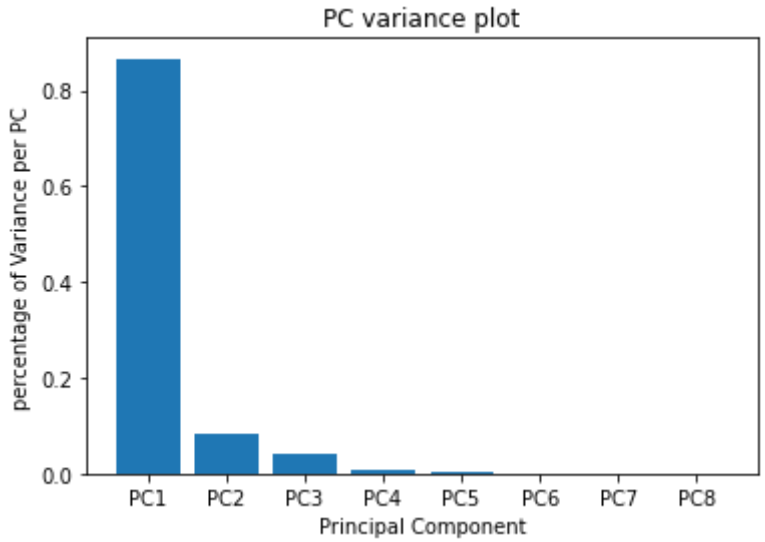
PULSAR DETECTION PROJECT

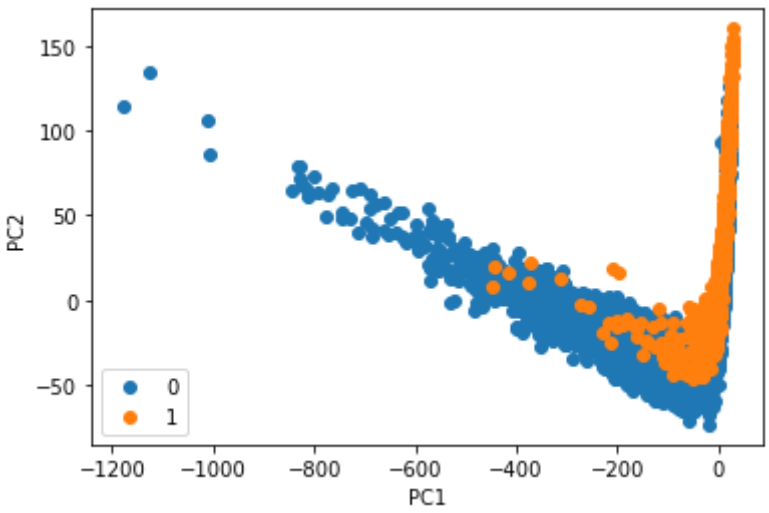
We‘re going to analyze data about stars in order to determine if they’re pulsar or not. Every sample has 8 feature and classes are unbalanced. We’re going to use probabilistic and non-probabilistic models both. Let’s start employing a dimensionality reduction method first.

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 implement it anyway to confirm our hypotesis.

GAUSSIAN CLASSIFIERS

Testing Full Covariance, Naive Bayes 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.

Now on we are going to evaluate our models by computing optimal Bayes decision. We did this simply retrieving the sample scores and then appending them in one vector at each step of k-fold. Then, we compared each score to the threshold $-\log\left(\frac{\pi}{1-\pi}\right)$ (where π is the effective prior) and assigned to each sample the predicted class label: 1 if the score is greater than the threshold, 0 otherwise. Then, through the resulting confusion matrix we evaluated minDCF and DCF on different applications:

Naive Bayes Classifier	**(0.5, 1, 1)**	
	minDCF | DCF	
Raw data	0.193	0.194
Gaussianized data	**0.152**	0.158
Z-normalized data	0.193	0.194

In order to prove our hypotesis we run also Naive Bayes model on the balanced application. As we can see comparing the results with the ones of the other tables we can state this model performs worse.

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, the **lowest minDCF.** It seems raw and z-normalized data give same results we can state pre-processing is almost useless with Gaussian models.

Overall the best candidate is the model with Tied Covariance matrix.

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. Initially we employed a single-fold approach in order to evaluate which Lambda parameters use for our analysis, then we stick to 5-fold validation approach. In the following we show the results obtained with the chosen Lambda:

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

After trying these values of LAMBDA we can state lambda = 0.0001 give us the best minDCF with Z-normalized data.

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	0.740	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	0.134	0.170
K = 1, C = 1	0.127	0.157
K = 1, C = 10	0.128	0.152
K = 10, C = 0.1	0.134	0.167
K = 10, C = 1	0.128	0.155
K = 10, C = 10	0.131	0.154

Z-normalized data	(0.5, 1, 1)	
	minDCF | DCF	
K = 1, C = 0.1	**0.109**	0.208
K = 1, C = 1	0.111	0.193
K = 1, C = 10	0.111	0.190
K = 10, C = 0.1	**0.109**	0.208
K = 10, C = 1	0.111	0.194
K = 10, C = 10	0.130	0.179

As we can see, Linear SVM performs better with Z-normalized data while the worst minDCF is obtained executing the model on raw data. So, we can deduce pre-processing methods are pretty useful for this kind of model. Now we're going to evaluate also non-linear model (kernel SVM).

POLYNOMIAL KERNEL SVM

In this model we are going to analyze performance on each pre-processed dataset trying different combinations of hyperparameters.

Raw data	(0.5, 1, 1)	
	minDCF | DCF	
K = 0, C = 1, d = 2, c = 0	0.950	0.995
K = 1, C = 1, d = 2, c = 0	0.706	0.835
K = 0, C = 1, d = 2, c = 1	0.634	0.718
K = 1, C = 1, d = 2, c = 1	0.908	0.995

Gaussianized data	(0.5, 1, 1)	
	minDCF | DCF	
K = 0, C = 1, d = 2, c = 0	0.953	1.114
K = 1, C = 1, d = 2, c = 0	0.883	0.993
K = 0, C = 1, d = 2, c = 1	**0.559**	0.608
K = 1, C = 1, d = 2, c = 1	0.570	0.633

Z-normalized data	(0.5, 1, 1)	
	minDCF | DCF	
K = 0, C = 1, d = 2, c = 0	**0.985**	1.220
K = 1, C = 1, d = 2, c = 0	0.948	1.001
K = 0, C = 1, d = 2, c = 1	0.575	0.919
K = 1, C = 1, d = 2, c = 1	0.561	0.920

As we can see from tables, in general without focusing on dataset type, minDCFs are pretty high. Anyway the best estimate is provided from the model trained with Gaussianized data. Model working with Z-normalized data gave the worst result.

RADIAL BASIS FUNCTION KERNEL SVM

Now we evaluate our data using the last non-linear SVM approach:

Raw data	(0.5, 1, 1)	
	minDCF | DCF	
K = 0, C = 1, gamma = 1	0.100	0.100
K = 0, C = 1, gamma = 10	0.105	0.187
K = 1, C = 1, gamma = 1	0.219	0.434
K = 1, C = 1, gamma = 10	**0.583**	0.637

Gaussianized data	(0.5, 1, 1)	
	minDCF | DCF	
K = 0, C = 1, gamma = 1	0.115	0.158
K = 0, C = 1, gamma = 10	0.095	0.095
K = 1, C = 1, gamma = 1	0.102	0.168
K = 1, C = 1, gamma = 10	**0.069**	0.210

Z-normalized data	(0.5, 1, 1)	
	minDCF | DCF	
K = 0, C = 1, gamma = 1	0.195	1.000
K = 0, C = 1, gamma = 10	0.103	0.121
K = 1, C = 1, gamma = 1	0.185	1.000
K = 1, C = 1, gamma = 10	0.147	1.000

In this analysis the model with Gaussianized data performs very well obtaining one of the best estimate ever, while model working with raw data returns worst result. So, in this case we can argue that pre-processing step is useful in order to obtain better estimates: indeed we obtain minDCFs with the same amplitude despite of what we got from the model with raw data.

GMM

At this point we're going to analyze different type of GMM classifiers (Full Covariance, Naive Bayes and Tied Covariance) for the balanced application. Running these models our code struggled because sometimes threw exception due a Singular Covariance Matrix. This happened in:

- Full Covariance Model with more than 16 GMM components for raw data, 8 for Gaussianized data and 8 for Z-normalized data;
- Naive Bayes Model with more than 16 GMM components for raw data, 8 for Gaussianized data and 16 for Z-normalized data ;
- Tied Covariance Model with more than 16 GMM components for raw data, 16 for Gaussianized data and 8 for Z-normalized data;

In the following we're going to show the minDCFs and DCFs for each classifier with the maximum number of components we were able to compute:

Raw data	(0.5, 1, 1)	
	minDCF | DCF	
Full Cov - components: 16	0.305	0.312
Diag Cov - components: 16	0.613	0.923
Tied Cov - components: 16	0.308	0.312

Gaussianized data	(0.5, 1, 1)	
	minDCF | DCF	
Full Cov - components: 8	0.390	0.595
Diag Cov - components: 8	0.405	0.461
Tied Cov - components: 16	0.409	0.666

Z-normalized data	(0.5, 1, 1)	
	minDCF | DCF	
Full Cov - components: 8	0.277	0.318
Diag Cov - components:16	0.295	0.308
Tied Cov - components: 8	**0.273**	0.324

As we can see, estimates are pretty high compared to the ones of the other analyzed models probably because the components number is still too low. So, we're going to discard this method.

FINAL EVALUATIONS

At this point, we can directly compare each model and choose the one which offers the best estimate in terms of minDCF.

Model	Data Type	minDCF
Gaussian Model (Tied Covariance Matrix)	Raw	0.113
Logistic Regression (lambda = 0.0001)	Z-normalized	0.112
Linear SVM (K = 1, C = 0.1)	Z-normalized	0.109
Polynomial SVM (K = 0, C= 1, d = 2, c = 1)	Gaussianized	0.559
Radial SVM (K = 1, C = 1, gamma = 10)	Gaussianized	**0.069**
GMM (Tied Covariance Matrix - components: 8)	Z-normalized	0.273

Comparing the best estimate for each model of classification we can see the best choice by far is Radial Support Vector Machine which offered the best minDCF. So, we're going to employ this model on our Test Data and see which estimates we can get out.

In order to get the lowest DCF we need to estimate the optimal threshold through a cross-validation step using a 5-fold approach employing Radial SVM model. Then we train our model using the full Training Set. After that we can classify each sample passing to the optimal Bayes decision algorithm scores, test sample labels and the optimal threshold. So we computed the predicted class labels and returned the Confusion Matrix through which we can retrieve some useful statistics such as DCF, error rate, sensitivity (TPR) and specificity (TNR). We obtain the following results:

- Application: [0.5, 1, 1]
- Error rate: 5.4 %
- DCF: 0.173
- Sensitivity (TPR): 87.4 %
- Specificity (TNR): 95.3 %

As we can see, our error rate is pretty low. As expected the dataset is unbalanced and the train step isn't able to learn very well how to identify sample belonging to True class instead of those beloning to the False class which are recognized pretty well.

AUTHORS

- Russo Francesco Pio, s292473
- Guarino Francesco, s280134