gamma=1$, k=1), no PCA	0.657	0.927	0.925
RBF (C= 0.01, $\gamma=1$, k=1), no PCA	0.650	0.934	0.938
RBF (C= 0.1, $\gamma=1$, k=1), no PCA	0.656	0.942	0.936
RBF (C= 1, $\gamma=1$, k=1), no PCA	0.656	0.946	0.936
RBF (C= 10, $\gamma=1$, k=1), no PCA	0.660	0.949	0.936
RBF (C= 0.001, $\gamma=e$, k=1), no PCA	0.719	0.948	0.958
RBF (C= 0.01, $\gamma=e$, k=1), no PCA	0.718	0.952	0.954
RBF (C= 0.1, $\gamma=e$, k=1), no PCA	0.719	0.960	0.954
RBF (C= 1, $\gamma=e$, k=1), no PCA	0.720	0.969	0.953
RBF (C= 10, $\gamma=e$, k=1), no PCA	0.720	0.970	0.953
RBF (C= 0.001, $\gamma=e^{2}$, k=1), no PC	0.896	0.974	0.957
RBF (C= 0.01, $\gamma=e^{2}$, k=1), no PCA	0.882	0.974	0.961
RBF (C= 0.1, $\gamma=e^{2}$, k=1), no PCA	0.884	0.974	0.961
RBF (C= 1, $\gamma=e^{2}$, k=1), no PCA	0.890	0.974	0.961
RBF (C= 10, $\gamma=e^{2}$, k=1), no PCA	0.887	0.974	0.961

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for *$\tilde{\pi}$ = 4/9, $\tilde{\pi}$ = 1/5, $\tilde{\pi}$ = 4/5).*

Gaussian Mixture Model

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = *4/9)*	minimum DCF ($\tilde{\pi}$ = *1/5)*	minimum DCF ($\tilde{\pi}$ = *4/5)*
GMM, 2 components, no PCA	0.428	0.642	0.767
GMM, 2 components, with PCA (m=10)	0.428	0.642	0.767
GMM, 2 components, with PCA (m=9)	0.428	0.643	0.769
GMM, 2 components, with PCA (m=8)	0.649	1	0.786
GMM, 2 components, with PCA (m=7)	0.717	1	0.840
GMM, 4 components, no PCA	0.573	0.733	0.966
GMM, 4 components, with PCA (m=10)	0.573	0.733	0.966
GMM, 4 components, with PCA (m=9)	0.581	0.746	0.966
GMM, 4 components, with PCA (m=8)	0.590	0.777	0.970
GMM, 4 components, with PCA (m=7)	0.581	0.740	0.977
GMM, 8 components, no PCA	0.840	0.949	0.976
GMM, 8 components, with PCA (m=10)	0.840	0.949	0.976
GMM, 8components, with PCA (m=9)	0.836	0.943	0.978
GMM, 8 components, with PCA (m=8)	0.852	0.961	0.970
GMM, 8 components, with PCA (m=7)	0.886	0.975	0.964

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9,* $\tilde{\pi}$ = *1/5,* $\tilde{\pi}$ = *4/5).*

In general, the performances are very better than those obtained with the k-fold cross validation of the training set.

Diagonal Covariance Gaussian Mixture Model

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = *4/9*)	minimum DCF ($\tilde{\pi}$ = *1/5*)	minimum DCF ($\tilde{\pi}$ = *4/5*)
Diag GMM, 2 components, no PCA	0.577	1	0.739
Diag GMM, 2 components, with PCA (m=10)	0.581	0.782	0.964
Diag GMM, 2 components, with PCA (m=9)	0.582	0.782	0.964
Diag GMM, 2 components, with PCA (m=8)	0.581	0.790	0.975
Diag GMM, 2 components, with PCA (m=7)	0.581	0.790	0.965
Diag GMM, 4 components, no PCA	0.847	1	0.991
Diag GMM, 4 components, with PCA (m=10)	0.728	0.865	0.976
Diag GMM, 4 components, with PCA (m=9)	0.727	0.862	0.976
Diag GMM, 4 components, with PCA (m=8)	0.728	0.873	0.980
Diag GMM, 4 components, with PCA (m=7)	0.704	0.860	0.979
Diag GMM, 8 components, no PCA	0.994	1	0.997
Diag GMM, 8 components, with PCA (m=10)	0.941	0.999	0.967
Diag GMM, 8 components, with PCA (m=9)	0.943	0.999	0.967
Diag GMM, 8 components, with PCA (m=8)	0.941	1	0.952
Diag GMM, 8 components, with PCA (m=7)	0.783	0.944	0.973

The best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi}$ = *4/9*, $\tilde{\pi}$ = *1/5*, $\tilde{\pi}$ = *4/5).*

In general, the performances are very better than those obtained with the k-fold cross validation of the training set.

Tied Covariance Gaussian Mixture Model

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)
Tied GMM, 2 components, no PCA	0.390	0.617	0.656
Tied GMM, 2 components, with PCA (m=10)	0.390	0.617	0.656
Tied GMM, 2 components, with PCA (m=9)	0.391	0.617	0.650
Tied GMM, 2 components, with PCA (m=8)	0.413	0.633	0.652
Tied GMM, 2 components, with PCA (m=7)	0.437	0.651	0.709
Tied GMM, 4 components, no PCA	0.410	0.640	0.685
Tied GMM, 4 components, with PCA (m=10)	0.410	0.640	0.685
Tied GMM, 4 components, with PCA (m=9)	0.411	0.640	0.691
Tied GMM, 4 components, with PCA (m=8)	0.407	0.642	0.747
Tied GMM, 4 components, with PCA (m=7)	0.437	0.661	0.790
Tied GMM, 8 components, no PCA	0.491	0.698	0.917
Tied GMM, 8 components, with PCA (m=10)	0.491	0.698	0.917
Tied GMM, 8 components, with PCA (m=9)	0.490	0.701	0.918
Tied GMM, 8 components, with PCA (m=8)	0.504	0.719	0.930
Tied GMM, 8 components, with PCA (m=7)	0.501	0.683	0.926

For each table, the best minimum DCF for each $\tilde{\pi}$ is highlighted by a different colour (respectively, red, green and blue for $\tilde{\pi} = 4/9$, $\tilde{\pi} = 1/5$, $\tilde{\pi} = 4/5$).

In general, the performances are similar to those obtained with the k-fold cross validation of the training set.

For the validation set, the performances were comparable to those of Tied MVG and LR, now, for the evaluation set, the performances of the other two models are better than those of Tied GMM.

Fusion

We will consider again the fusion between Tied MVG, LR (λ=*1e-4*, π_T=*1/5)* and Tied GMM (2 components).

We report in the table below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Fusion (3 models, no PCA)	**0.354**	**0.561**	**0.540**
Fusion (3 models, with PCA (m=10))	0.351	0.576	0.521
Fusion (3 models, with PCA (m=9))	0.352	0.588	0.532

We will compute again also the minimum DCF for the fusion of the pairs obtained from the three classifiers considered.

We report in the tables below the obtained computational results:

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Fusion ((MVG, LR), no PCA)	0.347	0.565	0.582
Fusion ((MVG, LR), with PCA (m=10))	0.360	0.571	0.592
Fusion ((MVG, LR), with PCA(m=9))	0.356	0.568	0.584

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Fusion ((MVG, GMM), no PCA)	0.369	0.568	0.541
Fusion ((MVG, GMM), with PCA (m=10))	0.364	0.581	0.511
Fusion ((MVG, GMM), with PCA(m=9))	0.366	0.589	0.514

	minimum DCF ($\tilde{\pi} = 4/9$)	minimum DCF ($\tilde{\pi} = 1/5$)	minimum DCF ($\tilde{\pi} = 4/5$)
Fusion ((LR, GMM), no PCA)	0.369	0.590	0.516
Fusion ((LR, GMM), with PCA (m=10))	0.370	0.588	0.514
Fusion ((LR, GMM), with PCA(m=9))	0.368	0.591	0.511

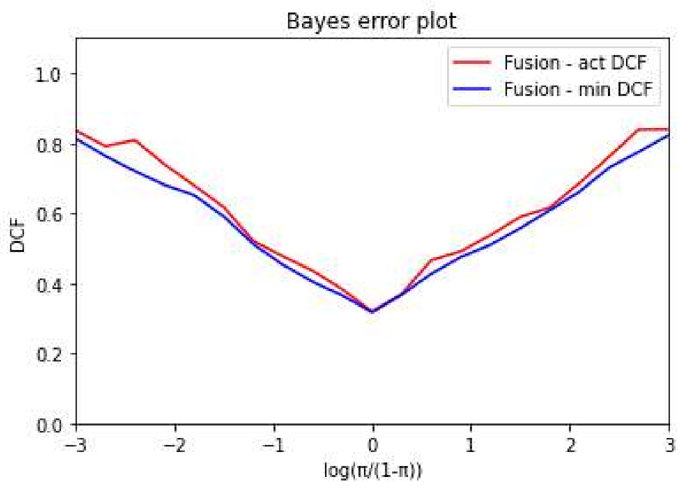
Actual Detection Cost Function

Up to now, we have analysed the systems on the evaluation set in terms of minimum DCFs, that, as mentioned, measure the potential capabilities of the systems to produce good decisions.

However, in practice we must make decisions without knowing what the optimal rule for the evaluation data in our case is, we need to map our scores to class labels, but we do not know the optimal score threshold. We want therefore to assess how good are the decisions that we are actually able to make using the recognizer scores. To evaluate the decision-making capabilities of our systems, we consider actual DCFs, which represent the (normalized) Bayes cost we would pay for our actual decisions. Actual DCFs are computed using the actual decisions (i.e. the actual class predictions) we make for evaluation data.

In the table below, we will show the minimum and the actual DCF for the three models used for the computation of the fusions (Tied MVG, LR (λ=*1e-4*, π_T=*1/5)* and Tied GMM (2 components)) and the fusion chosen as our final model (Tied MVG, LR (λ=*1e-4*, π_T=*1/5)* and Tied GMM (2 components), with no PCA).

	minimum DCF ($\tilde{\pi}$ = 4/9)	actual DCF ($\tilde{\pi}$ = 4/9)	minimum DCF ($\tilde{\pi}$ = 1/5)	actual DCF ($\tilde{\pi}$ = 1/5)	minimum DCF ($\tilde{\pi}$ = 4/5)	actual DCF ($\tilde{\pi}$ = 4/5)
Tied, no PCA	0.338	0.365	0.545	0.553	0.625	0.673
LR (λ=*1e-4*, π_T=*1/5)*, no PCA	0.367	0.374	0.572	0.581	0.567	0.568
Tied GMM, 2 components, no PCA	0.390	0.438	0.617	0.698	0.656	0.721
Fusion (3 models, no PCA)	**0.354**	**0.362**	**0.561**	**0.584**	0.540	**0.544**



Finally, our final model, consisting of the fusion of three subsystems, is effective, and produces well-calibrated scores for a wide range of applications.

It is not the best model ever for the evaluation set, because we have seen in the various tables above that there are models that had performed slightly better; however, we can say that is one of the best models among the considered and its performances are pretty good and especially well calibrated.

Overall, apart from some unexpected happenings, like the different behaviour applying the "Gaussianization" in the validation and in the evaluation set and the improved performances for the linear SVM applied with certain hyperparameters and the slight deterioration for the Tied GMM with 2 components, we can affirm that the validation (and, naturally, the training set, given that for the k-fold cross validation the two set fit together) and the evaluation population are quite similar, but their behaviour is not totally the same.

Bibliography

Many of the formulas and concepts in this report have been extrapolated from notes and slides of the course of *"Machine Learning and Pattern Recognition"* of the teacher Sandro Cumani.