**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**

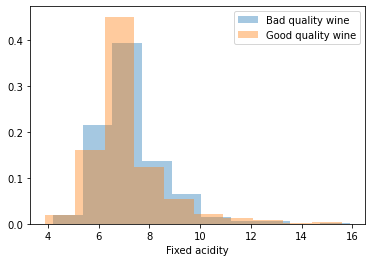
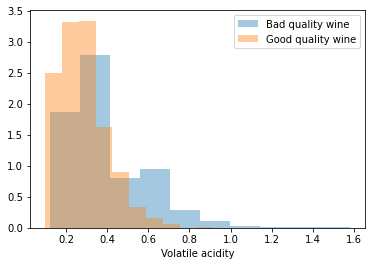
The dataset is taken from the UCI repository. The original dataset consists of 10 classes (quality 1 to 10). For this project, the dataset has been binarized, collecting all wines with low quality (score lower than 6) into class 0, and good quality (score greater than 6) into class 1. Wines with quality 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:

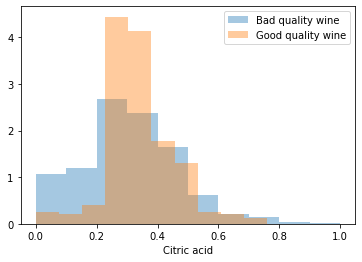
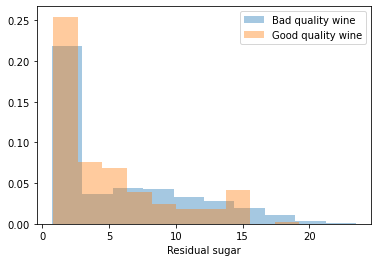
1. Fixed acidity
2. Volatile acidity
3. Citric acid
4. Residual sugar
5. Chlorides
6. Free sulfur dioxide
7. Total sulfur dioxide
8. Density
9. pH
10. Sulphates
11. Alcohol

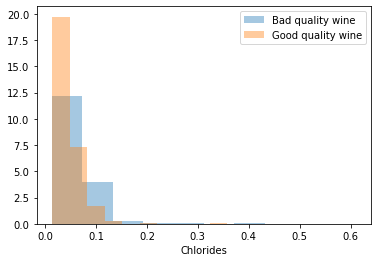
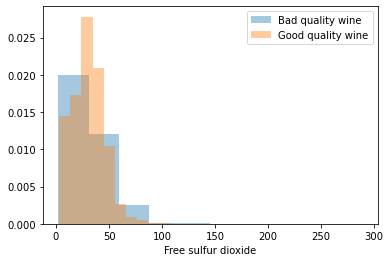
In the training set, there are 1839 samples, with their own features and another field with the class to which they belong (0 or 1). In particular, in this set, 1226 samples belong to the bad quality wine class and 613 to the good quality wine class. We modified the initial ‘Train.txt’ file to extract 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.

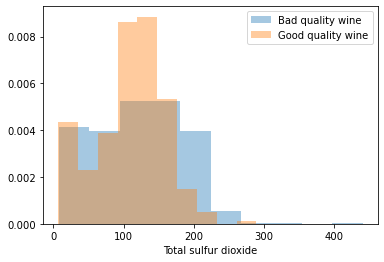
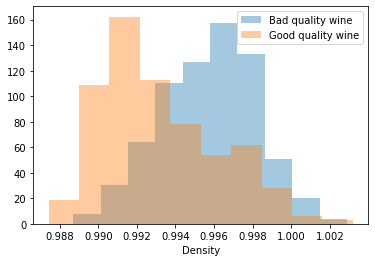
From this point, we will consider only the data of the training set. Later, when we will start the evaluation of the models, the use of the test set will be specified.

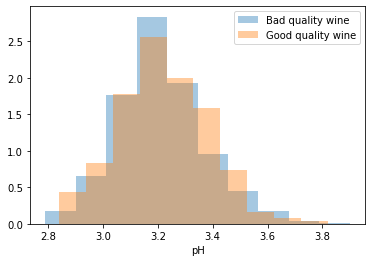
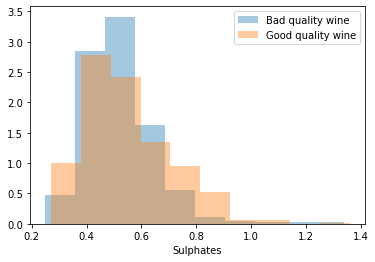
To understand how the several features of the two classes are distributed, for each of them, it is possible to plot the corresponding histogram:

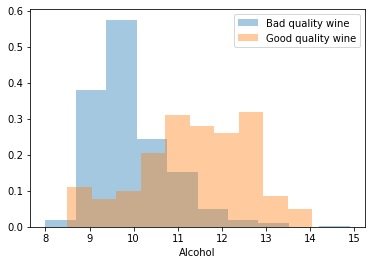
 

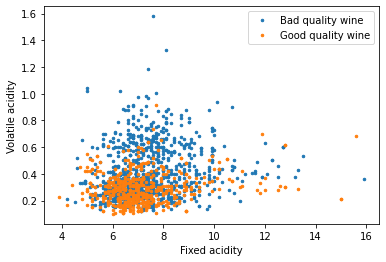
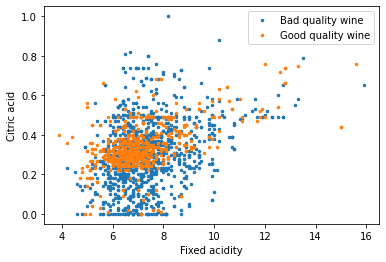
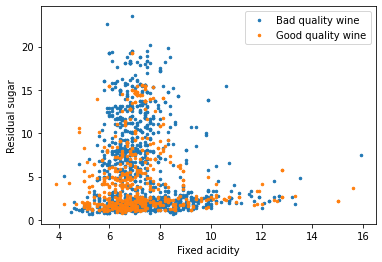
 

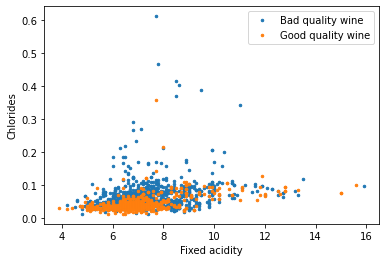
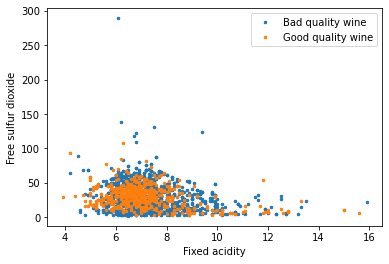
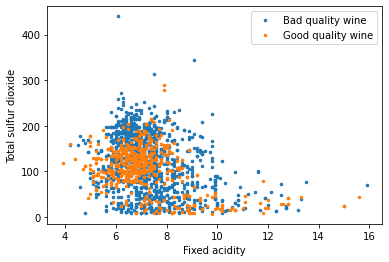
 

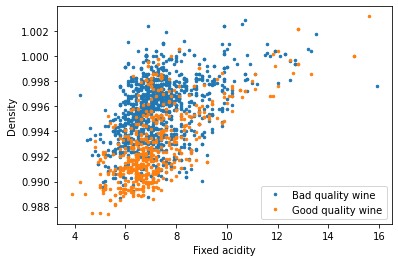
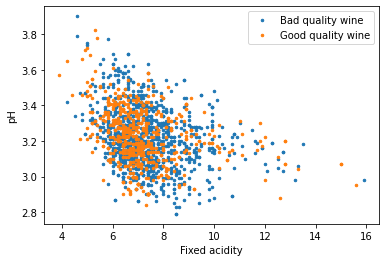
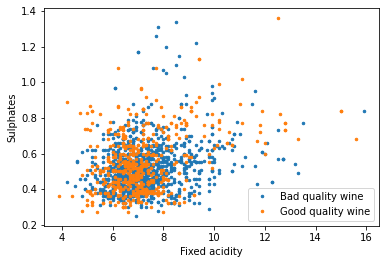


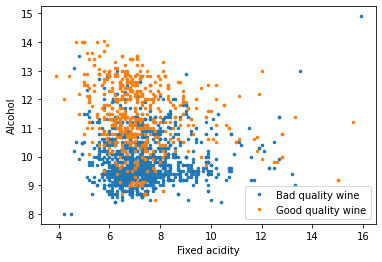
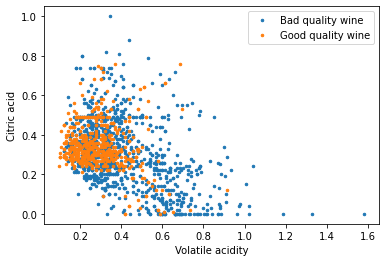
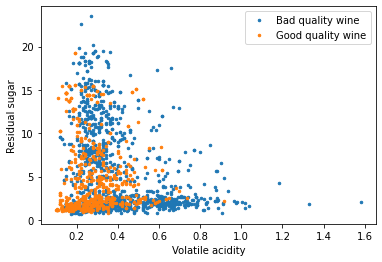
We can observe that, in general, for all features, there is no very well separation, vice versa, these is a large overlap and, sometimes, as in the case of *fixed acidity, residual sugar, pH* and *sulphates* the data is almost totally overlapping. For *density* and *alcohol*, it is possible to notice 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*

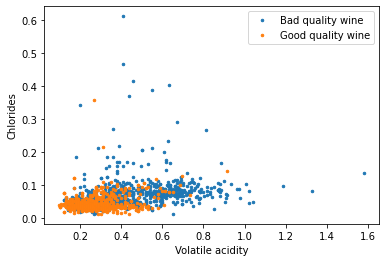
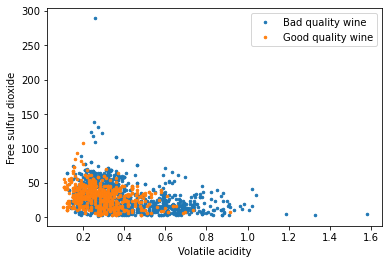
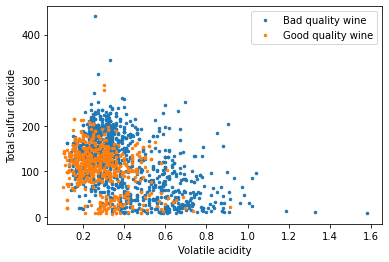
To have a view of the relationship between the different features for the different classes, we can visualize the scatter plots of the different features pairs for each class:

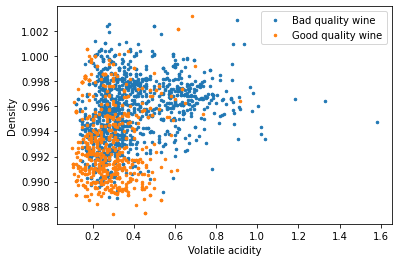
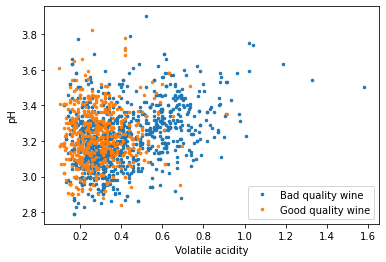
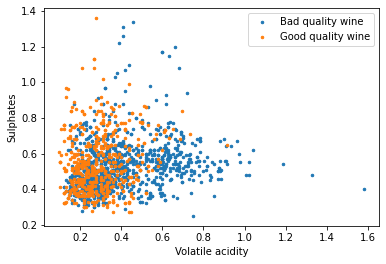
  

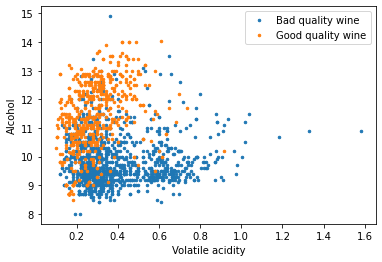
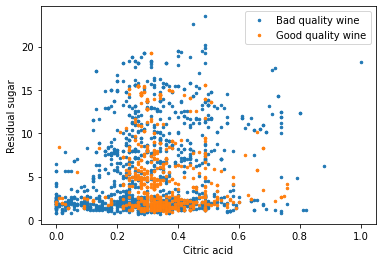
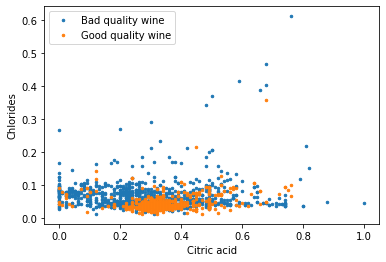
  

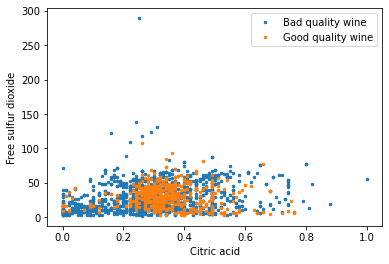
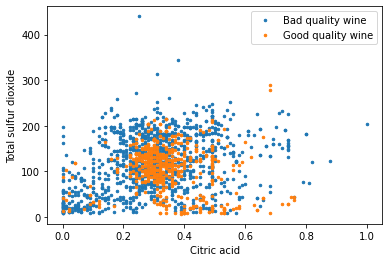
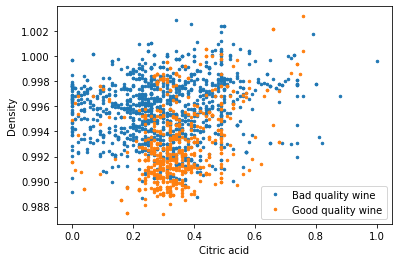
  

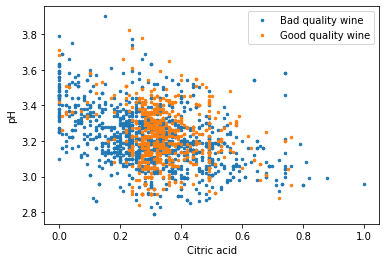
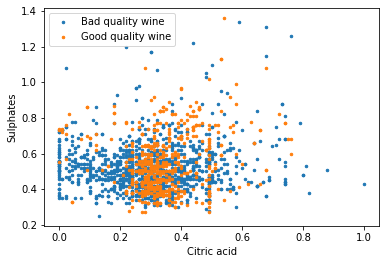
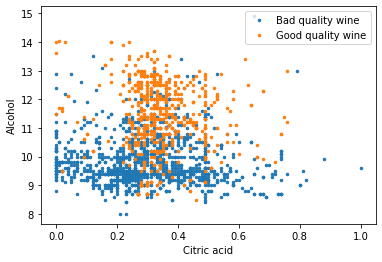
  

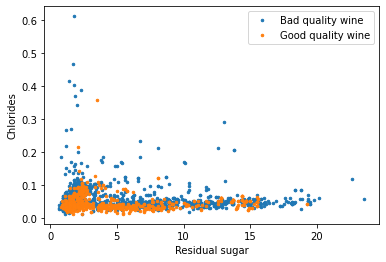
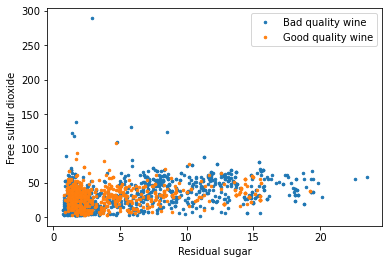
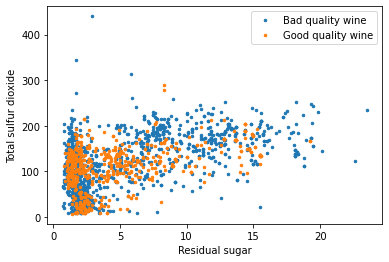
  

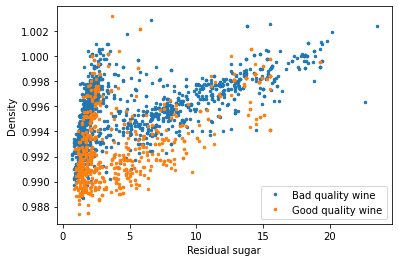
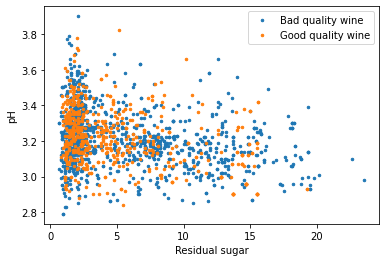
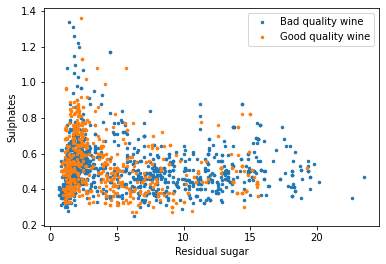
  

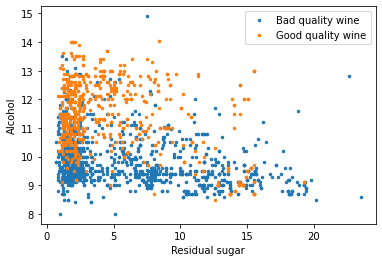
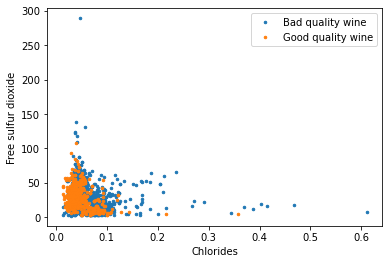
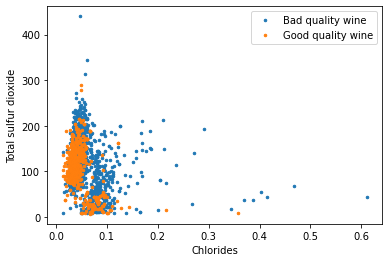
  

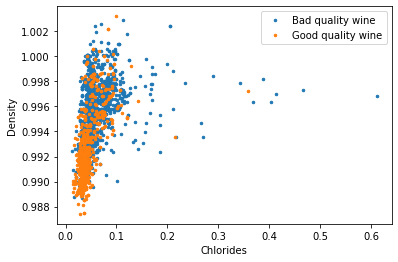
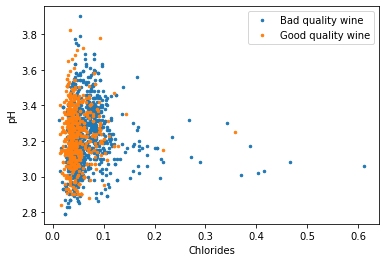
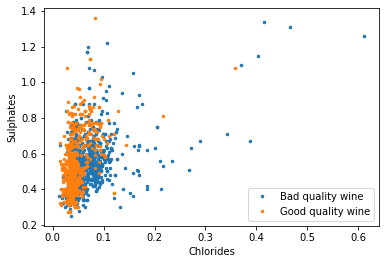
  

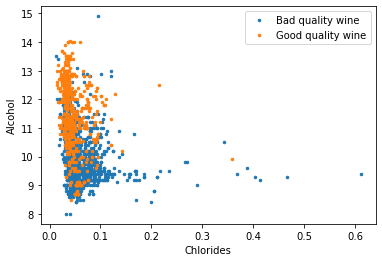
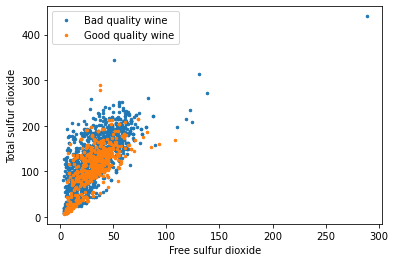
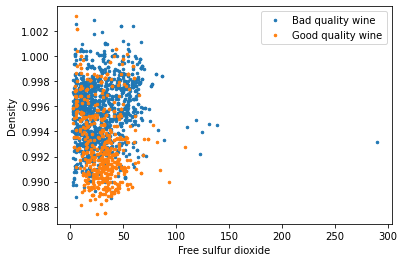
  

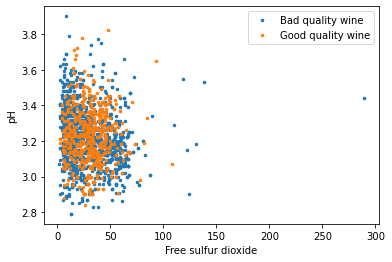
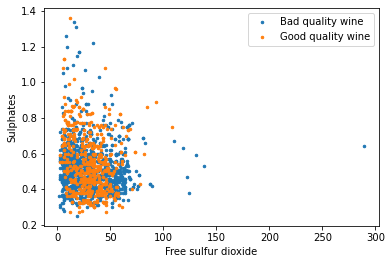
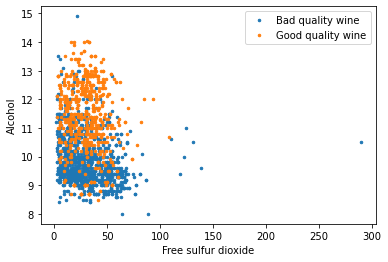
  

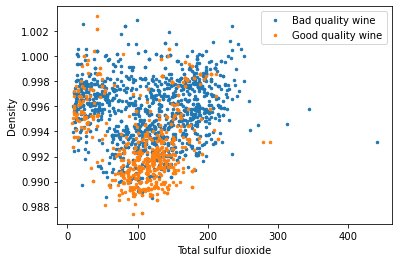
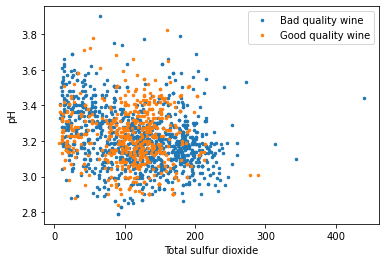
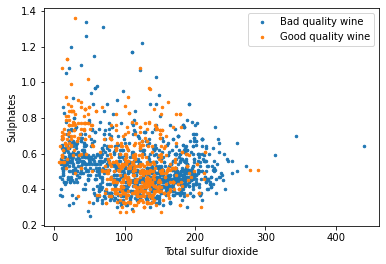
  

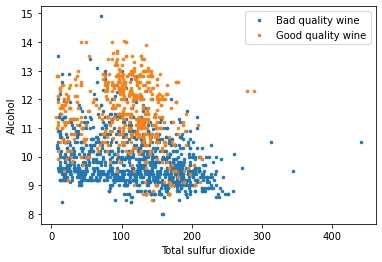
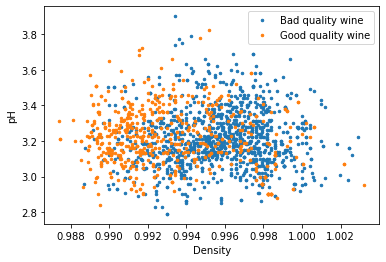
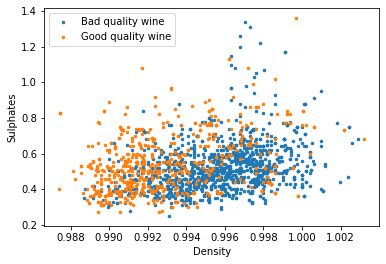
  

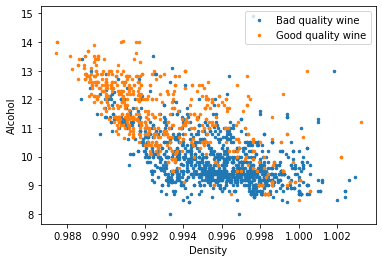
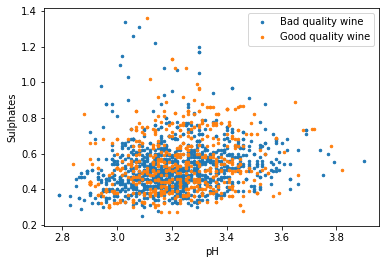
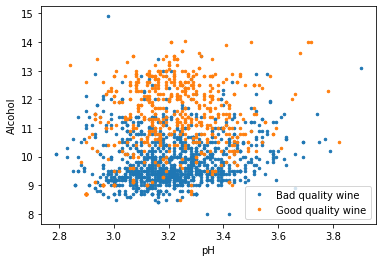
  

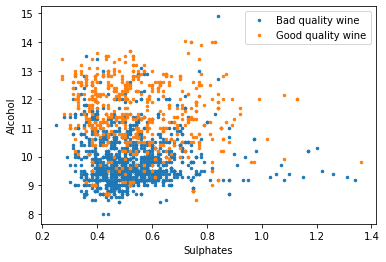
  



**Dimensionality Reduction**

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 ℝn . *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 =*

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

*C* can be decomposed as:

*C= UΣUT*

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 = [u1 … um, um+1 … un]*

*P = [u1 … um]*

*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 = PTX*

Through experimental tests, we can affirm that, for this problem, the best results are obtained for m in the range [7, 9].

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  *= 4/9*, as the prior probability that a wine belongs to the good quality wine class and *1– = 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 *Cfp=1* and *Cfn=1*, so our application will be: *(, Cfp, Cfn)= (4/9, 1, 1).*

We will also consider other two unbalanced applications: *(, Cfp, Cfn)= (1/5, 1, 1)*, *(, Cfp, Cfn)= (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.

To compute the minimum cost, we will consider a set of thresholds corresponding to (−∞, s1 . . . sn, +∞), where s1 . . . sn 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 using the code developed in the previous section. The minimum DCF is the minimum of the obtained values.

Min DCF measures the cost we would pay if we made optimal decisions for our case the validation set using the recognizer scores.

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)

**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.

where *N* is a normal distribution.

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

where *xc,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 ratio between the likelihood of observing the sample given that it belongs to class h1 or to class h0 (we denote the classes sometimes with *h* and sometimes with *c*):

We report in the table below the obtained computational results:

|  |  |  |  |
| --- | --- | --- | --- |
|  | minimum DCF  *= 4/9)* | minimum DCF  ( *= 1/5)* | minimum DCF  *= 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 |

**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:

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 table below the obtained computational results:

|  |  |  |  |
| --- | --- | --- | --- |
|  | minimum DCF  *= 4/9)* | minimum DCF  *= 1/5)* | minimum DCF  *= 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 |

**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:

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 table below the obtained computational results:

|  |  |  |  |
| --- | --- | --- | --- |
|  | minimum DCF  *= 4/9)* | minimum DCF  *= 1/5)* | minimum DCF  *= 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 |

**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:

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 table below the obtained computational results:

|  |  |  |  |
| --- | --- | --- | --- |
|  | minimum DCF  *= 4/9)* | minimum DCF  *= 1/5)* | minimum DCF  *= 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 |

**Logistic Regression**

The Logistic regression is a discriminative approach for classification.

The regularized Logistic Regression objective can be written as follows:

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

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

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:

We report in the table below the obtained computational results:

|  |  |  |  |
| --- | --- | --- | --- |
|  | minimum DCF  *= 4/9)* | minimum DCF  *= 1/5)* | minimum DCF  *= 4/5)* |
| LR (*λ=1e-5*, *πT=4/9)*, no PCA | 0.418 | 0.724 | 0.596 |
| LR (*λ=1e-5*, *πT=4/9)*, with PCA (m=10) | 0.418 | 0.723 | 0.600 |
| LR (*λ=1e-5*, *πT=4/9)*, with PCA (m=9) | 0.417 | 0.731 | 0.588 |
| LR (*λ=1e-5*, *πT=4/9)*, with PCA (m=8) | 0.421 | 0.685 | 0.678 |
| LR (*λ=1e-5*, *πT=4/9)*, with PCA (m=7) | 0.437 | 0.682 | 0.702 |
| LR (*λ=1e-4*, *πT=4/9)*, no PCA | 0.410 | 0.719 | 0.607 |
| LR (*λ=1e-4*, *πT=4/9)*, with PCA (m=10) | 0.411 | 0.716 | 0.600 |
| LR (*λ=1e-4*, *πT=4/9)*, with PCA (m=9) | 0.412 | 0.724 | 0.604 |
| LR (*λ=1e-4*, *πT=4/9)*, with PCA (m=8) | 0.416 | 0.682 | 0.684 |
| LR (*λ=1e-4*, *πT=4/9)*, with PCA (m=7) | 0.435 | 0.682 | 0.701 |
| LR (*λ=1e-3*, *πT=4/9)*, no PCA | 0.407 | 0.697 | 0.666 |
| LR (*λ=1e-3*, *πT=4/9)*, with PCA (m=10) | 0.406 | 0.697 | 0.665 |
| LR (*λ=1e-3*, *πT=4/9)*, with PCA (m=9) | 0.406 | 0.697 | 0.665 |
| LR (*λ=1e-3*, *πT=4/9)*, with PCA (m=8) | 0.420 | 0.695 | 0.727 |
| LR (*λ=1e-3*, *πT=4/9)*, with PCA (m=7) | 0.436 | 0.690 | 0.743 |
| LR (*λ=1e-2*, *πT=4/9)*, no PCA | 0.465 | 0.697 | 0.780 |
| LR (*λ=1e-2*, *πT=4/9)*, with PCA (m=10) | 0.465 | 0.700 | 0.780 |
| LR (*λ=1e-2*, *πT=4/9)*, with PCA (m=9) | 0.465 | 0.697 | 0.779 |
| LR (*λ=1e-2*, *πT=4/9)*, with PCA (m=8) | 0.471 | 0.698 | 0.797 |
| LR (*λ=1e-2*, *πT=4/9)*, with PCA (m=7) | 0.475 | 0.682 | 0.805 |
| LR (*λ=1e-1 πT=4/9)*, no PCA | 0.497 | 0.719 | 0.905 |
| LR (*λ=1e-1*, *πT=4/9)*, with PCA (m=10) | 0.497 | 0.719 | 0.906 |
| LR (*λ=1e-1*, *πT=4/9)*, with PCA (m=9) | 0.497 | 0.719 | 0.906 |
| LR (*λ=1e-1*, *πT=4/9)*, with PCA (m=8) | 0.500 | 0.724 | 0.909 |
| LR (*λ=1e-1*, *πT=4/9)*, with PCA (m=7) | 0.501 | 0.724 | 0.912 |
| LR (*λ=1, πT=4/9)*, no PCA | 0.651 | 0.949 | 0.993 |
| LR (*λ=1*, *πT=4/9)*, with PCA (m=10) | 0.650 | 0.949 | 0.994 |
| LR (*λ=1*, *πT=4/9)*, with PCA (m=9) | 0.651 | 0.949 | 0.993 |
| LR (*λ=1*, *πT=4/9)*, with PCA (m=8) | 0.652 | 0.949 | 0.993 |
| LR (*λ=1, πT=4/9)*, with PCA (m=7) | 0.652 | 0.949 | 0.993 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | minimum DCF  *= 4/9)* | minimum DCF  *= 1/5)* | minimum DCF  *= 4/5)* |
| LR (*λ=1e-3*, *πT=4/9)*,  with PCA (m=10) | 0.406 | 0.697 | 0.665 |
| LR (*λ=1e-3*, *πT=4/9)*, with PCA (m=9) | 0.406 | 0.697 | 0.665 |
| LR (*λ=1e-3*, *πT=1/5)*, with PCA (m=10) | 0.405 | 0.688 | 0.718 |
| LR (*λ=1e-3*, *πT=1/5)*, with PCA (m=9) | 0.402 | 0.687 | 0.719 |
| LR (*λ=1e-3*, *πT=4/5)*, with PCA (m=10) | 0.430 | 0.698 | 0.604 |
| LR (*λ=1e-3*, *πT=4/5)*, with PCA (m=9) | 0.426 | 0.695 | 0.603 |