Detection of Pulsars from Radio Telescope Data using Linear and Non-Linear Machine Learning Models

Federico Boscolo
s294908@studenti.polito.it

Antonio De Cinque
s303503@studenti.polito.it

27 June 2022

Abstract

Pulsars are a kind of Neutron star that, in the first phases of their formation, rapidly rotate emitting radio frequencies at regular intervals, detectable from Earth using large radio telescopes. Finding a pulsar involves looking for periodic radio signals, but the vast majority of such signals are generated by radio interference, making pulsars very hard to find. This report aims to examine some Machine Learning approaches, with the goal to classify the HTRU2 dataset. The following linear models proved to be the most effective: Linear Logistic Regression and MVG Tied with full covariance.

Contents

1 Introduction

1.1 A brief introduction to Pulsars

Pulsars are a type of Neutron star that is quite rare. They produce periodic radio signals that travel all the way to our solar system and Earth. They are of considerable scientific interest as probes of space-time, the inter-stellar medium, and states of matter.

The first pulsar was discovered by Jocelyn Bell in 1967, while examining data from a newly built radio telescope. Initially, the data was dismissed as radio interference but was then measured with another telescope, confirming the existence of a rotating radio source in the universe, later confirmed to be a neutron star. Since then, many pulsars have been found throughout the universe, which led to scientist being able to study neutron stars for the first time. This allowed researchers to get a glimpse at the behavior of matter at nuclear density. Other applications for pulsars include maps and clocks thanks to their very precise periods of rotation.

1.2 Data Set Information

HTRU2 is a dataset which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South).

The data set contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples, for a total of 17,898 total examples. These examples have all been checked by human annotators.

The dataset has been split into Training and Evaluation (Test) data. The training set used for this application contains 8108 spurious examples and 821 real pulsar examples. The evaluation set contains 8151 spurious examples and 818 real pulsar examples. The samples are encoded as follows: the list of features is stored on a single line of a file separated by a comma, and the class label lies at the end of the line.

1.3 Attribute Information

The data set is clean and complete. There is no missing data, and the number of features is small.

Each candidate is described by 8 continuous variables. The first four are simple statistics obtained from the integrated pulse profile. This is an array of continuous variables that describe a longitude-resolved version of the signal that has been averaged in both time and frequency. The remaining four variables are similarly obtained from the DM-SNR curve. They are summarized below:

Feature Name	Range	Type
Mean of the integrated profile	6.1796875 - 186.023437	Real
Standard deviation of the integrated profile	24.77204176 - 98.77891067	Real
Excess kurtosis of the integrated profile	-1.730781724 - 8.069522046	Real
Skewness of the integrated profile	-1.791885981 - 68.10162173	Real
Mean of the DM-SNR curve	0.213210702 - 209.3001672	Real
Standard deviation of the DM-SNR curve	7.370432165 - 110.6422106	Real
Excess kurtosis of the DM-SNR curve	-2.812353306 - 34.53984419	Real
Skewness of the DM-SNR curve	-1.976975603 - 1191.000837	Real

Table 1: Description of training set features and their value range

2 Data preparation

2.1 Z-normalization

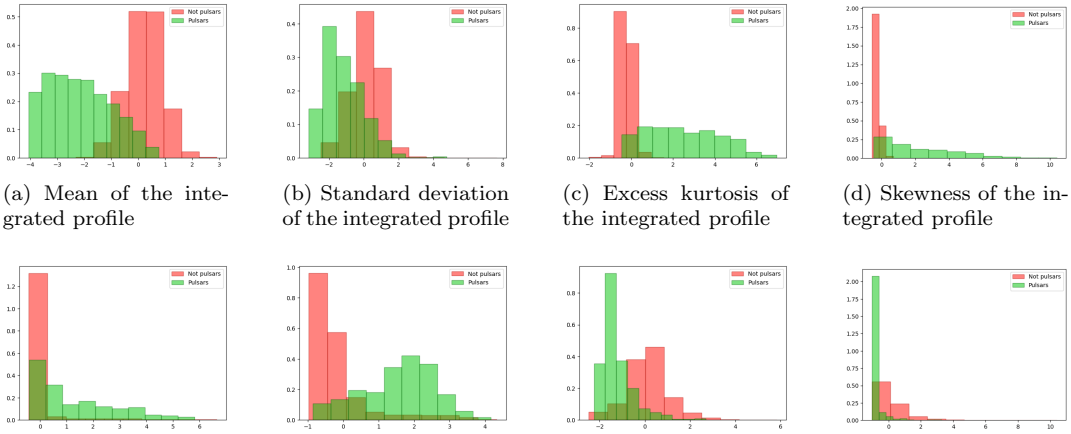
The numerical distribution of features turns out to be extremely unbalanced. The variation between the eight features is too large. Such distributions are not optimal for our purposes. Without normalization, training is difficult to converge and overflow problems may arise. Therefore, training data has been pre-processed with Z-normalization.

Z-normalization is a data preparation technique which puts all data in a range such that its mean is zero and its standard deviation equals one. It is described by the following mathematical formula:

$$Z = \frac{x - \mu}{\sigma}$$

where x represents the original feature vector, μ is the mean of that feature vector, and σ is the standard deviation.

Data pre-processed by means of Z-normalization can then be plotted and compared in a fairly straightforward manner.



(a) Mean of the integrated profile

(b) Standard deviation of the integrated profile

(c) Excess kurtosis of the integrated profile

(d) Skewness of the integrated profile

(e) Mean of the DM-SNR curve

(f) Standard deviation of the DM-SNR curve

(g) Excess kurtosis of the DM-SNR curve

(h) Skewness of the DM-SNR curve

Figure 1: Feature distributions of the Z-normalized HTRU2 training set

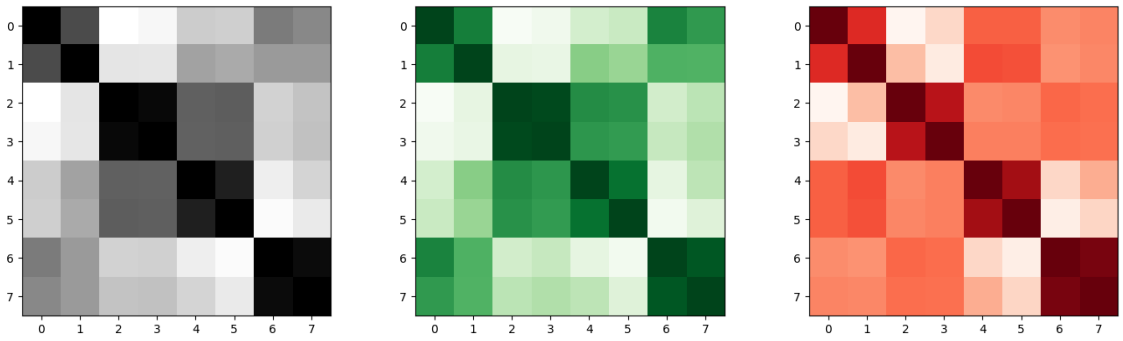
In Figure 1, green histograms refer to "Pulsars" and red histograms refer to "Not pulsars". A preliminary analysis of the training data shows that the normalized features seem well distributed, without evident outliers. Therefore, no further pre-processing has been considered.

2.2 Correlation analysis

Heatmaps of the Z-normalized features have been plotted to analyze features correlation. The heatmaps show, for each feature, the Person correlation coefficient:

$$\frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}$$

which allows for a graphical and intuitive way of picturing correlation among features.



(a) Gray: Whole training set (b) Green: True pulsar samples (c) Red: False pulsar samples

Figure 2: Heatmaps of the Z-normalized HTRU2 training set

In Figure 2, the heatmaps show that features 2-3, features 4-5 and features 6-7 are strongly correlated, while features 0-1 are moderately correlated. This suggests that using PCA (Principal Component Analysis) may be beneficial. PCA is a linear technique for dimensionality reduction, which performs a linear mapping of the data from the n-dimensional feature space to a m-dimensional linear space, with $m \ll n$, in such a way that the variance of the data in the low-dimensional representation is maximized.

PCA could be used to reduce the training set dimensionality up to 4, thus eliminating features which could be seen as redundant. Too steep of a reduction, however, and results may worsen significantly. The effects of dimensonality reduction on training data are analyzed in the following section.

3 Classifying HTRU2 features

Three different applications are considered: a uniform prior application and two unbalanced applications where the prior probability $\tilde{\pi}$ is biased towards one of the two classes (C_{fp} and C_{fn} refer, respectively, to the cost assigned to a false positive misclassification and to a false negative one):

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1)$$
$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1)$$

The target application is the balanced one.

To understand which model is most promising, and to assess the effects of using PCA, two main approaches may be adopted: single-fold and K-Fold cross validation.

The first approach consists in splitting the training set into development (for model training) and validation subsets. This approach is called **single-fold**. It entails a 2:1 ratio split between training data (66.6%) and validation data (33.3%) from the initial training set.

The second approach is called **K-Fold cross-validation**. Cross-validation is a re-sampling procedure used to evaluate machine learning models on a data set with limited samples. K represents the number of splits of the data set, and evaluation is performed with a single split while the other $K-1$ are used for training. A poorly chosen value for K may lead to misrepresentation of the data, and it may result in a score with a high variance or a high bias. Empirically, values of $K = 5$ or $K = 10$ have been shown to yield close to optimal error rates, which don't suffer from excessively high bias nor very high variance [4]. $K = 5$ is chosen for this application.

Both single-fold and K-fold approaches have to be preceded by an adequate shuffling of the training data, to ensure that the distribution of samples in each split is uniformly random.

To evaluate the most promising approach, performance is measured through a normalized minimum detection cost function (*min DCF*).

Min DCF measures the cost paid upon making optimal decisions using the recognizer scores. Performances are evaluated on the validation subset. An optimal decision threshold may then be chosen.

3.1 MVG classifiers

The first classifier considered is the Multivariate Gaussian Model, or *MVG*, which assumes that the input data follows a Gaussian distribution:

$$(X|C = c) \sim \mathcal{N}(\mu_c, \Sigma_c)$$

where μ_c and Σ_c represent the mean vector and covariance matrix of the data.

Depending on assumptions made on the covariance matrix, different types of MVG may be employed:

- MVG classifier with full covariance matrices for each class;
- Naive Bayes classifier with diagonal covariance matrix, which uses the Naive Bayes assumption that features are independent among each other;
- MVG classifier with tied covariance matrices, a classifier for which every class has its own mean $\mu_{\mathbf{c}}$, but the covariance matrix $\mathbf{\Sigma}$ is equal for all classes.

The Naive Bayes assumption supposes that features are independent. While this simplifies the covariance matrix and may be optimal for some applications, in general the opposite holds true. The assumption that features are independent of each other implies that the elements outside the main diagonal of the covariance matrix are zero. This results diagonal covariance matrices. Since some of the features are highly correlated, the Naive Bayes assumption is expected to be sub-optimal for this particular application.

Instead, the MVG classifier with full covariance matrices and the MVG classifier with tied covariance matrices can capture correlations and are expected to perform better.

Results for each MVG model in terms of min DCF are reported below.

	Single-fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
Z-normalized data - **no PCA**						
Full cov	0.116	0.559	0.257	0.138	0.602	0.269
Diag cov	0.171	0.621	0.264	0.184	0.683	0.300
Tied-full cov	0.091	0.474	0.209	0.107	0.485	0.215
Tied-diag cov	0.143	0.552	0.243	0.107	0.485	0.215
Z-normalized data - **PCA (m = 7)**						
Full cov	0.115	0.523	0.264	0.134	0.561	0.281
Diag cov	0.182	0.563	0.485	0.202	0.639	0.484
Tied-full cov	0.091	0.474	0.209	0.108	0.484	0.215
Tied-diag cov	0.104	0.567	0.236	0.108	0.484	0.215
Z-normalized data - **PCA (m = 6)**						
Full cov	0.120	0.544	0.261	0.145	0.567	0.276
Diag cov	0.195	0.568	0.506	0.217	0.678	0.513
Tied-full cov	0.109	0.507	0.236	0.131	0.532	0.246
Tied-diag cov	0.137	0.563	0.250	0.131	0.532	0.246
Z-normalized data - **PCA (m = 5)**						
Full cov	0.129	0.630	0.240	0.140	0.594	0.235
Diag cov	0.197	0.595	0.425	0.209	0.686	0.440
Tied-full cov	0.124	0.517	0.237	0.140	0.526	0.250
Tied-diag cov	0.143	0.581	0.267	0.140	0.526	0.250
Z-normalized data - **PCA (m = 4)**						
Full cov	0.159	0.711	0.284	0.174	0.764	0.311
Diag cov	0.188	0.533	0.422	0.202	0.618	0.434
Tied-full cov	0.125	0.521	0.240	0.140	0.530	0.250
Tied-diag cov	0.144	0.586	0.267	0.140	0.530	0.250

Table 2: MVG Classifiers - min DCFs on the validation set

Several observations can be made by looking at the data from Table 2.

First of all, it is clear that models trained on single-fold seem to perform better.

Secondly, the best results are obtained through the MVG classifier with Tied Full Covariance matrices. It is also apparent that all the models have worse performances for the imbalanced tasks. By applying PCA with $m = 7$, similar values of min DCF to the original set of features are obtained. This means that moderate dimensionality reduction does not degrade information, as was noted in the *Correlation analysis* section. Starting from $m = 6$, results start to get worse. Therefore, only models without PCA will be considered, or PCA with $m = 7$ and $m = 6$ will be applied.

3.2 Linear Logistic Regression

Logistic regression is a discriminative approach for classification. Rather than modeling the distribution of observed samples $X|C$, the class posterior distribution $C|X$ is modeled directly.

The objective function to initially be minimized is the logistic loss function:

$$J(\boldsymbol{w}, b) = \sum_{i=1}^{n} \log(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)})$$

This function is the sum of the cost paid in classifying each sample. Therefore, it can be interpreted as an empirical risk. If classes are linearly separable, the logistic regression solution is not defined. In fact, as $||w||$ increases, the loss becomes lower, and the objective function would approach 0 if $||w|| \longrightarrow \infty$. To make the problem solvable again, in such a way that this function does not decrease indefinitely as $||w||$ increases, a regularization term may be added, which acts as a penalty for increasing $||w||$ too much. The formulation is:

$$R(\boldsymbol{w}, b) = \frac{\lambda}{2}||\boldsymbol{w}||^2 + \frac{1}{n}\sum_{i=1}^{n} \log(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)})$$

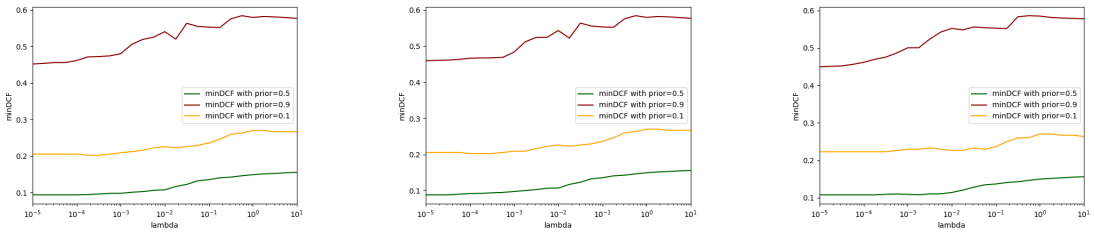
where the risk is averaged over all samples, and λ is a hyper-parameter, selected to optimize the performance of the classifier. The regularization term $\frac{\lambda}{2}||\boldsymbol{w}||^2$ helps obtaining simpler solutions in terms of lower norm of w. The hyper-parameter λ comes with a trade-off:

- if λ is too small, we may have good separation of classes, but poor generalization (over-fitting)
- if λ is too large, we may have poor separation of classes, because the solution is too simple with small norm of w and the model is too uncertain

The first step is to estimate a proper value of hyper-parameter λ. Since different empirical priors π_T are being simulated, a prior-weighted version of the objective function is optimized:

$$J(\boldsymbol{w}, b) = \frac{\lambda}{2}||\boldsymbol{w}||^2 + \frac{\pi_t}{n}\sum_{i=1|z_i=1}^{n} \log(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)}) + \frac{1-\pi_t}{n}\sum_{i=1|z_i=0}^{n} \log(1 + e^{-z_i(\boldsymbol{w}^T \boldsymbol{x_i} + b)})$$

This is the result obtained for different values of λ:



(a) Without PCA (b) With PCA ($m = 7$) (c) With PCA ($m = 6$)

Figure 3: Plots of min DCF for different values of lambda - Single fold

From Figure 3 it can be observed that the balanced applications yields far better results, as seen in the previous section. In any case, an optimal value of λ resides in the range of 10^{-5} to 10^{-3}. $\lambda = 10^{-4}$ is chosen for this application.

With the optimal λ, the Logistic Regression model can now be trained and evaluated, using single-fold and 5-fold approaches. The results for both approaches are reported in the following table.

	Single-fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
Z-normalized data - **no PCA**						
$LogReg(\lambda = 10^{-4}, \pi_T = 0.5)$	0.099	0.468	0.209	0.111	0.430	0.210
$LogReg(\lambda = 10^{-4}, \pi_T = 0.9)$	0.102	0.463	0.203	0.113	0.457	0.209
$LogReg(\lambda = 10^{-4}, \pi_T = 0.1)$	0.095	0.486	0.199	0.108	0.485	0.204
Z-normalized data - **PCA** $(m = 7)$						
$LogReg(\lambda = 10^{-4}, \pi_T = 0.5)$	0.095	0.477	0.206	0.109	0.453	0.210
$LogReg(\lambda = 10^{-4}, \pi_T = 0.9)$	0.101	0.477	0.206	0.112	0.461	0.211
$LogReg(\lambda = 10^{-4}, \pi_T = 0.1)$	0.098	0.482	0.199	0.108	0.483	0.203
Z-normalized data - **PCA** $(m = 6)$						
$LogReg(\lambda = 10^{-4}, \pi_T = 0.5)$	0.114	0.456	0.223	0.119	0.500	0.231
$LogReg(\lambda = 10^{-4}, \pi_T = 0.9)$	0.122	0.494	0.223	0.128	0.488	0.233
$LogReg(\lambda = 10^{-4}, \pi_T = 0.1)$	0.116	0.465	0.230	0.121	0.516	0.231

Table 3: Linear Logistic Regression Classifiers - min DCFs on the validation set

From the results in Table 4, these implementations give the best results:

- without PCA and $\pi_T = 0.1$
- with PCA $(m = 7)$ and $\pi_T = 0.5$

It is interesting to see that re-balancing through $\pi_T = 0.1$ gives a very similar result to not re-balancing. As already seen in MVG, the results obtained with $m = 6$ start to get worse. From now on, only data without PCA or PCA $(m = 7)$ is to be considered.

As seen in the MVG classifier section, single-fold performs better than 5-fold. Also, single-fold is much faster, because in the 5-fold approach the models have to be re-trained for each fold.

The two most promising models so far are reported in the following table:

	Single-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
Z-normalized data - **PCA (m=7)**			
MVG Tied-full covariance	0.091	0.474	0.209
$LogReg(\lambda = 10^{-4}, \pi_T = 0.5)$	0.095	0.477	0.206

Table 4: Comparing models

It can be noticed that both MVG Tied (with full covariance matrix) and Linear Logistic Regression are linear models. The most promising models are linear, and they will act as a reference when comparing with next models.

3.3 Support Vector Machines

SVMs models start from this fact: when classes are linearly separable there is an infinite number of separating hyperplanes. For this reason, in the Linear Logistic Regression model the regularization term $\frac{\lambda}{2}||\boldsymbol{w}||^2$ has been added, which helps in solving this problem.

SVMs add another assumption in finding an hyperplane: the hyperplane that separates the classes with the largest margin is to be used.

The (primal) SVM objective function is the minimization of:

$$J(\boldsymbol{w}, b) = \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} max(0, 1 - z_i(\boldsymbol{w^T x_i} + b))$$

where n is the number of training samples, C is a hyper parameter, z_i is the class label for the i-th sample

$$\begin{cases} +1 & \text{if } x_i \text{ belongs to class } \mathcal{H}_T \\ -1 & \text{if } x_i \text{ belongs to class } \mathcal{H}_F \end{cases}$$

and $\boldsymbol{H}$ is a matrix whose elements are

$$\boldsymbol{H}_{i,j} = z_i z_j \boldsymbol{x_i^T x_j}$$

To solve the SVM problem, the dual formulation may be considered as well:

$$J^D(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha^T H \alpha} + \boldsymbol{\alpha^T 1}$$

$$\text{subject to } 0 \leq \alpha_i \leq C_i, i = \{1...n\}, \textstyle\sum_{i=1}^{n} \boldsymbol{\alpha_i} z_i = 0$$

The second constraint is a bias constraint, but the SVM problem is slightly changed as to make this constraint disappear. This is done because the L-BFGS algorithm employed is not able to incorporate this particular constraint.

The primal problem can be reformulated as the minimization of:

$$\widehat{J}^D(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha^T}\widehat{\boldsymbol{H}}\boldsymbol{\alpha} + \boldsymbol{\alpha^T 1}$$

$$\text{subject to } 0 \leq \alpha_i \leq C_i, i = \{1...n\}$$

where $\widehat{x_i} = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$, $\widehat{\boldsymbol{w}} = \begin{bmatrix} \boldsymbol{w} \\ b \end{bmatrix}$ and the matrix $\widehat{\boldsymbol{H}}$ is modified accordingly:

$$\widehat{\boldsymbol{H}}_{i,j} = z_i z_j (\widehat{\boldsymbol{x}}_i^T \widehat{\boldsymbol{x}}_j + 1)$$

A good value for the hyper-parameter C has to be chosen. SVMs are large margin classifiers, as it is optimal to select an hyperplane that separates the classes with a certain margin. The hyper-parameter C allows for "soft margin classification", thereby allowing some samples to be inside the margin.

The "soft margin version" of the problem was the following:

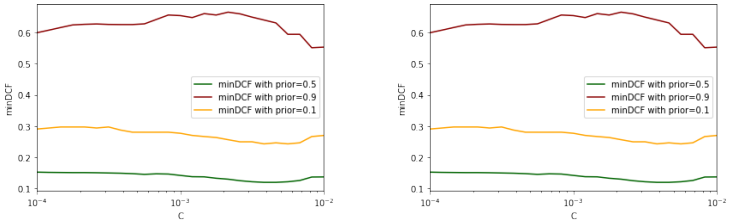
$$\text{minimize in } (\boldsymbol{w}, b)\ \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{i=1}^{n} \epsilon_i$$

$$\text{subject to } z_i(\boldsymbol{w}^T x_i + b) \geq 1 - \epsilon_i, \epsilon_i \geq 0$$

The term $\sum_{i=1}^{n} \epsilon_i$ is an upper bound on the number of mis-classified points (errors). The hyper-parameter C adds penalty to each mis-classified point and represents a trade-off between decision boundary and mis-classification terms.

- Small C means the penalty part (the right part) plays little role. We do not need the slack variables to be very small for the minimization process. Therefore, there is more tolerance of mis-classification. The margin will be wide.
- Large C means that the penalty parts plays an important role. The slack variables have to be as small as possible. Therefore, mis-classification is not tolerated. The margin will be narrow.

Firstly, a SVM model that does not balance the two classes is considered.



(a) Without PCA (b) With PCA ($m = 7$)

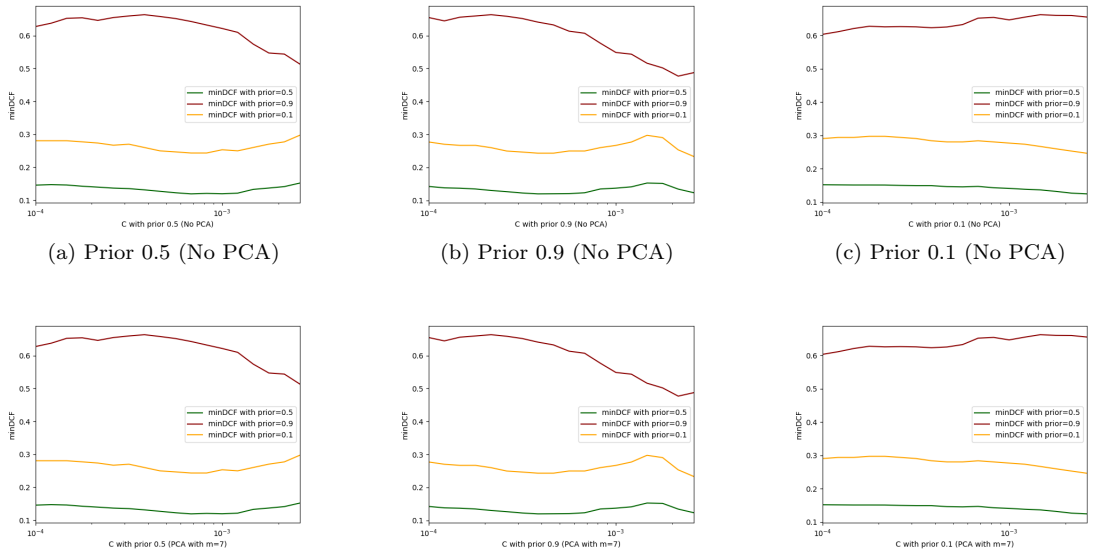
Figure 4: Unbalanced SVM - Plots of min DCF for different values of C - Single fold

From Figure 4, for the main application $C = 5 \cdot 10^{-3}$ is chosen, which seems to provide a lower min DCF.

Secondly, a balanced version is to be considered. In the balanced SVM the value of C is changed for the different classes.

$$\begin{cases} C_T = C \frac{\pi_T}{\pi_T^{emp}} \text{ for class true} \\ C_F = C \frac{\pi_F}{\pi_F^{emp}} \text{ for class false} \end{cases}$$

The values of C for balanced classes are again estimated by cross-validation.



(a) Prior 0.5 (No PCA) (b) Prior 0.9 (No PCA) (c) Prior 0.1 (No PCA)

(d) Prior 0.5 (PCA with $m = 7$) (e) Prior 0.9 (PCA with $m = 7$) (f) Prior 0.1 (PCA with $m = 7$)

Figure 5: Balanced SVM - Plots of min DCF for different values of C - Single fold

From the data in Figure 5, $C = 5 \cdot 10^{-4}$ is chosen.
Results for each SVM model in terms of min DCF are reported below

	Single-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
	Z-normalized data - no PCA		
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.121	0.642	0.243
Linear SVM ($C = 5 \cdot 10^{-4}$, $\pi_T = 0.5$)	0.131	0.653	0.260
Linear SVM ($C = 5 \cdot 10^{-4}$, $\pi_T = 0.9$)	0.121	0.642	0.243
Linear SVM ($C = 5 \cdot 10^{-4}$, $\pi_T = 0.1$)	0.147	0.633	0.290
	Z-normalized data - PCA (m = 7)		
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.121	0.642	0.243
Linear SVM ($C = 5 \cdot 10^{-4}$, $\pi_T = 0.5$)	0.131	0.654	0.257
Linear SVM ($C = 5 \cdot 10^{-4}$, $\pi_T = 0.9$)	0.121	0.642	0.243
Linear SVM ($C = 5 \cdot 10^{-4}$, $\pi_T = 0.1$)	0.147	0.633	0.291

Table 5: Linear SVM models, unbalanced and balanced - min DCFs on the validation set

Table 5 shows that there is no improvement in re-balancing. Therefore, only the unbalanced version of SVM model will be considered from now on.

SVMs with non-linear classification will also be discussed. SVM can be extended to solve nonlinear classification tasks when the set of samples cannot be separated linearly.

There are two main steps:

- In the first step, the original input data is transformed into a higher dimensional space using a non-linear mapping $\boldsymbol{x} \longrightarrow \Phi(\boldsymbol{x})$. Several common nonlinear mappings can be used in this step, as will be further described next.
- Once the data have been transformed into the new higher space, the second step searches for a linear separating hyperplane in the new space. This problem can be solved using the linear SVM formulation.

The maximal marginal hyperplane found in the new space corresponds to a non-linear separating hypersurface in the original space.

So far, linear models achieved better performances than non-linear models. This hypothesis would be proved by showing that non-linear SVM is sub-optimal.

The dual formulation of the SVM problem depends on the dot product between samples x_i and x_j, because of the matrix H

$$\widehat{\boldsymbol{H}}_{i,j} = z_i z_j (\widehat{\boldsymbol{x}}_i^T \widehat{\boldsymbol{x}}_j + 1)$$

By transforming the samples with the non-linear mapping $\boldsymbol{x} \longrightarrow \Phi(\boldsymbol{x})$, the dot products between non-linear mappings will have to be computed. Having to compute $\Phi(\boldsymbol{x_i})^T \Phi(\boldsymbol{x_j})$ can be computationally expensive. Instead of computing the dot product on the transformed data tuples, it turns out that it is mathematically equivalent to instead apply a *kernel function* $\boldsymbol{K(x_i, x_j)}$ that efficiently computes dot products, which can then be used for training and scoring.

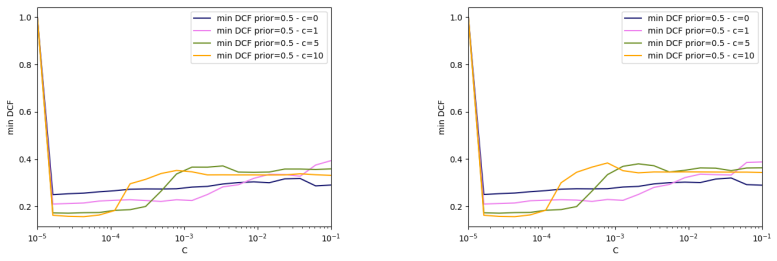
$$\boldsymbol{K(x_i, x_j)} = \Phi(\boldsymbol{x_i})^T \Phi(\boldsymbol{x_j})$$

In other words, $\Phi(\boldsymbol{x_i})^T \Phi(\boldsymbol{x_j})$ can be replaced with $\boldsymbol{K(x_i, x_j)}$. The calculations are simplified because the samples x_i and x_j are considered in the original feature space, which is of lower dimensionality.

Two different kernels will be employed:

- Polynomial kernel of degree $d = 2$ - $\boldsymbol{K(x_i, x_j)} = (\boldsymbol{x_i^T x_j} + c)^d$
- Radial Basis Function (RBF) kernel - $\boldsymbol{K(x_i, x_j)} = e^{-\gamma ||\boldsymbol{x_i} - \boldsymbol{x_j}||^2}$

In particular, polynomial kernel of of degree $d = 2$ is of note. Parameters C and c have to be estimated.



(a) Without PCA (b) With PCA ($m = 7$)

Figure 6: SVM with Polynomial kernel - Plots of min DCF for different values of C - Single fold

From this analysis it can be concluded that the best values for applying the polynomial kernel SVM are $C = 5 \cdot 10^{-5}$ and $c = 10$. Results are summarized below:

	Single-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
	Z-normalized data - no PCA		
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.121	0.642	0.243
Polynomial SVM ($C = 3 \cdot 10^{-5}, c = 10$)	0.152	0.583	0.280
	Z-normalized data - PCA (m = 7)		
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.121	0.642	0.243
Polynomial SVM ($C = 3 \cdot 10^{-5}, c = 10$)	0.152	0.583	0.280

Table 6: Polynomial SVM models compared to Linear SVM models - min DCFs on the validation set

As predicted, in the Table 6 it is shown that for the main application $\tilde{\pi} = 0.5$ the linear model performs better than the quadratic model.

Here is a comparison of the models seen so far:

	Single-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
	Z-normalized data - **PCA (m=7)**		
MVG Tied-full cov	0.091	0.474	0.209
$LogReg(\lambda = 10^{-4}, \pi_T = 0.5)$	0.095	0.477	0.206
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.121	0.642	0.243

Table 7: Comparing models

As shown in Table 7, linear SVM is not very effective if compared with previously analyzed models.

3.4 Gaussian Mixture Models

The last category of models analyzed for this application are Gaussian Mixture Models, or *GMM* for short. GMMs are based on the assumption that a set of data, in general, cannot be modeled by a single Gaussian distribution. However, if the data is sufficiently regular, a Gaussian mixture can be fitted over it with a certain degree of precision. That is, a data set can be split into subsets, called *components* or *clusters*, such that the distribution of samples in each cluster can be modeled by a Gaussian distibution.

The resulting Gaussian mixture is a weighted sum of Gaussian densities:

$$\boldsymbol{X} \sim GMM(\boldsymbol{M}, \boldsymbol{\Sigma}, \boldsymbol{\Pi}) \implies f_{\boldsymbol{X}}(\boldsymbol{x}) = \sum_{c=1}^{K} \pi_c \mathcal{N}(\boldsymbol{x}, \boldsymbol{\mu_c}, \boldsymbol{\Sigma_c})$$

where π_c is the probability that Gaussian component c has generated a certain sample x. A Gaussian mixture distribution is characterized by the means of its components $\boldsymbol{M} = [\mu_1 \dots \mu_K]$, their

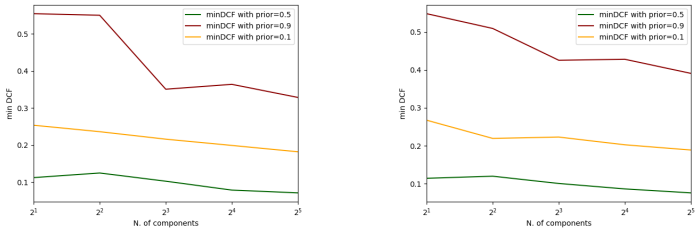
covariances $\boldsymbol{\Sigma} = [\Sigma_1 \dots \Sigma_K]$ and the weights $\boldsymbol{\Pi} = [\pi_1 \dots \pi_K]$. Naturally, for $f(\boldsymbol{x})$ to be a density its integral must equal one. Therefore, the weights, being probabilities, must satisfy the condition:

$$\sum_{c=1}^{K} \pi_c = 1$$

The Gaussian mixture problem is essentially a density estimation problem, therefore Maximum Likelihood estimates may be used to find the Gaussian components. However, the ML estimates lack an upper bound, and they can present the risk of degenerate solutions. Therefore, they are to be used in conjunction with heuristic techniques to avoid such solutions.

Since clusters are initially unknown, cluster membership is treated as an unknown random variable. Starting from an initial set of parameters, and through successive iterations, clusters can be computed either by K-means or, in a more general way, with the Expectation-Maximization (EM) algorithm. Unlike K-means, EM algorithm assigns probabilities of belonging to a certain cluster to each sample. This is called *soft* clustering, as opposed to K-means's *hard* clustering (in which each sample may belong to only one cluster).

The starting set of parameters for EM is quite important, so the LBG algorithm is used to provide a good distribution for the data. The LBG algorithm consists in splitting an initial Gaussian distribution along the direction of maximum spread, and applying EM to the resulting two Gaussian distributions. The process is then repeated until the desired number of Gaussian components is reached. Of course, the number of desired components is unknown at first. A cross-validation approach is used to determine the optimal number of components for the distribution.



(a) GMM Full covariance model - without PCA

(b) GMM Full covariance model - PCA with m = 7

Figure 7: min DCF for different numbers of GMM components

For the target application, the full covariance model with 2^4 components is chosen, which seems to give good results without too much computational time cost.

GMM is here compared with the other models seen so far:

	Single-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
	Z-normalized data - **PCA (m=7)**		
MVG Tied-full cov	0.091	0.474	0.209
$LogReg(\lambda = 10^{-4}, \pi_T = 0.5)$	0.095	0.477	0.206
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.121	0.642	0.243
GMM Full cov (16 components)	0.110	0.548	0.206

Table 8: Comparing models

For the target application, Linear Logistic Regression and MVG Tied with full covariance seem to perform best.

3.5 Score calibration

Up to now only min DCF metrics have been considered. Min DCF measures the cost paid if optimal decisions were made for the evaluation set using the recognizer scores. The actual cost, however, depends on the goodness of the threshold used to perform class assignment.

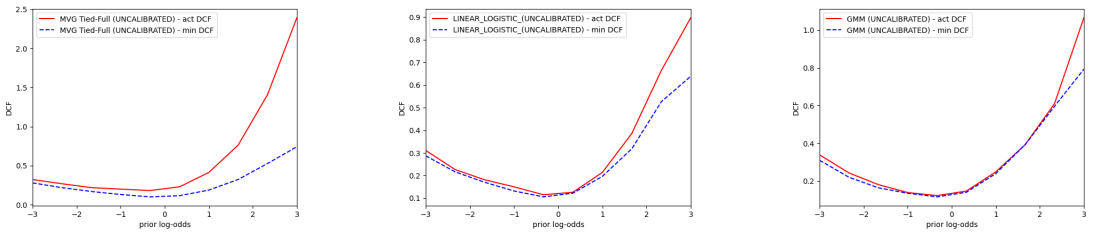
Actual DCFs will now be considered. If scores are well calibrated, the optimal threshold that optimizes the Bayes risk is the theoretical threshold $t = -\log \frac{\tilde{\pi}_t}{1-\tilde{\pi}_t}$.

Actual DCF have to be evaluated to assess how good the models would be if the theoretical threshold were used for each application. If the min DCF and the actual DCF are similar, it means that the model is well-calibrated.

	Single-fold					
	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$	$\pi = 0.5$	$\pi = 0.9$	$\pi = 0.1$
	min DCF - PCA ($m = 7$)			actual DCF - PCA ($m = 7$)		
MVG Tied-Full covariance	0.091	0.474	0.209	0.172	1.260	0.254
LogReg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.095	0.477	0.206	0.099	0.332	0.156
GMM Full covariance (16 components)	0.110	0.548	0.206	0.115	0.587	0.230

Table 9: min DCFs and actual DCFs

The differences among models may also be plotted with a Bayes error plot, as can be seen in the following figure.



(a) MVG Tied-Full (b) Linear Logistic Regression (c) GMM with 16 components

Figure 8: Bayes error plots for the three best models - without calibration

Figure 9 shows that the GMM seems to be already well-calibrated by default, while MVG and Logistic Regression seem to have some offset from the actual DCF.

Score calibration may now be employed on MVG and Linear Logistic Regression. The approach adopted here consists in transforming the scores so that the theoretical threshold $t = -\log \frac{\tilde{\pi}}{1-\tilde{\pi}}$ provides close to optimal values over a wide range of applications. This is done by computing a transformation function f that maps the scores s of each classifier to well-calibrated scores $s_{cal} = f(s)$.

Assume that the function f is linear in s:

$$f(s) = \alpha s + \beta$$

The function $f(s)$ maps scores to well-calibrated scores, which means that it can be interpreted as the log-likelihood ratio for the two class hypotheses:

$$f(s) = \frac{\log f_{S|C}(s|H_T)}{\log f_{S|C}(s|H_F)} = \alpha s + \beta$$

The class posterior probability for a prior $\tilde{\pi}$ can be written as:

$$f(s) = \frac{P(C = H_T|s)}{P(C = H_F|s)} = \alpha s + \beta + \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

The scores s can be interpreted as features. Therefore, each sample will be described by a single score (1-dimensional data). By rewriting:

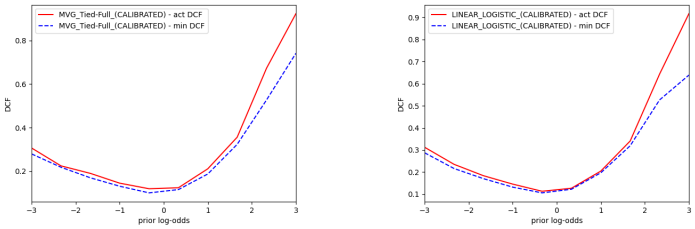
$$\beta' = \beta + \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

a linear expression in s is obtained. This expression is very similar to the log posterior ratio of the Logistic Regression model. Therefore, the prior-weighted Logistic Regression model will be employed to learn the parameters α, β' over the training scores.

To recover the calibrated scores $f(s)$ it is necessary to compute:

$$f(s) = \alpha s + \beta' - \log \frac{\tilde{\pi}}{1-\tilde{\pi}}$$

The main application with prior $\tilde{\pi} = 0.5$ is now being considered. This is the result of score calibration applied to MVG and Linear Logistic Regression:



(a) MVG Tied-Full (b) Linear Logistic Regression

Figure 9: Bayes error plots for MVG and Linear Logistic Regression - after calibration

Figure 9 shows how score calibration has proved to be effective in producing well-calibrated score for the MVG and Linear Regression models.

4 Test results and conclusions

As a final step, performance of the best models is evaluated by training them over the whole training set, and testing them over the whole test set. They are then compared in terms of Min DCF. The results are shown in the table below.

min DCF over Z-normalized data (PCA: $m = 7$)			
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.1$
MVG Full Covariance	0.139	0.569	0.292
MVG Diag Covariance	0.196	0.765	0.504
MVG Tied-Full Covariance	0.109	0.580	0.203
MVG Tied-Diag Covariance	0.139	0.547	0.257
LogReg($\lambda = 10^{-4}, \pi_T = 0.5$)	0.109	0.545	0.198
Linear SVM ($C = 5 \cdot 10^{-3}$, unbalanced)	0.147	0.599	0.286
Polynomial SVM ($C = 5 \cdot 10^{-5}, c = 10$)	0.183	0.607	0.348
Gaussian Mixture Model - 16 Gau	0.125	0.585	0.252

Table 10: Comparison of all models trained over the whole training set and tested over the test set.

The Linear Logistic Regression model with $\lambda = 10^{-4}$ and calibrated with $\pi_T = 0.5$ proves to be the most effective and most versatile model, with the best scores for all applications. For the main application, the multivariate Gaussian with tied full covariance matrices ties with logistic regression for effectiveness.

The Gaussian Mixture Model with sixteen components follows in terms of effectiveness for the main application.

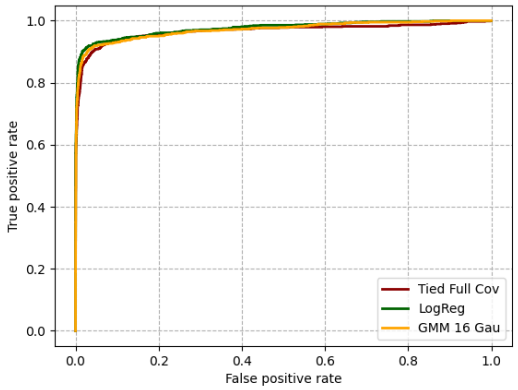


Figure 10: ROC plot of the three best models for the balanced application.

As seen in the ROC plot in Figure 10, all three models perform very similarly over the test set. False positives are kept to a minimum, and Logistic Regression has a very slightly higher AUC (area under curve).

In terms of min DCF, a score of 0.109 is achieved by the logistic regression, which is very good for a balanced application. However, effectiveness of the model quickly decreases for unbalanced applications. The scores are quite similar to the ones obtained from single-fold and 5-fold cross validation (where performed), proving that the choices made over the training set were representative of the final test set.

References

[1] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, *Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach*, MNRAS, 2016.

[2] R. J. Lyon, *"PulsarFeatureLab"*, 2015,
https://dx.doi.org/10.6084/m9.figshare.1536472.v1.

[3] Nora Roberts, D. R. Lorimer, M. Kramer, *Handbook of Pulsar Astronomy (illustrated, herdruk ed.)*, Cambridge University Press, 2015.

[4] Gareth James et al., *An Introduction to Statistical Learning*, Springer, 2013