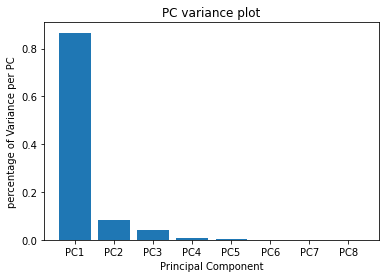
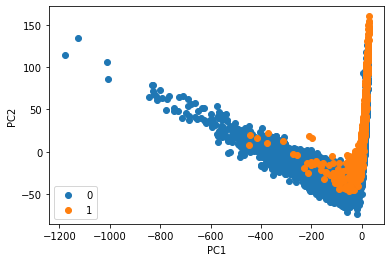
**PCA**

First of all we tried to see if the class separation could be helped by appling a PCA step on the raw data.



We isolated only the first two principal components because the algorithm identified them as the ones with the greatest variance.



But as we can see this method didn’t enanched the class separation, but we decided to implementi t anyway to confirm our hypotesis.

**GAUSSIAN CLASSIFIERS**

Testing Full Covariance, NaiveBayes and Tied Covariance Classifier it highlights Naive Bayes and Full Covariance classifiers perform worst w.r.t. Tied Covariance Classifier on raw data. We used for our first evaluations a single fold and then compared the results with a 5-fold approach dataset with different pre-processing method.

|  |  |  |  |
| --- | --- | --- | --- |
| SINGLE FOLD | **Raw Data** | **Gaussianized Data** | **Z-Normalized Data** |
| ***Full Covariance*** | 0.041 | 0.083 | 0.043 |
| ***Naive Bayes*** | 0.063 | 0.058 | 0.063 |
| ***Tied Covariance*** | **0.024** | 0.060 | 0.024 |

|  |  |  |  |
| --- | --- | --- | --- |
| 5-FOLD | **Raw Data** | **Gaussianized Data** | **Z-Normalized Data** |
| ***Full Covariance*** | 0.039 | 0.078 | 0.039 |
| ***Naive Bayes*** | 0.062 | 0.057 | 0.062 |
| ***Tied Covariance*** | **0.022** | 0.055 | 0.022 |

Analyzing the obtained results we infer that a 5-fold approach give us better overall result on errors. Now on we are going to evaluate our models only with a 5-fold approach.

We chose to leave behind the Naive Bayes because gave us the worst result. Then we evaluated minDCF and DCF on different applications:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Raw Data | **(0.5, 1, 1)** | | **(0.1, 1, 1)** | | **(0.9, 1, 1)** | |
| **minDCF | DCF** | | | | | |
| ***Full Covariance*** | 0.141 | 0.161 | 0.281 | 0.338 | 0.662 | 0.954 |
| ***Tied Covariance*** | **0.113** | 0.191 | 0.221 | 0.270 | 0.568 | 1.433 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Gaussianized Data | **(0.5, 1, 1)** | | **(0.1, 1, 1)** | | **(0.9, 1, 1)** | |
| **minDCF | DCF** | | | | | |
| ***Full Covariance*** | 0.153 | 0.177 | 0.241 | 0.399 | 0.702 | 0.868 |
| ***Tied Covariance*** | 0.131 | **0.140** | 0.231 | 0.246 | 0.533 | 0.607 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Z normalized | **(0.5, 1, 1)** | | **(0.1, 1, 1)** | | **(0.9, 1, 1)** | |
| **minDCF | DCF** | | | | | |
| ***Full Covariance*** | 0.140 | 0.161 | 0.281 | 0.338 | 0.662 | 0.954 |
| ***Tied Covariance*** | **0.113** | 0.191 | 0.222 | 0.270 | 0.568 | 1.433 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PCA (m = 5) | **(0.5, 1, 1)** | | **(0.1, 1, 1)** | | **(0.9, 1, 1)** | |
| **minDCF | DCF** | | | | | |
| ***Full Covariance*** | **0.181** | 0.185 | 0.424 | 0.482 | **0.723** | 0.783 |
| ***Tied Covariance*** | 0.178 | *0.198* | 0.312 | 0.322 | 0.599 | 1.126 |

As supposed in the relative paragraph, PCA didn’t improve our estimate. The Tied Covariance model with z-normalized and raw data obtains in 5-fold cross validation protocol providing the **lowest minDCF.** It seems raw and z-normalized data give same results we can state pre-processing is useless with Gaussian models.

Overall the best candidate is the MVG model with Tied Covariance matrices.

**LOGISTIC REGRESSION**

After we gave to the algorithm raw and PCA pre-processed data, we obtained overflow errors which didn’t allow us to proceed with analysis. So we transformed our data using Z-normalization and gaussianization.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Gaussianized Data | **(0.5, 1, 1)** | | **(0.1, 1, 1)** | | **(0.9, 1, 1)** | |
| **minDCF | DCF** | | | | | |
| ***Lambda = 0*** | 0.128 | 0.160 | 0.227 | 0.547 | 0.524 | 0.757 |
| ***Lambda = 0.1*** | 0.158 | 0.425 | 0.286 | 0.983 | 0.605 | 0.855 |
| ***Lambda = 0.01*** | 0.147 | 0.215 | 0.268 | 0.711 | 0.532 | 0.886 |
| ***Lambda = 0.0001*** | 0.130 | 0.167 | 0.230 | 0.556 | 0.523 | 0.791 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Z-normalized Data | **(0.5, 1, 1)** | | **(0.1, 1, 1)** | | **(0.9, 1, 1)** | |
| **minDCF | DCF** | | | | | |
| ***Lambda = 0*** | 0.113 | 0.183 | 0.211 | 0.315 | 0.545 | 0.865 |
| ***Lambda = 0.1*** | **0.167** | 0.422 | 0.278 | 0.706 | 0.616 | 1.124 |
| ***Lambda = 0.01*** | 0.143 | 0.268 | 0.249 | 0.451 | 0.567 | 1.012 |
| ***Lambda = 0.0001*** | **0.112** | 0.193 | 0.211 | 0.326 | 0.551 | 0.866 |

We tried different value of LAMBDA and we can state lambda = 0 give us the best minDCF with Z-normalized data. Both on balanced and unbalanced apps.

**SUPPORT VECTOR MACHINE**

Here, we’re going to analyze Linear SVM first, then kernel SVM (polynomial and RBF).

**LINEAR SVM**

We used for our analysis raw, gaussianized and z-normalized data providing only a balanced application cause this kind of approach requires a lot of time and computational power:

|  |  |  |
| --- | --- | --- |
| Raw data | **(0.5, 1, 1)** | |
| **minDCF | DCF** | |
| ***K = 1, C = 0.1*** | 0.147 | 0.225 |
| ***K = 1, C = 1*** | **0.74** | 0.907 |
| ***K = 1, C = 10*** | 0.809 | 0.872 |
| ***K = 10, C = 0.1*** | **0.184** | 0.194 |
| ***K = 10, C = 1*** | **0.822** | 0.831 |
| ***K = 10, C = 10*** | **0.758** | 0.852 |

|  |  |  |
| --- | --- | --- |
| Gaussianized Data | **(0.5, 1, 1)** | |
| **minDCF | DCF** | |
| ***K = 1, C = 0.1*** |  |  |
| ***K = 1, C = 1*** |  |  |
| ***K = 1, C = 10*** |  |  |
| ***K = 10, C = 0.1*** |  |  |
| ***K = 10, C = 1*** |  |  |
| ***K = 10, C = 10*** |  |  |

|  |  |  |
| --- | --- | --- |
| Z-normalized data | **(0.5, 1, 1)** | |
| **minDCF | DCF** | |
| ***K = 1, C = 0.1*** | 0.147 | 0.225 |
| ***K = 1, C = 1*** | **0.74** | 0.907 |
| ***K = 1, C = 10*** | 0.809 | 0.872 |
| ***K = 10, C = 0.1*** | **0.184** | 0.194 |
| ***K = 10, C = 1*** | **0.822** | 0.831 |
| ***K = 10, C = 10*** | **0.758** | 0.852 |