# **Wine quality detection report**

##### **D’Ascenzi Giulia s287720, De Girolamo Patrizio s292497**

Report for the “Machine Learning and Pattern Recognition” course.

Academic year 2020-2021

# **Introduction**

T

he dataset used for the purpose of this analysis is related to red and white variants of the Portuguese "Vinho Verde" wine and is taken from the UCI repository.

The dataset was collected to predict human wine taste preferences. As a matter of fact, it associates the physiochemical characteristics of the analyzed wine to sensory evaluations made by experts.

The original dataset consists of 10 classes (quality 1 to 10), but for this project the dataset has been binarized, collecting all wines with low quality (lower than 6) into class 0 and good quality (grater than 6) into class 0. Wines with quality 6 have been discarded to simplify the task.

The dataset contains both red and white wines that were originally separated, whereas in this analysis they have been merged.

## **Classes Balance**

The dataset has been divided into a training set and a test set.The training set contains 613 samples belonging to the "high quality" class and 1226 belonging to the "low quality" class. On the other hand, the evaluation set contains 664 samples of class "high quality" and 1158 samples of class "low quality". Therefore, the classes are partially balanced.

The following graphs show the comparison between the number of samples of the two classes in both training and test data set. It emerges that both the number of samples and the ratio between the two classes in the different datasets is similar.

|  |  |
| --- | --- |
|  |  |

## **Attributes Information**

The input variables have 11 continuous features based on physiochemical tests: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol.

The output variable is a discrete value representing the quality of the wine sample (0 low quality/1 high quality)

## **Preprocessing and feature analysis**

The features of the dataset refer to different types of variables and have therefore different measuring scales.

Thus, in order to compare similarities between features, the dataset has been z-normalized, through centering and scaling to unit variance the features.

The histograms below show the distributions of the training dataset features. Orange histograms refer to high quality wines, whereas blue histograms to low quality wines.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

The analysis of the training data reveals that most of the features have an irregular distribution.

Consequently, the classification (especially of Gaussian based methods) may produce sub-optimal results. We therefore further preprocessed our data by "Gaussianizing" the features.

The gaussianization process allows mapping the features values to ones whose empirical comulative distribution function is well approximated by a Gaussian c.d.f. To do so, the features have been mapped to a uniform distribution and then transformed through the inverse of Gaussian cumulative distribution function.

The histograms below show the distributions of the gaussianized features.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

correlation analysis of the Gaussianized features shows that feature 5 and 6 are strongly correlated.

The heatmaps showing the Pearson correlation coefficient

can be found below.

From right to left: whole dataset, high quality features, low quality features.

|  |  |  |
| --- | --- | --- |
|  |  |  |

This suggests that the classification may benefit from using PCA to map data to 10 or 9 uncorrelated features to reduce the number of parameters to estimate.

# **Classification of wine quality features**

## **Methodologies used for validation**

In order to understand which model is most promising and to assess the effects of using PCA, both single fold-validation and K-Fold cross-validation have been adopted.

At first, a single fold approach has been used, since the training with this approach is faster. Indeed, single fold approach consists in splitting the training dataset into two subsets, one (66% of the original set) for development and the other one for validation. Subsequently, the K-Fold approach has been used to get more robust results. In this case, the training set are iteratively split into 5 folds, 4 used for training and 1 for validation, after being shuffled. The validation scores have been eventually put together and used to compute the performance metrics. In this way, there is more data available for training and validation.

This document focuses on the analysis of the balanced uniform prior application:

(π, ,) = (0.5,1,1)

However, two other unbalanced applications have been considered:

(π, ,) = (0.9,1,1) and (π, ,) = (0.1,1,1)

In the fist part, the main aim of the analysis was to choose the most promising approach. Therefore, the performances have been measured in terms of normalized minimum detection costs, namely the costs that would be paid by making optimal decisions for the validation set through the use of recognizers scores.

## **MVG CLASSIFIERS**

The first type of classifier analysed are the multivariate gaussian classifiers that assume that the data, given the class, can be described by a Gaussian distribution.

The following table show the results in terms of minimum DCF of the gaussian classifiers studied.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Single Fold** | | | **5-Fold** | | |
|  |  |  |  |  |  |  |
| **Raw Features – no PCA** | | | | | | |
| **Full - Cov** | 0.304 | 0.812 | 0.777 | 0.312 | 0.842 | 0.778 |
| **Diag -Cov** | 0.437 | 0.875 | 0.818 | 0.420 | 0.921 | 0.845 |
| **Tied Full-Cov** | 0.334 | 0.733 | 0.779 | 0.333 | 0.748 | 0.812 |
| **Tied Diag- Cov** | 0.412 | 0.901 | 0.832 | 0.402 | 0.932 | 0.866 |
| **Gaussianized Features – no PCA** | | | | | | |
| **Full - Cov** | 0.270 | 0.740 | 0.807 | 0.306 | 0.790 | 0.784 |
| **Diag -Cov** | 0.456 | 0.848 | 0.860 | 0.448 | 0.914 | 0.834 |
| **Tied Full-Cov** | 0.348 | 0.812 | 0.867 | 0.354 | 0.884 | 0.803 |
| **Tied Diag- Cov** | 0.451 | 0.891 | 0.839 | 0.451 | 0.942 | 0.879 |
| **Gaussianized Features – PCA (m=10)** | | | | | | |
| **Full - Cov** | 0.312 | 0.817 | 0.779 | 0.328 | 0.862 | 0.807 |
| **Diag -Cov** | 0.362 | 0.747 | 0.780 | 0.378 | 0.824 | 0.804 |
| **Tied Full-Cov** | 0.324 | 0.682 | 0.791 | 0.328 | 0.753 | 0.809 |
| **Tied Diag- Cov** | 0.335 | 0.656 | 0.822 | 0.334 | 0.782 | 0.822 |
| **Gaussianized Features – PCA (m=9)** | | | | | | |
| **Full - Cov** | 0.300 | 0.817 | 0.783 | 0.319 | 0.814 | 0.800 |
| **Diag -Cov** | 0.349 | 0.764 | 0.810 | 0.368 | 0.814 | 0.804 |
| **Tied Full-Cov** | 0.331 | 0.677 | 0.786 | 0.327 | 0.752 | 0.816 |
| **Tied Diag- Cov** | 0.337 | 0.673 | 0.822 | 0.335 | 0.783 | 0.825 |

The diagonal covariance models (both full diagonal and tied diagonal) do not give good results if compared to the other models with or without Gaussianization and PCA. These models work under the naive-bayes-assumption, according to which the different components for each class are uncorrelated. Therefore, in this case, this assumption does not produce accurate results. Applying PCA slightly improves the performance, probabily because by removing the low variances directions the whithin-class correlation decreases

The tied full covariance model performs pretty good on the validation data, confirming the similiarity between classes shown by the correlation analysis. It performs better than the diagonal models accounting the within-class correlations. In this case, PCA does not help in producing better results.

The best-performing model is the full covariance model, which is able to account for correlations. What is more, having enough data compared to the dimensionality of the samples, the results we succeed obtaining robust results.

Gaussianization fails in improving significantly the results, although it still performs slightly better if compared to raw features.

Even if it does not help increasing much the performances, PCA can still be used to reduce the number of parameters and

therefore to reduce the complexity of the model. Nevertheless, given the limited effectiveness, it will not be considered for the future analysis.

The results between single-fold and K-fold are consistent, suggesting that the amount of data is enough for validation and model training also when it comes to the single-fold set up.

None of the models produce accurate results for the unbalanced applications.

In summary, the best candidate is currently the MVG model with full covariance matrices. The chosen one is the one with Gaussianized features, since it shows slightly better results than the one with raw features, and the K-fold approach, even if it is a bit worse than the single fold result. Nonetheless, this approach provides more robust results.

Since the best-performing models are the full covariance and the tied full models (which respectively have a quadratic surface rule and a linear surface rule), the decision was to proceed by analyzing both the quadratic and the linear models.

## **LOGISTIC REGRESSION**

Up to now, the focus of the analysis was only on generative models, but now turns on the discriminative model. The first one is the linear logistic regression.

### **Linear Logistic Regression**

Linear logistic regression is a discriminative model and therefore it models the class posterior distribution, rather than the distribution of the observed samples X|C

To do so, logistic regression looks for the linear hyperplanes that maximizes the likelihood of the training labels. This also corresponds both on the minimization of the average cross entropy between the empirical distribution of the data and the predicted labels distribution and the minimization of the empirical risk of mis-classify the data.

Since the classes of this dataset are not perfectly balanced and since the logistic regression model automatically embeds the prior empirical probability, the original model has been modified to take in account the different weights of the loss' costs due to the samples of each class in order to reflect the prior of our applications.

The first step was the tuning of the hyperparameter λ. The following graphs show the how the minDCF for the three chosen application varies to respect to λ. (Left: single fold. Right: K-fold. Top: raw features. Bottom: gaussianized features)

|  |  |
| --- | --- |
|  |  |
|  |  |

Results with K-fold and single fold are similar. For this reason the following analysis have been done using the K-fold approach, since the validation results are more reliable.

Regularization did not provide much benefit. Best results are obtained with values of λ less than . In order to avoid over confident models, that could be obtain with a λ value too low, but at the same time to get an optimal minDCF, the value chosen for λ was

Also in this case the gaussianization does not improve much the performance of the model. Indeed, logistic regression does not require assumptions on the data distribution.

In the following table are also reported the result got using different prior to see the effects on the other applications.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Raw Features** | | | |
| LR (λ= , ) | 0.352 | 0.833 | 0.680 |
| LR (λ= , ) | 0.336 | **0.818** | 0.733 |
| LR (λ= , ) | 0.368 | 0.852 | **0.653** |
| LR (λ= ) | 0.340 | 0.839 | 0.673 |
| **Gaussianized features** | | | |
| LR (λ= , ) | 0.360 | 0.858 | 0.764 |
| LR (λ= , ) | 0.340 | **0.780** | 0.933 |
| LR (λ= , ) | 0.376 | 0.898 | **0.698** |
| LR (λ= ) | 0.359 | 0.829 | 0.836 |

Overall, the MVG model with full covariances perform better.

Furthermore, comparing this results with the best linear MVG classifier (Tied covariance model) the logistic regression performed slightly worse.

Since the full covariances MVG corresponds to quadratic separation rules, the analysis has been repeated with a quadratic logistic regression model.

Class re-balancing helps for the unbalanced applications, whereas for the main application the best results are got with unbalanced classes meaning that the class re-balancing was not necessary in this case.

### **Quadratic Logistic Regression**

The following graphs show the minDCF for different values of LAMBDA.

(Top: raw features. Bottom: gaussianized features. Left: single fold. Right: K-fold)

|  |  |
| --- | --- |
|  |  |
|  |  |

Also in this case, varying lambda does not produce improvement for lambda low than . Therefore the regularized terms has been set to for the same reason of before.

In the following table are also reported the result got using different prior to see the effects on the other applications.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Raw Features** | | | |
| Quad LR (λ= , ) | 0.275 | 0.771 | 0.692 |
| Quad LR (λ= , ) | 0.275 | **0.752** | 0.703 |
| Quad LR (λ= , ) | 0.287 | 0.807 | **0.642** |
| Quad LR (λ= ) | **0.274** | 0.769 | 0.686 |
| **Gaussianized features** | | | |
| Quad LR (λ= , ) | 0.291 | 0.676 | 0.653 |
| Quad LR (λ= , ) | 0.292 | 0.700 | 0.644 |
| Quad LR (λ= , ) | 0.307 | 0.747 | **0.625** |
| Quad LR (λ= ) | 0.288 | 0.676 | 0.637 |

The results of the quadratic logistic regression outperforms the MVG classifier results.

Also in this case the gaussianization does not improve the results with respect to the raw features results. Although, even if the results of the unbalanced applications are still pretty poor, the quadratic logistic regression with gaussianized feature is so far the model that gives the best results for those two applications.

For all the applications the re-balancing does not improve the performances.

## **SVM**

The analysis continues with the Support Vector Machine model.

The SVM classifier is a discriminative classifier that aims to find a separation hyperplane between two classes that have the maximum margin, so the maximum distance between the hyperplane and the closest points. In case the classes are not linearly separable, the approach followed is the "soft margin" approach, that looks for the highest margin hyperplane trying to minimize the points that lie inside the margin or that are on the wrong side of the classification rule (therefore misclassified points).

Again, It has been considered a re-balanced version of the model to reflect the effective priors of the chosen applications.

Subjected to:

It has been selected and , where and are the empirical priors (i.e sample proportions) forthe two classes computed over training set.

### **Linear SVM**

The linear svm model has as hyperparameter C that is used to define a trade off between having a solution with a wide margin and a solution with a low number of points laying inside the margin.

Higher values for C corresponds to solutions with less permitted points inside the margin and a thin margin. These solutions have an higher risk of being overconfident and to not perform well on unseen data. On the other hand, a lower value of C means allowing to have more points inside the margin, a wider margin that provides a solution less confident, that generalizes better.

The following graphs show how minDCF varies for different values of the hyperparameter C.

Right: single fold. Left: 5 fold.

First row: gaussianized features, class balancing (pi\_T= 0.5).

Second row: raw features, class balancing.

Third row: raw features, class without balancing.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

For the hyperparameter C the choice followed the same strategy used before: it has been set to since it is the biggest value that assure good performances.

The graphs already show re-balancing the model does not result in much higher performances for the target application of this analysis.

Also the gaussianized preprocessing do not improve the performance, so it has not been considered for the following analysis.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Raw Features** | | | |
| SVM (C=0.1 , ) | 0.339 | 0.849 | 0.668 |
| SVM (C=0.1, ) | 0.892 | 0.995 | 0.996 |
| SVM (C=0.1 , ) | 0.386 | 0.876 | 0.693 |
| SVM (C=0.1) | 0.338 | 0.817 | 0.761 |

Since the linear SVM is the last linear classifier analysed, these results prove once again that the linear classifiers perform worst than quadratic classifiers in this dataset.

|  |  |
| --- | --- |
| **Comparison between the best linear models** () | **minDCF** |
| MVG Tied Full-Cov | *0.334* |
| SVM, C=0.1 | 0.338 |
| Log reg, λ= | 0.340 |

All these three classifiers have similar results. Also, it can be noticed that the assumption of gaussian distribution made by the tied full model does not worsen the performance compared to models as SVM and LR that do not make any assumption on the data distributions.

Also, noticing that the preprocessing step of gaussianizing the features is most of the time useless, the conclusion can be that the distribution of this dataset can be sufficiently approximated to a Gaussian distribution even if the best results so far are still gotten with quadratic logistic regression that does not need assumptions of this type.

Therefore the analysis proceeds with the quadratic version of the SVM.

### **SVM with quadratic kernel**

The first non linear SVM model used is the polynomial quadratic kernel, that is similar to the quadratic Logistic Regression model and therefore we expect similar results.

The chose model has degree d = 2 (quadratic), whereas c is an hyper-parameter selected through cross-validation.

To add a regularized bias to the non- linear SVM version, has been added a constant value K to the kernel function.

In order to estimate the values of c and K that achieve the best results, different models have been trained varying those parameters. In the following graphs are shown the minDCF got from the four best models found through this analysis, varying the value of the hyper-parameter C.

Up: Raw features. Bottom: gaussianized features. Left: single fold. Right: 5 fold.

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Raw Features** | | | |
| RBF SVM ( = 1, C=1, ) | **0.218** | 0.536 | 0.567 |
| RBF SVM ( = 1, C=1,) | 0.348 | **0.534** | 0.690 |
| RBF SVM ( = 1, C=1, ) | 0.285 | 0.595 | **0.567** |
| RBF SVM ( = 1, C=1) | 0.225 | 0.548 | 0.606 |
| **Gaussianized features** | | | |
| RBF SVM ( = 1, C=1, ) | 0.233 | 0.511 | 0.573 |
| RBF SVM ( = 1, C=1,) | 0.338 | 0.522 | 0.613 |
| RBF SVM ( = 1, C=1, ) | 0.269 | 0.566 | 0.578 |
| RBF SVM ( = 1, C=1) | 0.230 | 0.498 | 0.570 |

The best results are got with hyper-parameter K=0, c=1 and C=0.1.

Again, the K-fold results are consistent with the single fold results.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| **Raw Features** | | | |
| Quad SVM (K=0, c=1, C=0.1, ) | **0.273** | 0.798 | 0.691 |
| Quad SVM (K=0, c=1, C=0.1,) | 0.311 | **0.760** | 0.866 |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.325 | 0.959 | **0.663** |
| Quad SVM (K=0, c=1, C=0.1) | 0.277 | 0.774 | 0.723 |
| **Gaussianized features** | | | |
| Quad SVM (K=0, c=1, C=0.1, ) | **0.282** | 0.730 | 0.627 |
| Quad SVM (K=0, c=1, C=0.1,) | 0.322 | 0.706 | 0.942 |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.308 | 0.853 | 0.613 |
| Quad SVM (K=0, c=1, C=0.1) | 0.296 | 0.674 | 0.718 |

These results confirms that quadratic surfaces are better to discriminate the classes of our dataset.

The best results are got with the balanced version of the model. Gaussianizing the features does not help improving the performances.

Now the attenction is turned to RBF kernel.

### **SVM with RBF kernel**

The Gaussian Radial Basis Function in another kernel function that can be used for non- linear SVM models.

The kernel function is

Where is the width of the kernel: small correspond to a wide kernel, while large to a narrow kernel.

The estimation of the kernel width has been done through a grid search in order to optimize C and .

In the graphs below are plot the minDCFs of models trained with different and different C.

Left: raw features. Right: gaussianized features.

|  |  |
| --- | --- |
|  |  |

The plot shows that both and C influence the results. To optimize jointly both hyperpararmeters, the values chosen are = 1 and C=1.

In the table are shown the results got from the model trained with those hyper-parameters with and without re-balancing of the classes costs.

The RBF results significantly outperforms our previous models for both the target application and the unbalanced ones.

Class re-balancing helps increasing the performances.

Therefore, for the target application the best model found so far is the class-balanced RBF SVM trained with raw features.

Comparing the quadratic models in term of minDCF on the target application:

|  |  |
| --- | --- |
| **Comparison between the bet quadratic models (** | **minDCF** |
| MVG Full - Cov | 0.304 |
| Quad LR (λ= ) | 0.274 |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.273 |
| RBF SVM ( = 1, C=1, ) | 0.218 |

The RBF is clearly the one that best performs on the target application.

The last model considered is the GMM.

## **GAUSSIAN MIXTURE MODEL**

The Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

It allows to approximate any sufficiently regular distribution.

It can be used as a classification model assuming that each class can be approximated by a GMM with a given number of sub-components.

Since this model benefits on having a large amount of training data, the approach used for the following analysis was the K-fold approach.

Also, different type of GMM have been tried (GMM with full covariances, GMM with diagonal covariances, GMM with tied covariances and GMM with tied diagonal covariances)

The graphs below show the performances in terms of minDCF obtained with raw features and gaussianized features varying the number of components of each GMM.

|  |  |
| --- | --- |
|  |  |
|  |  |

It can be seen that the model trained with gaussianized features have overall similar performances of the one trained with raw features.

Also, the best results are got with GMM with full covariances and GMM with tied covariances.

In the table below are reported all the minDCF got in the analysis.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number of components:** | **2** | **4** | **8** | **8** | **16** | **32** |
| **Raw Features** | | | | | | |
| Full Cov | 0.312 | 0.300 | 0.305 | **0.286** | 0.292 | 0.302 |
| Diag Full | 0.420 | 0.384 | 0.344 | 0.337 | 0.335 | 0.319 |
| Tied Full | 0.312 | 0.312 | 0.305 | 0.310 | 0.309 | **0.295** |
| Tied Diag | 0.420 | 0.404 | 0.362 | 0.328 | 0.314 | 0.324 |
| **Gaussianized features** | | | | | | |
| Full Cov | 0.306 | 0.327 | 0.303 | 0.309 | 0.292 | 0.336 |
| Diag Full | 0.448 | 0.358 | 0.327 | 0.318 | 0.305 | 0.309 |
| Tied Full | 0.306 | 0.306 | 0.309 | 0.307 | 0.296 | **0.296** |
| Tied Diag | 0.448 | 0.407 | 0.327 | 0.327 | 0.323 | 0.307 |

The results got for the one-component model are consistent with the ones of MVG models.

The best non-tied model is Full Covariance with 8 components trained with raw features. Instead, some degrees of overfitting can be seen in this model increasing more the number of components.

Also, in this model, the gaussianization shows slightly worst performances. This can be explained since gaussianization reduce the dynamic range of samples far from the data mean and thus reduces the separability of some clusters.

The best tied model is the Tied full model trained with raw features but also the one trained with gaussianized features that has a similar performance.

From this first analysis of the studied classifiers, the two selected models are **SVM with RBF kernel (parameters: = 1, C=1, )** trained with **raw features** and the **SVM with quadratic kernel (parameters C=0.1, ))** trained with **raw features**.

## **Analysis in terms of actual DCF**

Up to now the analysis focused only on minimum DCF metrics.

Min DCF measures the cost that would be paid making optimal decisions for the validation set using the recognizer scores.

The actually paid cost, however, depends on the goodness of the decisions made using those scores that in the binary case corresponds on the goodness of the threshold used in practice to perform class assignment.

Therefore, now the analysis turns the attention to actual DCFs

If the scores are well calibrated, the optimal threshold that optimizes the Bayes risk is

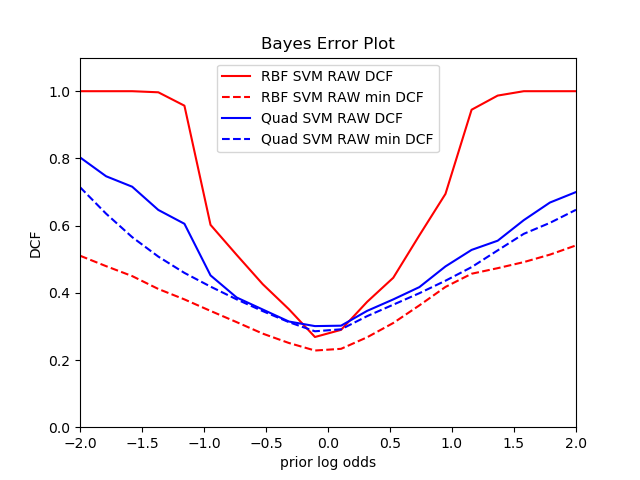
The following table compares the values got as minDCF and actDCF for the three different applications using the two selected models.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | |  | |  | |
|  | **minDCF** | **actDCF** | **minDCF** | **actDCF** | **minDCF** | **actDCF** | |
| **RBF SVM** | 0.218 | 0.218 | 0.536 | 1.0 | 0.567 | 1.0 | |
| **Quad SVM** | 0.273 | 0.296 | 0.798 | 0.852 | 0.691 | 0.749 | |

These results show that both models are quite calibrated for the target application of this report. In particular, with the RBF model the minDCF corresponds exactly to the actual DCF, despite both models lack a probabilistic interpretation.

On the other hand, for the unbalanced applications the scores are overall not well calibrated especially using the RBF classifier.

For a better visualization, these results have been plotted in a Bayes Error Plot, which shows the DCFs for different applications.



It is clearly visible that the RBF outputs (quite-) calibrated scores only for the target application (), wheras changing prior log odds (and therefore the application) the curve of the minDCFs and the actualDCF depart from each other.

Instead the quadratic SVM is well calibrated in a wider range of applications.

Therefore for the target application for both classifiers, using the theoretical optimal threshold would give good results. But, since this would not apply for the unbalanced applications, the solution followed was estimating a close-to-optimal threshold. The results are shown in the table below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **minDCF** | **actDCF** | **DCF with estimated t\*** |
|  | | | |
| **RBF SVM** | 0.223 | 0.226 | 0.229 |
| **Quad SVM** | 0.289 | 0.315 | 0.314 |
|  | | | |
| **RBF SVM** | 0.551 | **1.0** | **0.601** |
| **Quad SVM** | 0.788 | **0.859** | **0.898** |
|  | | | |
| **RBF SVM** | 0.635 | **1.0** | **0.662** |
| **Quad SVM** | 0.697 | **0.752** | **0.731** |

These procedure looks quite accurate for the two unbalanced application since the results got with the estimated threshold are surely closer to the minDCF compared to the DCF computed with the theoretical threshold.

Instead, the target application, being already quite calibrated does not get a great improvement from this procedure.

Since this analysis focuses only on the target application, the chosen model is the **SVM with RBF kernel** since it gives the best validation results and it also produces well calibrated scores.

# **EXPERIMENTAL RESULTS**

The last step of this report assesses the quality of the chosen model on held-out data (the evaluation set).

Also, the performances got are verified for unseen data.

For the experimental results the choice was to verify only the ones on the target application.

Again, first the performances have been evaluate using the minimum DCF and then using the actual DCF.

## **MVG Classifiers**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **66% Data** | **100% data** | |
| **Raw Features – no PCA** | | |
| **Full - Cov** | **0.329** | **0.344** | |
| **Diag -Cov** | 0.379 | 0.473 | |
| **Tied Full-Cov** | **0.316** | **0.320** | |
| **Tied Diag- Cov** | 0.370 | 0.369 | |
| **Gaussianized Features – no PCA** | | |
| **Full - Cov** | 0.325 | 0.326 | |
| **Diag -Cov** | 0.390 | 0.384 | |
| **Tied Full-Cov** | 0.329 | 0.325 | |
| **Tied Diag- Cov** | 0.396 | 0.387 | |
| **Gaussianized Features – PCA (m=10)** | | |
| **Full - Cov** | 0.334 | 0.330 | |
| **Diag -Cov** | 0.332 | 0.331 | |
| **Tied Full-Cov** | 0.323 | 0.326 | |
| **Tied Diag- Cov** | 0.325 | 0.327 | |
| **Gaussianized Features – PCA (m=9)** | | |
| **Full - Cov** | 0.314 | 0.324 | |
| **Diag -Cov** | 0.332 | 0.336 | |
| **Tied Full-Cov** | 0.322 | 0.325 | |
| **Tied Diag- Cov** | 0.325 | 0.324 | |

## **LOGISTIC REGRESSION**

### **Linear Logistic Regression**

|  |  |
| --- | --- |
| **Raw Features** | |
| LR (λ= , ) | 0.333 |
| LR (λ= , ) | 0.317 |
| LR (λ= , ) | 0.350 |
| LR (λ= ) | 0.340 |
| **Gaussianized features** | |
| LR (λ= , ) | 0.342 |
| LR (λ= , ) | 0.310 |
| LR (λ= , ) | 0.363 |
| LR (λ= ) | 0.359 |

### **Quadratic Logistic Regression**

|  |  |
| --- | --- |
| **Raw Features** | |
| Quad LR (λ= , ) | 0.253 |
| Quad LR (λ= , ) | 0.265 |
| Quad LR (λ= , ) | 0.258 |
| Quad LR (λ= ) | 0.254 |
| **Gaussianized features** | |
| Quad LR (λ= , ) | 0.299 |
| Quad LR (λ= , ) | 0.293 |
| Quad LR (λ= , ) | 0.299 |
| Quad LR (λ= ) | 0.288 |

## **SVM**

### **Linear SVM**

|  |  |
| --- | --- |
| **Raw Features** | |
| SVM (C=0.1 , ) | 0.334 |
| SVM (C=0.1, ) | 0.643 |
| SVM (C=0.1 , ) | 0.357 |
| SVM (C=0.1) | 0.312 |

### **SVM with quadratic kernel**

|  |  |
| --- | --- |
| **Raw Features** | |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.270 |
| Quad SVM (K=0, c=1, C=0.1,) | 0.305 |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.283 |
| Quad SVM (K=0, c=1, C=0.1) | 0.274 |
| **Gaussianized features** | |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.307 |
| Quad SVM (K=0, c=1, C=0.1,) | 0.300 |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.298 |
| Quad SVM (K=0, c=1, C=0.1) | 0.283 |

### **SVM with RBF kernel**

|  |  |
| --- | --- |
| **Raw Features** | |
| RBF SVM ( = 1, C=1, ) | 0.309 |
| RBF SVM ( = 1, C=1,) | 0.446 |
| RBF SVM ( = 1, C=1, ) | 0.298 |
| RBF SVM ( = 1, C=1) | 0.308 |
| **Gaussianized features** | |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.297 |
| Quad SVM (K=0, c=1, C=0.1,) | 0.425 |
| Quad SVM (K=0, c=1, C=0.1, ) | 0.286 |
| Quad SVM (K=0, c=1, C=0.1) | 0.320 |

## **GAUSSIAN MIXTURE MODELS**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Number of components:** | **2** | **4** | **8** | **8** | **16** | **32** |
| **Raw Features** | | | | | | |
| Full Cov | 0.344 | 0.305 | 0.311 | **0.298** | 0.321 | 0.326 |
| Diag Full | 0.420 | 0.384 | 0.344 | 0.337 | 0.335 | 0.319 |
| Tied Full | 0.312 | 0.312 | 0.305 | 0.310 | 0.309 | **0.295** |
| Tied Diag | 0.420 | 0.404 | 0.362 | 0.328 | 0.314 | 0.324 |
| **Gaussianized features** | | | | | | |
| Full Cov | 0.306 | 0.327 | 0.303 | 0.309 | 0.292 | 0.336 |
| Diag Full | 0.448 | 0.358 | 0.327 | 0.318 | 0.305 | 0.309 |
| Tied Full | 0.306 | 0.306 | 0.309 | 0.307 | 0.296 | **0.296** |
| Tied Diag | 0.448 | 0.407 | 0.327 | 0.327 | 0.323 | 0.307 |