Immagine che contiene testo, Elementi grafici, Carattere, design

Descrizione generata automaticamente

Gender Identification Project

# Master of Science in Computer Engineering

# Machine learning and pattern recognition exam

# 2022/2023

Gelsi Alessandro

S303525

# Abstract

This report is dedicated to the analysis of a dataset consisting of low-level images depicting males and females, employing a range made of different Machine Learning (ML) algorithms. The goal is to determine the models that achieve the highest classification performance. Initially it was conducted an examination of the dataset's features, followed by the exploration of various classifiers, including Multivariate Gaussian Models (MVG), Logistic Regression (both linear and quadratic), Support Vector Machine (linear, RBF, and quadratic), Gaussian Mixture Models, and Fusion. A validation dataset is derived from the training dataset using K-fold cross-validation, in order to find the best hyperparameters for each model. The performances are evaluated at first taking into account minimum detection cost function(min DCF), followed then also by considerations of actual DCF and score calibration. Finally, the models equipped with the chosen hyperparameters were tested on evaluation set, made of unseen data.

It will be demonstrated that all classifiers yield good results on the provided dataset. As a best model it was selected SCRIVERE QUA IL BEST MODEL

# Index

[Master of Science in Computer Engineering 1](#_Toc144297980)

[Machine learning and pattern recognition exam 1](#_Toc144297981)

[2022/2023 1](#_Toc144297982)

[Abstract 2](#_Toc144297983)

[Index 3](#_Toc144297984)

# Dataset overview

The training dataset consists of 2,400 samples, comprising 720 males and 1,680 females, made extracting speaker embeddings from face images. A speaker embedding is a small-dimensional, fixed sized representation of an image, where features are continuous values that represent a point in the m-dimensional embedding space. The dataset results in a substantial bias towards females, accounting for 70% of the dataset. Each sample is made of 12 features that do not have a physical interpretation. It is also known that the samples belong to three distinct age groups, each characterized by a different distribution of embeddings. however, no age information is available. The test dataset instead is characterized by 6,000 samples, with 4,200 males and 1,800 females. As such, dataset are imbalanced, with the training set that has significantly more female samples, whereas the test set has significantly more male samples.

The mean (μ) and standard deviation (σ) of each feature for the training datasetare:

[  8.0, 10.6,  6.8,  4.5,  8.5,  9.4,  6.5,  6.5,  10.4,  4.3,  6.2 ]

It can be observed that features of the data set have different scales, they have large differences between their ranges. So, in this case, **Z-normalization** on the data-set to bring all the features on the same scale could be useful. Z-normalization centers the feature columns at mean 0 with standard deviation 1. Thus before applying any operation each sample of the training set has been transformed through the expression:

# Dataset Analysis

Before involve any classifier it is necessary first to perform an analysis of the dataset features. The mean (μ) and standard deviation (σ) of each feature for the training datasetare:

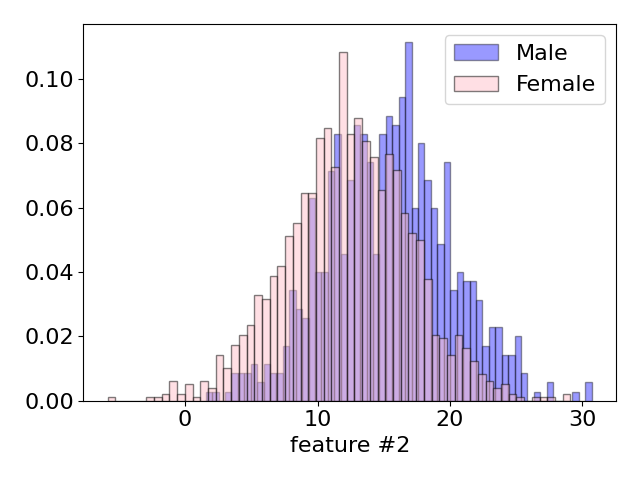
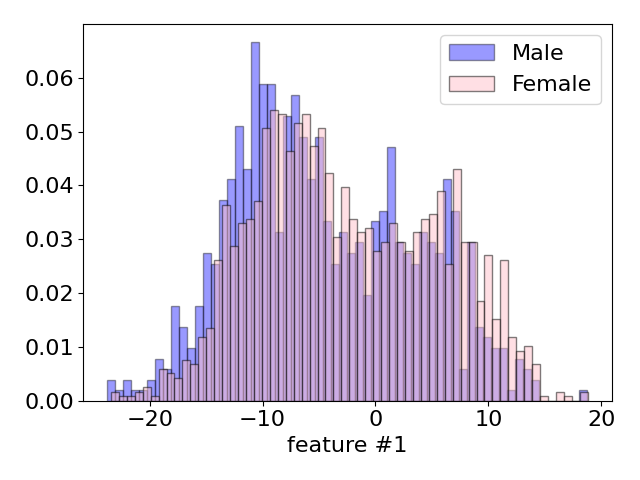
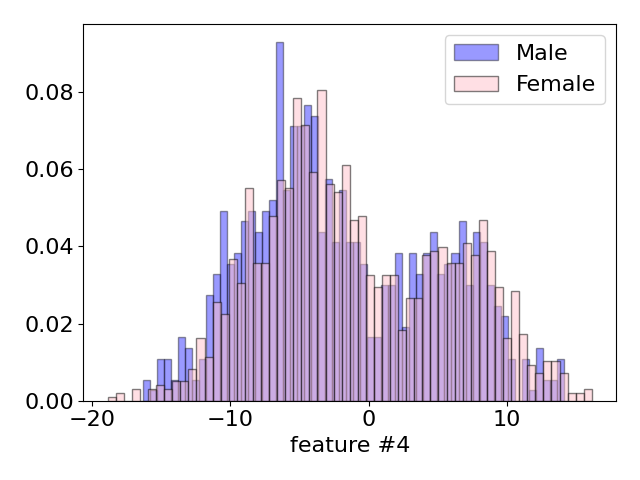
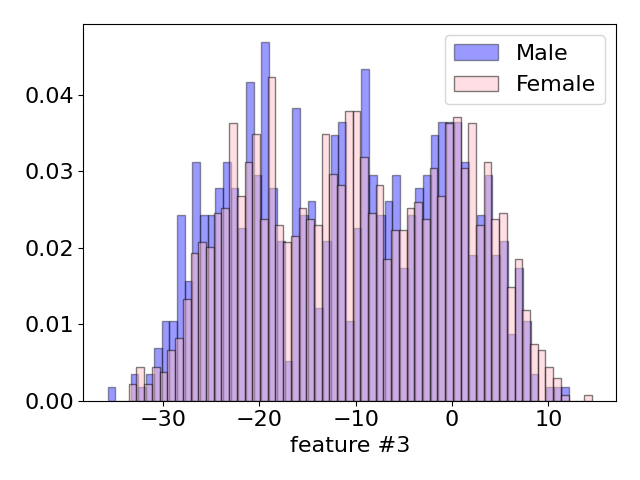
[  8.0, 10.6,  6.8,  4.5,  8.5,  9.4,  6.5,  6.5,  10.4,  4.3,  6.2 ]

The features do not exhibit significantly different scales., there is not a large differences between their ranges. A technique called **Z-normalization** generally is useful to bring all the features on the same scale, by centering the feature columns at mean 0 and with standard deviation 1. It is possible to consider it inside this study, despite not expecting substantial improvements. So, in parallel with the analysis of the raw features, an analysis of the normalized features was conducted by normalizing before applying any operation. Each sample of the training set has been transformed through the expression:

Where 𝐱′ is the sample after the Z-score normalization, while 𝐱 is the original sample in the data set.

Histograms

The initial step involves plotting histograms of each dataset feature to examine their distributions, after we normalized the dataset.



We can see that there are some distributions, such as the ones relative to features 5, 8, 2, that recall directly to a gaussian density. It is also important to observe that some plots, like for example plot number 3, 6 and 7, resemble a distribution made of three gaussian. This can be associable to the 3 groups of ages from where the features are extracted of.

Altogether, the distribution of individual features is consistent across both classes, but there are certain distributions that enable us to differentiate between classes in an easier way. This happens for example in figure 11, where it is possible to observe the most distinguishable feature distribution in our dataset.

Immagine che contiene schermata, testo, Diagramma, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, diagramma

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, linea

Descrizione generata automaticamenteImmagine che contiene testo, schermata, Diagramma, diagramma

Descrizione generata automaticamente

LDA

Scatter plots

The analysis continues by leveraging scatter plots, which are particularly useful to visualize the relationship between two continuous variables. They can help identify patterns, trends, correlations, or clusters within the data. By examining the distribution and dispersion of the dots, you can gain insights into how the variables interact with each other. For our dataset, these plots are aligned to the one present in the gaussian model, and for this reason, we expect that gaussian model are able to perform well on this kind of data.

The following images contains some of the most significant plot. For example in the scatter plot relative to feature #6 it is evident the presence of three clusters. It can be associable again to a gaussian distribution with more components, and directly related to the three groups of age from where the dataset sample are taken.

Immagine che contiene testo, schermata, diagramma, Policromia

Descrizione generata automaticamente Immagine che contiene testo, schermata, diagramma, Policromia

Descrizione generata automaticamente Immagine che contiene testo, schermata, diagramma

Descrizione generata automaticamente

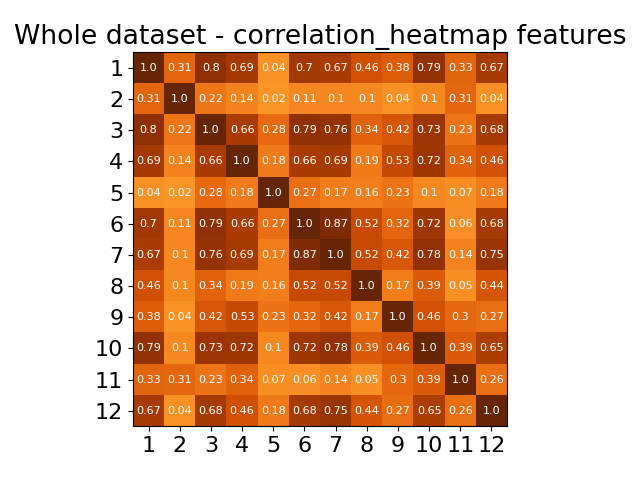
Correlation

A way to analyze features interaction is to compute the **correlation** of features. This is useful also to understand if PCA (Principal Component Analysis) could be useful and how many features can be discarded. **Pearson correlation coefficient** can be used to measure correlation between two features and it can be computed as:

In this analysis only the absolute value of Pearson correlation is considered, because we are only interested to understand if there is correlation or not:

The absolute value of Pearson correlation coefficient can take value between 0 and 1. If 0 it means that the two considered features are uncorrelated, while 1 means that the features are completely correlated (one feature is the scaled version of the other).

To visualize the correlation between features we employed heatmaps. In the following heatmaps, darker colors indicate a strong correlation between two features, while lighter colors suggest a weaker correlation between the two features.

Immagine che contiene testo, schermata, quadrato, Rettangolo

Descrizione generata automaticamente

Immagine che contiene testo, schermata, quadrato, modello

Descrizione generata automaticamente

We plot heatmaps for the correlation considering: the whole data training samples (figure XX); training samples belonging to Male class (figure XX); training samples belonging to Female class (figure XX).

Turning the attention to the heatmap within Figure XX. It becomes evident that some features exhibit strong correlations, such as 1-10 and 3-6, while others appear uncorrelated, like 5-11. Most of the features demonstrate mild correlation between each other, with coefficients hovering around 0.4 to 0.6. This observation hints that it is possible to gain advantages by transforming our data from a 12-dimensional space to a 10-dimensional one, effectively reducing the number of parameters to be estimated. So it is possible to apply **PCA**, that is a technique useful in order to reduce the dimensionality of a dataset. PCA finds a m-dimensional subspace, that is a set of directions over which to project our data set points. More in details PCA finds map projection that minimize the average reconstruction error (i.e. the reconstructed points are as close as possible to the original points). The dimension of the subspace (**m**) is an hyperparameterthat needs to be tuned using a validation set. In the following steps it was considered results of classifiers with hyperparameter **m** up to 10, as suggested in figure XX, where is it possible to notice that we will maintain the 97% of the variance by removing 1/2 dimensions and maintaining the most of the information.

Moving the focus over single class heatmaps, we can notice a strong similarity between the two plots, meaning that the gaussian model (Multivariate Gaussian Model-MVG and TiedMVG) will behave in a similar way in terms of performances. At the same time the reasonable correlation between the features should bring the gaussian models based on the Naïve-bayes assumption to give worst performances. Anyway we will evaluate also the Gaussian classifiers with diagonal covariance matrix to compare them with other Gaussian classifiers.

# Validation approach

In the following pages we took into consideration different machine learning models for classification. Each of them is first trained and then validated on the training data set and then obtained costs .

Generally, for binary problems, the cost function can be divided into two components: the effectiveness of the classifier and the effectiveness of the selected threshold. Initially, we will focus on the classifier's ability to discriminate scores, and thus, we will use the minimum detection cost (min DCF) as a measure of performance.

So models are trained using training set (without validation set samples) then scores and min DCF are computed on validation samples. The model can be trained with different combinations of hyperparameters, results (min DCF computed on the scores of validation samples) of the same model with different hyperparameters are compared to find the most promising combination of hyperparameters.

K-fold cross-validation operates by dividing the training dataset into K folds. In each iteration, K-1 of these folds are utilized for training the model, while the remaining fold is used for validation. This process is repeated K times, and during each iteration, scores are computed for the validation samples. After K iterations, scores have been computed for every sample in the training set, allowing us to calculate the minimum Detection Cost Function (min DCF) using this set of scores.

Once selected all the hyperparameters, the final classifier will be obtained training again over the whole training dataset. Altogether K-fold cross-validation has a significant drawback: it necessitates training and validating models K+1 times, which can demand a substantial amount of computational time.

Generally if feasible the K-fold cross-validation leads to more robust results rather than single split, so to measure the performance of the different classifiers we will employ K-fold cross-validation over the single-fold. Data has been shuffled before splitting, so that the data of different folds are homogeneous.

Because of the increase of the amount of time required to estimate model parameters with higher value of K, we will consider K= 5.

We will consider different applications , reduced to the effective prior . In particularly the applications considered are:

* **Primary application**: uniform prior application : = (0.5, 1, 1)
* Unbalanced application: = (0.1, 1, 1)
* Unbalanced application: = (0.9, 1, 1)

# Multivariate Gaussian classifier (MVG)

The first model that we consider in this analysis are MVG classifiers. We try different pre-processing techniques, such as Z-Score and PCA, to determine if they are effective, also in configurations based on a combination of them.

MVG are generative models**,** based on the idea of trying to model the class distribution of observed samples. MVG classifiers assume that both the training set and evaluation samples are independent and identically distributed (i.i.d.)given a set of parameters 𝜃. In particular, Gaussian distribution for a samples given the class is assumed:

(𝑿𝑖∣𝐶𝑖=𝑐,𝜽)∼(𝑿𝑡∣𝐶𝑡=𝑐,𝜽)∼(𝑿∣𝐶=𝑐,𝜽)∼𝒩(𝝁𝑐,𝚺𝑐)

Using the maximum likelihood estimation is it possible to estimate the model parameters:

𝜇𝑐∗=1𝑁𝑐Σ 𝑖∣𝑐𝑖=𝑐𝑥𝑖, Σ𝑐∗=1𝑁𝑐Σ 𝑖∣𝑐𝑖=𝑐(𝑥𝑖−𝜇𝑐∗)(𝑥𝑖−𝜇𝑐∗)𝑇

𝑁𝑐 is the number of samples belonging to class 𝑐. The log-likelihood ratio (**llr**) acts as a **score**:

s(𝒙𝑡)= llr (𝒙𝑡)=log 𝑓𝑋∣𝐶(𝒙𝑡∣𝑐1)𝑓𝑋∣𝐶(𝒙𝑡∣𝑐0)

The models we consider are different in terms of covariance matrix computation:

• **Full Covariance matrix**(referred as Full-Cov): computes covariance matrix (Σ𝑐∗) without any simplifications. This model is expected to be robust if the number of training samples is far bigger than number of dimensions of samples.

• **Diagonal Covariance matrix – Naïve Bayes** (referred as Diag-cov): Multivariate Gaussian classifier with diagonal covariance matrices where the diagonal element of row-*i* are the variances of the feature-*i* of the training samples. Diag-Cov is a diagonal version of the original full covariance matrix. This model works well in scenario where features are enough uncorrelated. In our dataset there is a reasonable correlation between features, so the Gaussian classifier with diagonal hypothesis is expected to have worst results than full covariance Gaussian classifier.

• **Tied Covariance matrix**: it assumes that gaussian parameters of different classes are related one to each other, and so that covariance matrices of different classes are the same. There is only one covariance matrix corresponding to a weighted average empirical covariance matrix for each class. This model works well when there are classes with few samples.

Our analysis takes into account all the previously mentioned model and in the following the respective results are reported:

*Table 1: Min DCF for MVG with K-fold cross validation (K=5)*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RAW | | | Z-score | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| Full-cov | 0.113 | 0.297 | 0.350 | 0.113 | 0.297 | 0.350 |
| Diag-cov | 0.463 | 0.770 | 0.777 | 0.463 | 0.770 | 0.777 |
| Tied full-cov | 0.109 | 0.299 | 0.341 | 0.109 | 0.299 | 0.341 |
| Tied diag-cov | 0.457 | 0.769 | 0.780 | 0.457 | 0.769 | 0.780 |

*Table 2: Min DCF for MVG with K-fold cross validation (K=5) and PCA(m=11)*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RAW- PCA(11) | | | Z-score -PCA(11) | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| Full-cov | 0.117 | 0.308 | 0.349 | 0.121 | 0.311 | 0.358 |
| Diag-cov | 0.126 | 0.324 | 0.370 | 0.124 | 0.311 | 0.348 |
| Tied full-cov | 0.118 | 0.288 | 0.355 | 0.118 | 0.298 | 0.355 |
| Tied diag-cov | 0.124 | 0.302 | 0.360 | 0.123 | 0.294 | 0.354 |

*Table 3: Min DCF for MVG with K-fold cross validation (K=5) and PCA(m=10)*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | RAW- PCA(10) | | | Z-score -PCA(10) | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| Full-cov | 0.163 | 0.401 | 0.492 | 0.187 | 0.406 | 0.537 |
| Diag-cov | 0.168 | 0.448 | 0.468 | 0.184 | 0.434 | 0.545 |
| Tied full-cov | 0.161 | 0.392 | 0.474 | 0.183 | 0.427 | 0.535 |
| Tied diag-cov | 0.170 | 0.396 | 0.479 | 0.182 | 0.421 | 0.543 |

Overall the results are aligned with the previous expectation. The Z-score normalization pre-processing technique provides results equals to the one of the raw features, because it simply subtracts the mean and scale by the standard deviation each feature.

Relatively to the Naïve-Bayes assumption, it is observable that, as expected, it is not so successful. This because of the presence of a reasonable correlation between the features, which is in contrast with the suitable working conditions for this classifier.

A possible solution in order to avoid this problem is to leverage Principal Component Analysis (PCA). PCA preserves the principal discriminant information in the leading directions, and in this way MVG and TMVG are in condition not to lose information and to keep performances the same. MVG and TMVG have the same performance, in general, because the MVG assumes that we have two different covariance matrices for the classes, but, given that the covariance matrixes of two classes are similar, the TMVG performs in the same way.

SCRIVERE QUELLO CHE SUCCEDE ALLE DIAGONALIZED QUANDO SI FA PCA(MIGLIORA UN SACCO)

Taking into account the results relative to dimensionality reduction, it can be observed that results for the 11 dimensions space are similar to the 12 one. As expected after observing the explained variance plots (figure XX), in this way it is possible to maintain a good fraction of variance.

Best results are obtained without applying PCA, even if the ones obtained with PCA(11) are quite similar. Furthermore, the features pre-processed with Z-score normalization provide results slightly better than the raw ones. Given the limited effectiveness of PCA for generative models, we are going to focus on the whole set of features but still use PCA in the following analysis to be sure.

To conclude, the best MVG classifier option is Tied Full covariance MVG, trained on features pre-processed with Z-score normalization and without applying PCA.

# Logistic Regression

The following pages refers instead to an analysis of **discriminative models,** starting from Logistic Regression. Discriminative models, differently from the generative ones, try to directly model the class posterior distribution, rather than modelling the distribution of observed samples (generative models). It can be expected that PCA has limited effects on Logistic Regression models, since discriminative models does not require specific assumptions on data distribution. Despite this, in the following, we continue taking into consideration results on data pre-processed with PCA and Z-score normalized features to check it.

Linear Logistic Regression

The first logistic regression model that we consider is the **regularized linear** one. It has 2 parameters (𝒘,**𝑏**) obtainable by minimizing the following expression :

𝐽(𝒘,𝑏)=𝜆2∥𝒘∥2+1𝑛Σ 𝑛𝑖=1log (1+𝑒−𝑧𝑖(𝒘𝑇𝒙𝑖+𝑏)),𝑧𝑖={1 if 𝑐𝑖=1−1 if 𝑐𝑖=0( i.e. 𝑧𝑖=2𝑐𝑖−1)

𝑱(𝒘,𝑏) expression refers to the average cross-entropybetween the distribution of observed and predicted labels, plus a regularization term.𝜆 is an hyper-parameter used in order to weight the regularization term.

Since classes on training set are unbalanced, we rebalance the cost of the different classes by minimizing:

𝐽(𝒘,𝑏)=𝜆2∥𝒘∥2+𝜋𝑇𝑛𝑇Σ 𝑛𝑖=1∣𝑐𝑖=1log (1+𝑒−𝑧𝑖(𝑤𝑇𝑥𝑖+𝑏))+1−𝜋𝑇𝑛𝐹Σ 𝑛𝑖=1∣𝑐𝑖=0log (1+𝑒−𝑧𝑖(𝑤𝑇𝑥𝑖+𝑏))

where:

• “” represents the prior that allow to generalize

• “” is the vector that has elements -1 or 1 for x

• “” represents the numbers of training samples belonging to class 1 and 0

As first steps of our analysis we consider the main application, so is used to estimate the value of 𝜆. Then results with and will be explored. After the minimization, that is performed leveraging *scipy.optimize.fmin\_l\_bfgs\_b* function,the scoreof a sample can be computed as:

where:

• “” refers to the orthogonal vector respect to the hyperplane that we have defined

• “” refers to the bias term

The estimation of a value of that give good results is made computing minDCF with different values. The interval chosen to find the best value is from to in a logarithmic way.

Immagine che contiene testo, diagramma, linea, Diagramma

Descrizione generata automaticamenteImmagine che contiene testo, diagramma, linea, schermata

Descrizione generata automaticamente

Figure 1: minDCFwrt Lambda RAW and on the right is ZSCORE

From the plots we can observe that with large value of 𝜆 the model performs worst and is not so good in correctly classify samples. This is related to the fact that the increase of the contribution of the regularization makes the model too simple and unable to behave in an efficient way. A good value for the hyper-parameter could be =, so the following results are obtaining considering this value, which can be considered as a useful trade-off between overfitting and underfitting risk.

*Table 3: Min DCF results for Linear logreg with K-fold cross validation (K=5)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | Linear LR-RAW | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| LR(λ=1e-5, πT=0.5 ) | 0.112 | 0.283 | 0.339 |
| LR(λ=1e-5, πT=0.1 ) | 0.120 | 0.299 | 0.377 |
| LR(λ=1e-5, πT=0.9 ) | 0.110 | 0.315 | 0.344 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Linear LR-ZSCORE | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| LR(λ=1e-5, πT=0.5 ) | 0.112 | 0.283 | 0.339 |
| LR(λ=1e-5, πT=0.1 ) | 0.120 | 0.299 | 0.377 |
| LR(λ=1e-5, πT=0.9 ) | 0.110 | 0.315 | 0.344 |

INSERIRE TABELLA DEI RISULTATI DOPO PCA?

The results obtained with features pre-processed with Z-score normalization are exactly the same of the ones on raw features. Using different values for does not help significantly in the classification performance for specific application. We can however notice that slightly better results are obtained using = 0.9, probably because of the class imbalance of the dataset. As we expected and we can observe in the tables above, scores after PCA pre-processing are really like the one obtained without the technique.

Quadratic Logistic Regression

The other kind of logistic regression model that we are going to consider in our analysis is non-linear Logistic Regression model. To compute quadratic separation surfaces it is possible to define an expanded feature space , where

and is the operator that arrange the columns of matrix .

Non-linear LR model works by leveraging and not simply **.** In this way the model has is characterized by a linear separation surface in the expanded space, but effectively computes quadratic separation boundaries within the original space.

It is expected that the quadratic logistic regression model performs worst than the linear one, because of the characteristics of data. As seen in the dataset analysis section, the scatter plots and the histograms show that classes are suitable for being separated with linear decision rules.

Plotting the min DCF with different values of λ , we find out that the best value of lambda is for raw features and . Differently from the linear model, results on raw features and on normalized ones are different, because of the feature expansion and non-linear transformation to a different embedding space. Another difference from the linear model is that, for raw features, in this case the regularization term is useful and provides benefit to the quadratic classifier.

Altogether the results are worse than the linear model. PCA does not improve the performance and so results are not reported.

Overall the most promising option is Linear Logistic regression with and , both on raw and z-scored features and without PCA.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Quadratic LR-RAW | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| LR(λ=100, πT=0.5 ) | 0.121 | 0.299 | 0.363 |
| LR(λ=100, πT=0.1 ) | 0.125 | 0.313 | 0.406 |
| LR(λ=100, πT=0.9 ) | 0.122 | 0.331 | 0.356 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Quadratic LR-ZSCORE | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| LR(λ=1e-3, πT=0.5 ) | 0.125 | 0.334 | 0.332 |
| LR(λ=1e-3, πT=0.1 ) | 0.131 | 0.353 | 0.360 |
| LR(λ=1e-3, πT=0.9 ) | 0.135 | 0.323 | 0.313 |

# Support Vector Machines

Our analysis now shifts the focus to the Support Vector Machine (SVM), which represents another instance of a supervised discriminative model for classification.

Similar to logistic regression (LR), the SVM model aims to discover a hyperplane that effectively separates the classes. However, unlike the LR model, SVM seeks to identify the hyperplane that maximizes the margin.

Linear SVM

Linear SVM seek for separation hyperplane which maximizes the margin. The linear SVM objective consists in minimizing the following expression:

𝐽̂(𝒘̂)=12∥𝒘̂∥2+𝐶Σ 𝑛𝑖=1𝑚𝑎𝑥(0,1−𝑧𝑖(𝒘̂𝑇𝒙̂𝑖))

Where 𝒘̂ includes also the bias term b and 𝒙̂𝑖 is 𝒙𝒊 extended with 1. C is an hyperparameter.

𝒙̂𝑖=[𝒙𝑖1], 𝒘̂=[𝒘𝑏]

We take into consideration the **dual formulation,** because more easily manageable. It can be obtained using the Lagrange multiplier. SVM objective becomes the maximization of the expression

Immagine che contiene Carattere, testo, tipografia, calligrafia

Descrizione generata automaticamente

Where



To maximize the dual formulation we used the function *scipy.optimize.fmin\_l\_bfgs\_b.* This implementation compute minimizer of a function, for this reason we perform minimization of −𝐽̂𝐷(𝜶) and not maximization of 𝐽̂𝐷(𝜶).

Once solution with respect to 𝜶 is computed, the primal solution can be retrieved as:

Immagine che contiene Carattere, bianco, diagramma, linea

Descrizione generata automaticamente

Finally the **score** for a sample 𝒙𝑡 is:

Immagine che contiene Carattere, schermata, Elementi grafici, calligrafia

Descrizione generata automaticamente

Different values of hyperparameter C are used for two different classes. This because we want to make the classes balanced :

-

- For samples of class 1 the hyperparameter considered is 𝐶𝑖= 𝐶𝑇 =𝐶𝜋𝑇𝜋𝑇𝑒𝑚𝑝

- For samples of class 0 the hyperparameter considered is 𝐶𝑖= 𝐶𝐹 =𝐶𝜋𝐹𝜋𝐹𝑒𝑚𝑝

*Table 3: Min DCF results for Linear svm with K-fold cross validation (K=5)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | RAW | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| LR(C=0.1, πT=0.5 ) | 0.115 | 0.294 | 0.351 |
| LR(C=0.1, πT=0.1 ) |  |  |  |
| LR(C=0.1, πT=0.9 ) |  |  | 0.339 |

*Table 3: Min DCF results for Quadratic svm with K-fold cross validation (K=5)*

|  |  |  |  |
| --- | --- | --- | --- |
|  | RAW | | |
| Model | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| LR(C=0.1, πT=0.5 ) | 0.115 | 0.294 | 0.351 |
| LR(C=0.1, πT=0.1 ) |  |  |  |
| LR(C=0.1, πT=0.9 ) |  |  | 0.339 |

# Gaussian Mixture Model

Our analysis take into considerations as last classifier Gaussian Mixture Model (GMM). This is a generative approach which can approximate generic distributions without any supposition on distribution of data (unlike MVG). In general these assumes that data can be distributed as gaussian with one or more components. The models that we take into consideration are Full covariance, diagonal covariance, full tied covariance and diagonal tied covariance. Once again a tuning to select the number of Gaussians G was done utilizing the K-fold approach.

In the following we plot the min DCF with different number of components to understand which is the optimal number of components for each GMM models. The plotting is made without PCA and with . We leverage LBG algorithm to incrementally start from a GMM with 2 components to a GMM with G components. In this case the hyper-parameter to tune is the number of components.

The previous analysis regarding Gaussian models and dataset analysis indicate that the models that can perform very well are GMM and Tied GMM, for the same reasons we explained earlier.

Remembering that the training set is characterized by a distribution that is similar to a gaussian with 3 components, we expect that best results are obtained with a GMM model with 2 components or with 4 components.

CAMBIA I COLORI E I TITOLI DEI GRAFICI!!!

For each GMM option we select the number of components that provides the lowest min DCF for and we proceed with a further analysis. Here are shown the results of the selected GMM’s.

TABELLE GMM FOR EACH COMPONENT AND FOR ZSCORE AND RAW FEATURES

As in the MVG classifier, diagonal models have performances that are lower than the full ones, again because of the reasonable correlation between features, shown in the heatmaps. We do not consider gaussian with one single component.

Altogether the GMM produces really encouraging results. The best options are GMM Full covariance with 4 components and GMM Tied covariance with 8 components. Results relative to raw features and z-score ones are very similar. Despite the Full-Tied Covariance GMM with 8 components provide results slightly better than the one with 4 components, our knowledge on the fact that the data samples are taken from 3 groups of different ages brings to choose 4 components.

# Best Model

INSERISCI TABELLA CON I BEST MODEL DI OGNI FAMIGLIA

From the overall results relative to the validation phase, the model that we will consider as the best for the evaluation phase is GMM Full-tied Covariance with 4 components, without PCA. During the evaluation phase we will also perform experiments with other classifier for completeness.

GMM full 4components pi=0.5 raw: 0.0706

GMM full tied 8components pi=0.5 raw : 0.06

GMM FULL TIED 4 Component zscore is the best

# Score calibration

Until this moment we use minDCF to compare the models performances. This is the cost if the optimal threshold is acknowledged. MinDCF measures the cost to pay if optimal decisions for the evaluation set are made using the recognizer scores. However what it is paid in practice is not the minimum cost, but the **actual cost**. In practice, determining the optimal threshold for evaluation data is impossible, since it necessitates knowledge of the evaluation labels, which is unfeasible. In general if scores are well calibrated the optimal threshold is a threshold that optimize the Bayes risk, so scores have an optimal threshold that corresponds to the theoretical one**:**

where is the effective prior. **Score could be well calibrated** and this is related to the fact that recognizer has given well calibrated scores or to the fact that has been performed re-calibration. In this case it is possible to use theoretical threshold to compute the actual cost. Sometimes it can happen that score **are not well calibrated**, and in this case it is necessary to estimate, leveraging the validation set, a good threshold for the application. It is possible to check if the output scores of selected the models are well calibrated comparing the min DCF with the actual cost obtained using the theoretical threshold for each application.

In the following we compare minimum and actual cost, relative to target application with different prior, leveraging Bayes error plot. To estimate the calibration function's parameters, we will utilize a K-Fold Approach.

INSERIRE GRAFICI GMM CALIBRATION WITH RAW AND ZSCORE FEATURES

As we can see from the bayes error plot, calibration does not help with any consistent improvement, so overall we can say that this transformation is unnecessary. As demonstrated by the results below, there is not a consistent benefit, but only the unbalanced application with 𝜋̃ = 0.9 showed a small enhancement.

*Table 3: Min DCF and Act DCF for GMM full tied wit 4 components*

|  |  |  |  |
| --- | --- | --- | --- |
| GMM | Z score feature | | |
|  | **π= 0.5** | **π= 0.1** | **π= 0.9** |
| MinDCF | 0.067 | 0.237 | 0.223 |
|  |  |  |  |
|  |  |  |  |
| ActDCF | 0.071 | 0.249 | 0.244 |

We can see that there is not a consistent difference between results on calibrated and uncalibrated scores. Gmm already provides good performance without a consistent need of calibration, so in the following we continue to operate without taking into account this transformation.

In order to check its effectiveness we also tried score calibration on other models, such as logistic regression classifier. In the part below is it possible to see the Bayes error plot of uncalibrated and calibrated versions of logistic regression:

INSERIRE GRAFICI LOGReg CALIBRATION WITH RAW AND ZSCORE FEATURES

It is clear that the scores in this case are not well calibrated, for this reason a calibration can easily bring to a consistent improvement in terms of results.

INSERIRE risultsti LOGReg CALIBRATION

# Fusion

As said before, the model that we are considering as the best (GMM full-tied with 4 components) does not require score calibration. Furthermore, the analysis in the following will proceed taking into consideration only z-score features.

In this section we leverage the idea of combining different classifier with a fusion approach. In general classifiers, because of their different assumptions, will provide different results depending on the type. They could agree on some decisions while disagree on others. The overall idea is to combine the decisions of both in order to result in better predictions labels. In practice, two classifiers can be combined in simple voting scheme approach: each classifiers assign a label and at the end the label assigned more often is selected. The simple voting approach has some issues, if one classifier is almost certain about class 1 and two other classifiers are only slightly in favor of class 0 it is not granted that assigning label 0 is a good choice. So, rather than fusing classifiers at decision level, is better to perform a **score-level fusion** voting. The idea is to introduce a **fused score** which is a function of the scores of different classifiers. Considering a sample , if is the score provided by classifier A, while is the score provided by classifier B, the fused score for sample will be a fusion of this, based on a function like this

Where are parameters to be estimated.

The scores of different classifiers are treated as a feature vector. A prior-weighted logistic regression is used to train the model parameters similar to what has been done for score calibration.

We perform fusion considering three different models: GMM full tied with 4 components, Linear logistic regression with and , and SVM RBF with .

GMM è il full tied 4 component

LR è logreg linear con piT=0.9 e lambda 10^-5

SVM è piT=0.9 RBF SVM no rebalancing -gamma =0.100000 -C=10.000000 - pi = 0.500000 -> minDCF = 0.090675

INSERIRE IMMAGINI FUSION ROC AND BAYES ERROR PLOT

DA FIXARE SCORE CALIBRATION E ROC QUANDO FA SVM.TRAIN ecc, ci sono errori

As we can see from the bayes error plots, the fused models do no need calibration.

In the table below are reported the results of the fusion models compared with the results of selected models.

INSERIRE TABELLA CON RISULTATI SENZA FUSION E CON FUSION

# Experimental results on evaluation set