Classification of HTRU2 dataset

Francesca Russo, Claudio Tancredi
s287935@studenti.polito.it, s292523@studenti.polito.it

July 6, 2021

Abstract

This work brings together some of the most common Machine Learning (ML) algorithms, and the goal is to make a comparison of the results obtained from the HTRU2 dataset. This dataset is composed of almost 18 thousand observations made to astronomical objects to identify pulsars. Our work will start from feature analysis to understand data distribution and features correlation, then it will move to the analysis of different classifiers followed by score calibration. The results will lead us to choose linear models which will prove to be promising solutions for the proposed classification task.

1 Introduction

HTRU2[1] is a dataset which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South)[2]. Pulsars are a rare type of Neutron star that produce radio emission detectable here on Earth. They are of considerable scientific interest as probes of space-time, the interstellar medium, and states of matter.
As pulsars rotate, their emission beam sweeps across the sky, and when this crosses our line of sight, produces a detectable pattern of broadband radio emission. As pulsars rotate rapidly, this pattern repeats periodically. Thus pulsar search involves looking for periodic radio signals with large radio telescopes.
Each pulsar produces a slightly different emission pattern, which varies slightly with each rotation. Thus a potential signal detection known as a 'candidate', is averaged over many rotations of the pulsar, as determined by the length of an observation. In the absence of additional info, each candidate could potentially describe a real pulsar. However in practice almost all detections are caused by radio frequency interference (RFI) and noise, making legitimate signals hard to find.
Machine Learning tools are now being used to automatically label pulsar candidates to facilitate rapid analysis. Classification systems in particular are being widely adopted, which treat the candidate data sets as binary classification problems. Here the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class.
The dataset contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. These examples have all been checked by human annotators.
The dataset has been split into Train and Evaluation (Test) data.
The training set is stored in the file "Train.txt" and contains 8108 false pulsar signal and 821 true pulsar signal. The evaluation set is stored in the file "Test.txt" and contains 8151 false pulsar signal and 818 true pulsar signal.
Candidates are stored in both files in separate rows. Each row lists the variables first, and the class label is the final entry. The class labels used are 0 (false pulsar signal) and 1 (true pulsar signal).
Our goals will be to perform classification for the proposed task by employing different models, to analyse results and to motivate our choices.

2 HTRU2 features analysis

Each candidate is described by 8 continuous variables, and a single class variable. The features are summarised below:

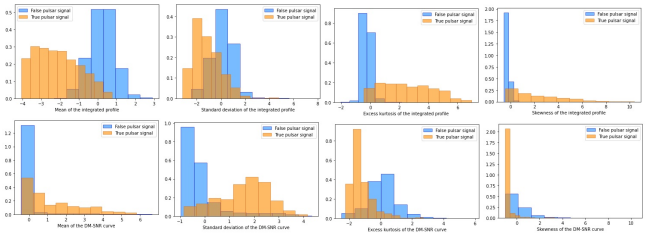
1. Mean of the integrated profile.
2. Standard deviation of the integrated profile.
3. Excess kurtosis of the integrated profile.
4. Skewness of the integrated profile.
5. Mean of the DM-SNR curve.
6. Standard deviation of the DM-SNR curve.
7. Excess kurtosis of the DM-SNR curve.
8. Skewness of the DM-SNR curve.

Since dataset features values turned out to be high, to simplify computations and to avoid overflow problems that may arise (especially when working with exponentials) data has been pre-processed by means of Z-normalization:

$$\boldsymbol{z}_i = \frac{\boldsymbol{x}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \ \forall i \in \{1, ..., n\}$$

where $\boldsymbol{x}_i$ is the observed value for the i-th sample, $\boldsymbol{\mu}$ is the mean vector of the samples computed along columns and $\boldsymbol{\sigma}$ is the standard deviation vector of the samples computed along columns.
This procedure allows centering and scaling data so that the Z-normalized R.V. Z will have a standard distribution with zero-mean and one-variance.

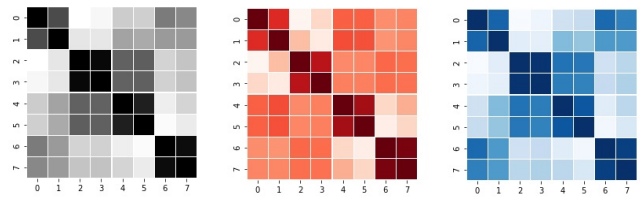
Figure 1: *Histograms of the Z-normalized HTRU2 training set features*



Z-normalized features shown in Figure 1 seem already well distributed, with no evident outliers. For this reason, no further pre-processing has been considered.
Furthermore, heatmaps of the Z-normalized training set features have been plotted to analyse features correlation. The heatmaps show the Pearson correlation coefficient:

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

Figure 2: *Gray: whole Z-normalized training set. Red: false pulsar signal samples. Blue: true pulsar signal samples. Darker color implies larger value for the Pearson correlation coefficient*



It is evident from the heatmaps in Figure 2 that features 2-3 and 6-7 are strongly correlated, while features 4-5 are moderately correlated.
Dimensionality reduction techniques could therefore prove to be beneficial.
Specifically, PCA is a dimensionality reduction technique that computes a linear mapping from the n-dimensional feature space to a m-dimensional space, with $m \ll n$, while preserving the directions with highest variance.
We'll exploit PCA to reduce the training set dimensionality up to 5 and see how models behave.
We expect that dimensionality reduction will not cause major losses of information and at the same time it will allow us to reduce the number of parameters to estimate.

3 Classifying HTRU2 features

In order to perform our analysis, we will adopt two approaches. In the first one we will split the training set into training and validation subsets (this approach is called single-fold in the following), but we will also exploit K-fold cross-validation approach. The single-fold consists of 66.6% training data and 33.3% validation data. The K-fold is implemented with K=3. In both cases data has been shuffled before splitting.
We will consider three different applications: a uniform prior application and two unbalanced applications where the prior is biased towards one of the two classes:

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

Our target application will be the balanced one.
We want to find the most promising approach, so we can measure performances through the metric of the normalized minimum Detection Cost Function (min DCF), which measures the cost that we would pay if we made optimal decisions using the recognizer scores. We evaluate performances on the validation subset (extracted from the training set).

3.1 Gaussian classifiers

We start considering Gaussian classifiers (MVG classifier, MVG classifier with Naive Bayes assumption, MVG classifier with tied covariance). All of them assume gaussian distributed data, given the class:

$$\boldsymbol{X}|C = c \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$$

The MVG classifier with tied covariance matrices assumes that each class has its own mean $\boldsymbol{\mu}_c$, but the covariance matrix $\boldsymbol{\Sigma}$ is the same for all classes.
The Naive Bayes Gaussian classifier corresponds to a MVG classifier with diagonal covariance matrices for the different classes. The Naive Bayes assumption supposes that features are independently distributed and, if this is the case, then the features are also uncorrelated and will have zero-covariances, which are the elements off-diagonal on the covariance matrices and this justify the diagonalization. Since some of our features are highly correlated, we expect covariances off-diagonal not to be approximately zero, so we have no guarantee that the Naive Bayes assumption will actually work out well, we may get not-optimal results.
Instead, the MVG classifier with full covariance matrices and the MVG classifier with tied covariance matrices are able to capture correlations and may output better results, also considering the sufficient number of samples at our disposal.

Figure 3: *MVG Classifiers — min DCFs on the validation set*

	Single-fold			3-folds		
	$\tilde{\pi}$ = 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	0.142	0.662	0.285
Diag-Cov	0.180	0.592	0.455	0.213	0.728	0.476
Tied-Cov	0.091	0.474	0.209	0.112	0.573	0.224
Z-normalized data - PCA (m=7)						
Full-Cov	0.115	0.523	0.264	0.139	0.631	0.302
Diag-Cov	0.182	0.563	0.485	0.214	0.721	0.504
Tied-Cov	0.091	0.474	0.209	0.112	0.569	0.224
Z-normalized data - PCA (m=6)						
Full-Cov	0.120	0.544	0.261	0.151	0.632	0.286
Diag-Cov	0.195	0.568	0.506	0.222	0.722	0.531
Tied-Cov	0.109	0.507	0.236	0.138	0.578	0.256
Z-normalized data - PCA (m=5)						
Full-Cov	0.129	0.630	0.240	0.149	0.636	0.250
Diag-Cov	0.197	0.595	0.425	0.219	0.742	0.459
Tied-Cov	0.124	0.517	0.237	0.149	0.574	0.261

By comparing results in Figure 3 from single-fold and 3-folds we can notice that models trained on the single-fold seem to perform better, but with both approaches values of the minDCF are consistent.
Therefore, we will continue to analyse both approaches but we will rely more on the 3-folds approach, which should prove more robust and will actually allow us to re-train the models over the whole dataset.
Overall, the best results are obtained through the gaussian classifier with tied covariance matrices.
It is also clear that all the models have worse performances for the imbalanced tasks.
By applying PCA with m = 7 we obtain almost the same values for the min DCFs that we get by employing the models without PCA. This is evidence that dimensionality reduction can be exploited without loss of useful information, as forecasted from the correlation analysis. Instead, starting from m = 6, results get a little bit worse, so in the following we'll mainly refer to m = 7 and models without PCA but we'll also consider m = 6 to better understand if our hypothesis holds.
We now turn our attention to discriminative models.

3.2 Linear Logistic Regression

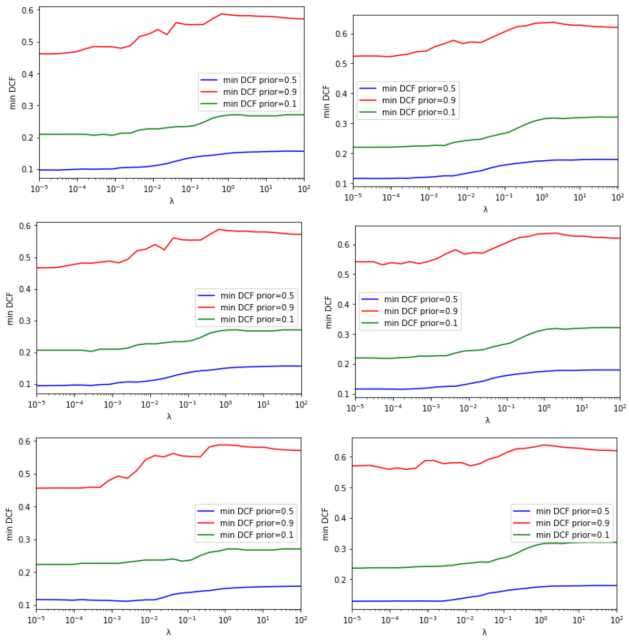
We start considering regularized Linear Logistic Regression. Since classes are unbalanced, we change the objective function that we need to minimize so that costs of the different classes are re-balanced:

$$J(\boldsymbol{w},b) = \frac{\lambda}{2}\|\boldsymbol{w}\|^2 + \frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n}\log\left(1+e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}\right) + \frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n}\log\left(1+e^{-z_i(\boldsymbol{w}^T\boldsymbol{x}_i+b)}\right)$$

The model parameters are (**w**, b). The model assumes that decision rules are linear hyperplanes orthogonal to **w**.

We also exploit 3-folds cross-validation to estimate a proper value for the hyper-parameter λ.

Figure 4: *Linear Log-Reg: min DCFs for our target application using $\pi_T = \frac{1}{2}$ for different values of λ. Top: no PCA. Center: PCA with m = 7. Bottom: PCA with m = 6. Left: single fold. Right: 3-folds.*



Results with 3-folds and single-fold from Figure 4 are fairly similar, at least for our target application. This is additional proof that we may use the K-fold version, whose evaluation results should be more reliable.
The regularization term $\frac{\lambda}{2}\|\mathbf{w}\|^2$ helps obtaining simpler solutions in terms of lower norm of **w**. This will lead to not so certain models, helping in reducing over-fitting of the training data.
The hyper-parameter λ allows for a tradeoff between:

1. poor separation of classes and simpler solutions with small norm of **w**, when $\lambda \gg 0$

2. good separation of classes but poor generalization on unseen data, when $\lambda \simeq 0$

Regularization with $\lambda \gg 0$ seems to provide no benefit, so $\lambda = 10^{-4}$ could possibly be a good choice.
Now we can also consider different priors π_T to see how the model behaves. We restrict the analysis to the 3-folds approach.

Figure 5: *Linear Log-Reg: min DCFs computed using the estimated hyper-parameter λ*

	3-folds		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
	Z-normalized data - no PCA		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.116	0.523	0.221
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.123	0.520	0.223
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.115	0.549	0.214
	Z-normalized data - PCA (m=7)		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.538	0.219
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.120	0.538	0.220
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.116	0.550	0.215
	Z-normalized data - PCA (m=6)		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.129	0.558	0.238
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.139	0.552	0.238
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.129	0.600	0.244

Overall, as we can see from the results in Figure 5, the implementation with m = 7, $\pi_T = 0.5$ seems to provide the best results. Re-balancing the costs of different classes through π_T has proven to output very similar min DCFs.
As seen in MVG and then in linear Logistic Regression, the results obtained on data reduced through PCA with m = 6 are slightly worse. For this reason, from now on we won't consider this particular dimensionality reduction anymore.

Figure 6: *Best candidate models up to now*

	3-folds		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
	Z-normalized data - PCA (m=7)		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.538	0.219
Tied-Cov	0.112	0.569	0.224

Since the MVG Tied-Cov and the linear Logistic Regression, which are linear models, perform better than the MVG Full-Cov, which is a quadratic model, we may start to assume that linear models perform better than quadratic models on this dataset. We will further test our assumption later on, by employing other quadratic models (e.g. quadratic SVM).

3.3 SVMs

We now turn our attention to SVMs. We will consider the linear SVM, the polynomial quadratic kernel and the Radial Basis Function kernel formulations, altough from the previous results we expect the linear SVM to perform better.

We start considering a linear model that does not balance the two classes. To solve the SVM problem, we can consider the dual formulation:

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

$$subject\ to\ 0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}, \sum_{i=1}^{n} \alpha_i z_i = 0$$

where n is the number of training samples, C is a hyper-parameter, $\mathbf{1}$ is a n-dimensional vector of ones, z_i is the class label for the i-th sample encoded as

$$z_i = \begin{cases} +1, \ if\ x_i\ belongs\ to\ class\ 1\ (true\ pulsar\ signal) \\ -1, \ if\ x_i\ belongs\ to\ class\ 0\ (false\ pulsar\ signal) \end{cases}$$

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

$$\boldsymbol{H}_{i,j} = z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

From the dual formulation we then derive the primal solution.
Due to the fact that we use the L-BFGS-B algorithm and it is only able to handle box contraints, we have to modify the formulation to already include the second constraint. We will have that the modified dual formulation corresponds to

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

$$subject\ to\ 0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}$$

and, since we use the mapping

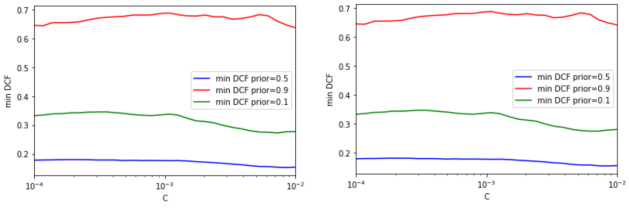
$$\widehat{\boldsymbol{x}}_i = \begin{bmatrix} \boldsymbol{x}_i \\ K \end{bmatrix}$$

with K=1, the matrix $\widehat{\boldsymbol{H}}$ is modified accordingly:

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

We resort to 3-folds cross-validation approach to choose a good value for the hyper-parameter C.

Figure 7: *Linear SVM: 3-folds cross-validation. Left: no PCA. Right: PCA with m = 7*

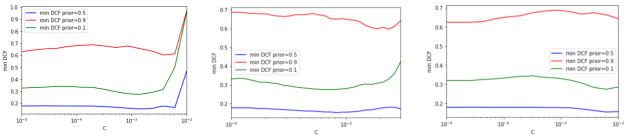


Given the analysis of Figure 7, we choose $C = 10^{-2}$ since it seems to provide a lower min DCF.
We can also consider the balanced version. Balancing is done by considering a different value of C for the different classes in the box constraint of the dual formulation:

$$0 \leq \alpha_i \leq C_i, \forall i \in \{1, ..., n\}$$

where $C_i = C_T$ for samples of the True class and $C_i = C_F$ for samples of the False class. We select $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$ where π_T^{emp} and π_F^{emp} are the empirical priors.

Figure 8: *Linear SVM: 3-folds cross-validation, no PCA. Left:* $\pi_T = 0.5$. *Middle:* $\pi_T = 0.9$. *Right:* $\pi_T = 0.1$



The table in Figure 9 summarizes the obtained results and the choices for the hyper-parameter C.

Figure 9: *Linear SVM: 3-folds cross validation. min DCFs computed using the estimated hyper-parameter C*

	3-folds		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
	Z-normalized data - no PCA		
Linear SVM ($C=10^{-2}$, *unbalanced*)	0.154	0.683	0.277
Linear SVM ($C=10^{-3}$, π_T=0.9)	0.154	0.651	0.278
Linear SVM ($C=10^{-3}$, π_T=0.5)	0.157	0.680	0.276
Linear SVM ($C=5*10^{-3}$, π_T=0.1)	0.157	0.680	0.276
	Z-normalized data - PCA (m=7)		
Linear SVM ($C=10^{-2}$, *unbalanced*)	0.154	0.642	0.280
Linear SVM ($C=10^{-3}$, π_T=0.9)	0.153	0.652	0.280
Linear SVM ($C=10^{-3}$, π_T=0.5)	0.159	0.681	0.276
Linear SVM ($C=5*10^{-3}$, π_T=0.1)	0.159	0.681	0.276

We can observe that there is no need for class rebalancing, so we will mainly consider the unbalanced version of the linear SVM model.
We now analyse the non-linear formulations.
In SVM, the non-linearity is obtained through an implicit expansion of the features in a higher dimensional space. The dual SVM formulation, as seen in (1), depends on the training samples through the dot-product and it's possible to compute scores through scalar products between training and evaluation samples. For this reason, it's not required to explicitly compute the feature expansion, it's enough to be able to compute the scalar product between the expanded features, the so called *kernel function* k:

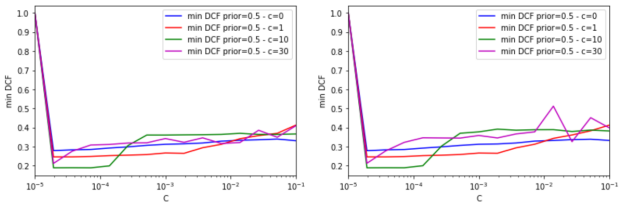
$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}_j)$$

We will employ two different kernels:

1. Polynomial kernel of degree d = 2: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i^T \boldsymbol{x}_j + c)^d$
2. Radial Basis Function (RBF) kernel: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$

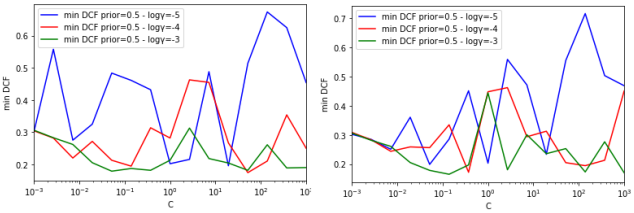
We first focus on the quadratic polynomial kernel. We have to jointly estimate the parameters C and c, so we perform a grid search with 3-folds cross-validation approach on our target application.

Figure 10: *Quadratic kernel: 3-folds cross-validation. Left: no PCA. Right: PCA with* $m = 7$



As shown in the two linecharts of Figure 10, the choice of the two hyper-parameters appears to be critical. Our analysis leads us to the choice of c = 10 and $C = 5 * 10^{-5}$.
We now employ RBF kernel formulation. We need to jointly estimate the two hyper-parameters γ and C. We again resort to a grid search with 3-folds cross-validation approach on our main application.

Figure 11: *RBF kernel: 3-folds cross-validation. Left: no PCA. Right: PCA with* $m = 7$



The choice of the two hyper-parameters is again difficult, as we can see from Figure 11. The adopted values, in the end, are $\gamma = 10^{-3}$ and $C = 10^{-1}$.

Figure 12: *All SVM models: 3-folds cross-validation. min DCFs computed using the estimated hyper-parameters*

	3-folds		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
	Z-normalized data - no PCA		
Linear SVM ($C=10^{-2}$)	0.154	0.683	0.277
Polynomial SVM ($C=5*10^{-5}$, c=10, d=2)	0.188	0.671	0.336
RBF SVM ($C=10^{-1}$, $\gamma=10^{-3}$)	0.172	0.708	0.257
	Z-normalized data - PCA (m=7)		
Linear SVM ($C=10^{-2}$)	0.154	0.642	0.280
Polynomial SVM ($C=5*10^{-5}$, c=10, d=2)	0.188	0.671	0.336
RBF SVM ($C=10^{-1}$, $\gamma=10^{-3}$)	0.171	0.704	0.258

As we expected, the comparison reported in Figure 12 shows that in our main application with $\widetilde{\pi} = 0.5$ the linear model outperforms the quadratic ones. Furthermore, it is interesting to notice that the reduction to 7 dimensions does not affect in any way the performances.

The hyper-parameter C adds a penalty for each misclassified sample. If C is small, the penalty will be low, so a decision boundary with a large margin is chosen at the expense of a greater number of misclassifications. If C is large, SVM tries to minimize the number of misclassified samples and this results in a decision boundary with a smaller margin.
For the linear SVM, $C = 10^{-2}$ and this allows to have larger margin for the separation rule. For the quadratic SVM and RBF SVM, the chosen hyper-parameters C are in the lower range of the values tested through 3-folds cross-validation, which suggests that also in these cases performances are better with larger margin for the separation rule.
For quadratic SVM and RBF SVM, larger margins correspond to less complex separation rules while for high values of C, in order to minimize the cost due to misclassificated points, separation rules tend to have more complex shapes. Since linear models seem to perform better on this dataset, this explains why we obtain better results with small values of C (which correspond to simpler separation rules).

Figure 13: *Comparison with previous models*

	3-folds		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
	Z-normalized data - PCA (m=7)		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.538	0.219
Tied-Cov	0.112	0.569	0.224
Linear SVM ($C=10^{-2}$)	0.154	0.642	0.280

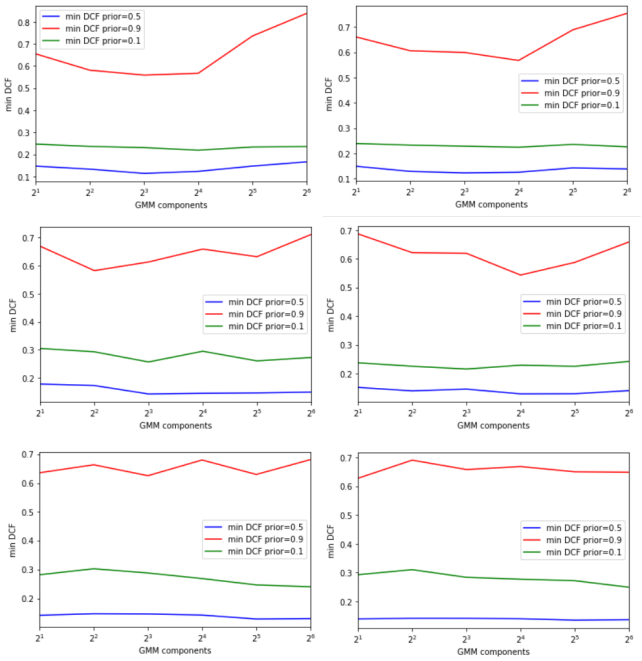
As we can see in Figure 13, linear SVM is not very effective if compared with previously analysed models.

3.4 GMM

The last model we will analyse is a generative one based on training a Gaussian Mixture Model (GMM) over the data of each of the two classes.
We will consider the full covariance model, the diagonal model and the one with covariance tying. For the model with tied covariance, tying takes place at class level, so different classes will have different covariance matrices.
We resort again to the 3-folds approach to choose the number of components and to compare different models.
The analysis is reported in Figure 14, from which it is clear that the models used on the application with $\widetilde{\pi}$=0.9 show a very unstable behaviour and their performances tend to worsen when the number of components is increased. This is clearly an indication of data over-fitting.
On the other hand, the other two applications seem to provide more stable min DCFs and a smaller degree of over-fitting.

For our target application, the full covariance model with 2^3 components and without PCA seems to give the best results.

Figure 14: *GMM: min DCFs for different priors, different models and different numbers of GMM components. Left: no PCA. Right: PCA with m = 7. Top: full covariance model. Middle: diagonal covariance model. Bottom: tied covariance model.*



In conclusion, the results of the best models so far are reported in Figure 15.

Figure 15: *min DCFs for the best models*

	3-folds		
	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
	Z-normalized data - no PCA		
Full-Cov, 8 GMM components	0.114	0.559	0.219
	Z-normalized data - PCA (m=7)		
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.538	0.219
Tied-Cov	0.112	0.569	0.224

For our target application and for the application with $\widetilde{\pi} = 0.1$ the performances are very similar among the three classifiers, while for the application with $\widetilde{\pi} = 0.9$ there are some differences and the Linear Logistic Regression model seems to outperform the other two.

3.5 Scores calibration

Up to now we have only considered the min DCF metric. As already discussed, min DCF measures the cost we would pay if we made optimal decisions for the validation set using the scores of the recognizer.

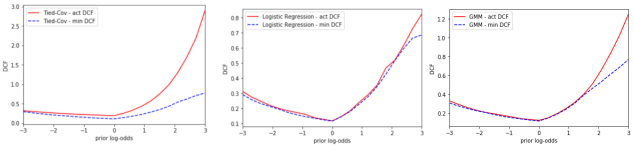
The cost that we actually pay, however, depends on the goodness of the threshold we use to perform class assignment.
We therefore start considering actual DCFs.
If scores are well calibrated, the optimal threshold that optimizes the Bayes risk is the theoretical threshold $t = -log\frac{\widetilde{\pi}}{1-\widetilde{\pi}}$.
We now evaluate the actual DCF to measure how good the models would behave if we were using the theoretical threshold for each application, which means we are assuming that scores are already well-calibrated:

Figure 16: *min DCFs and actual DCFs, 3-folds approach*

	$\widetilde{\pi}$ = 0.5		$\widetilde{\pi}$ = 0.9		$\widetilde{\pi}$ = 0.1	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
	Z-normalized data - no PCA					
Full-Cov, 8 GMM components	0.114	0.122	0.559	0.692	0.219	0.242
	Z-normalized data - PCA m=7					
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.117	0.538	0.545	0.219	0.227
MVG Tied-Cov	0.112	0.191	0.569	1.422	0.224	0.274

From Figure 16 we can observe that the Logistic Regression produces scores that are already well-calibrated for all the considered applications.
In the case of the MVG Tied-Cov model, for our main application there is a loss of $\approx 70\%$ that becomes very large in case of $\widetilde{\pi} = 0.9$, while there is a loss of $\approx 25\%$ in case of the $\widetilde{\pi} = 0.1$ application.
In case of the GMM model with 8 components, there is a loss between $\approx 10\%$ and $\approx 20\%$ among the different applications.
This analysis can also be verified through Bayes error plots, which show the DCFs for different applications.

Figure 17: *Bayes error plots with 3-folds approach. Left: MVG Tied-Cov model, PCA with m = 7. Middle: Linear Logistic Regression model, PCA with m = 7. Right: GMM Full-Cov model with 8 GMM components, no PCA.*



The Bayes error plot from Figure 17 for the MVG Tied-Cov model shows poor calibrated scores over a wide range of applications, while for the Linear Logistic Regression and for the GMM Full-Cov scores seem to be already well-calibrated over a sufficiently wide range of applications.
However, we can try employing score calibration on all the three models. A general approach consists in computing a transformation function f that maps the scores s of each classifier to well-calibrated scores $s_{cal} = f(s)$.
We assume that function f is linear in s:

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

Since *f(s)* should output well-calibrated scores, it can be interpreted as the log-likelihood ratio for the two class hypotheses:

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

so the class posterior probability for a prior $\widetilde{\pi}$ can be written as:

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

We can interpret the scores s as features so that this expression will be similar to the log posterior ratio of the Logistic Regression model. If we rewrite:

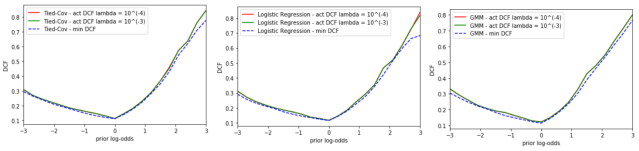
$$\beta' = \beta + log\frac{\widetilde{\pi}}{1-\widetilde{\pi}}$$

then we have exactly the same model and we can use the prior-weighted Logistic Regression model that we already discussed to learn the model parameters α, β' over our training scores.
To recover the calibrated scores *f(s)* we need to compute:

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

We are specifying a certain prior $\widetilde{\pi}$ (0.5 in our case), so we are effectively optimizing the calibration for this specific application, but this approach will provide good calibration for different applications anyway.
This time we start from Bayes error plots so that we can try different values for the hyper-parameter λ and see how they affect score calibration.

Figure 18: *Bayes error plots after calibration with 3-folds approach and prior-weighted logistic regression. Left: MVG Tied-Cov model, PCA with m = 7. Middle: Linear Logistic Regression model, PCA with m = 7. Right: GMM Full-Cov model with 8 GMM components, no PCA.*



As evident from Figure 18 and as expected from the previous considerations, score calibration has proven to be effective for the Tied-Cov model, while the Logistic Regression was already producing well-calibrated scores. For the GMM model, where the previous Bayes error plot in Figure 17 was showing already well-calibrated scores the calibrated scores provide a slightly worse actual DCF, while in the right part where scores were not calibrated the procedure is very effective. It is also evident that the choice of the prior does not influence the performances for application with a **different** effective prior.
Since there are no significant differences between the two calibrations with $\lambda = 10^{-4}$ and $\lambda = 10^{-3}$, we can again choose $\lambda = 10^{-4}$.

We can now evaluate again actual DCFs after calibration. Since the final scores should be calibrated, actual DCFs are computed using the theoretical threshold.

Figure 19: *min DCFs and actual DCFs after score calibration, 3-folds approach*

	$\tilde{\pi}$ = 0.5		$\tilde{\pi}$ = 0.9		$\tilde{\pi}$ = 0.1	
	minDCF	actDCF (cal)	minDCF	actDCF (cal)	minDCF	actDCF (cal)
Z-normalized data - no PCA						
Full-Cov, 8 GMM components	0.114	0.122	0.559	0.599	0.219	0.239
Z-normalized data - PCA m=7						
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.115	0.117	0.538	0.545	0.219	0.226
MVG Tied-Cov	0.112	0.114	0.569	0.600	0.224	0.229

The results shown in Figure 19 confirm our previous analysis. Indeed, the model which benefits the most from score calibration is the MVG Tied-Cov, whose actual DCFs are now almost equal to the min DCFs. As already mentioned, the Linear Logistic Regression was already providing well-calibrated scores and the procedure is not improving the classification. For the Full-Cov GMM model, there is an improvement especially for the application with prior $\widetilde{\pi} = 0.9$.
In the end, all the models provide actual DCFs that are close to the computed min DCFs.

4 Experimental results

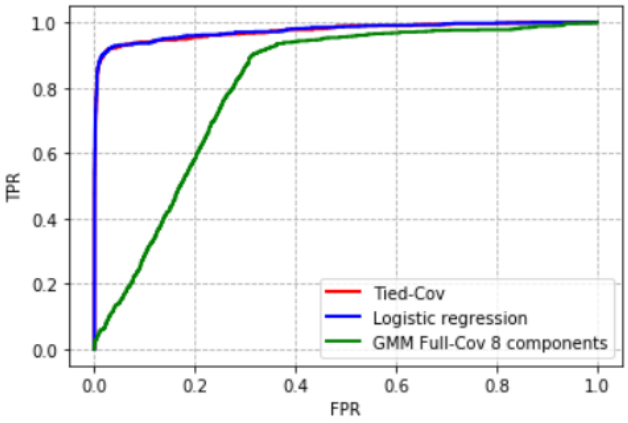
We now analyse the different models performances on the test set in terms of min DCFs.
In the previous analysis of the models we have considered both single-fold and 3-folds cross validation approaches, but our focus was on the 3-folds approach which now allows us to re-train the models over all the training set.
The results are collected in Figure 20. They are consistent with the ones obtained in the previous analysis and the best models for our target application remain MVG Tied-Cov PCA with m = 7, Linear Logistic Regression PCA with m = 7 and GMM Full-Cov 8 GMM components no PCA.
The similarity between validation and evaluation results suggests that there are no relevant differences between the evaluation population and the training population.

Figure 20: *min DCFs over test set for all models trained over all training set*

	$\tilde{\pi}$ = 0.5	$\tilde{\pi}$ = 0.9	$\tilde{\pi}$ =0.1
Z-normalized data - no PCA			
MVG Full-Cov	0.141	0.683	0.283
MVG Diag-Cov	0.187	0.616	0.325
MVG Tied-Cov	0.110	0.586	0.206
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.110	0.542	0.198
Linear SVM ($C=10^{-2}$)	0.150	0.604	0.300
Polynomial SVM ($C=5*10^{-5}$, c=10, d=2)	0.182	0.610	0.345
RBF SVM ($C=10^{-1}$, $\gamma=10^{-3}$)	0.176	0.811	0.259
GMM - Full Cov- 2 components	0.153	0.748	0.264
GMM - Full Cov- 4 components	0.122	0.530	0.250
GMM - Full Cov- 8 components	0.109	0.563	0.223
GMM - Full Cov- 16 components	0.124	0.522	0.240
GMM - Diag Cov- 2 components	0.179	0.643	0.323
GMM - Diag Cov- 4 components	0.165	0.579	0.333
GMM - Diag Cov- 8 components	0.131	0.546	0.261
GMM - Diag Cov- 16 components	0.134	0.594	0.259
GMM - Tied Cov- 2 components	0.138	0.594	0.275
GMM - Tied Cov- 4 components	0.143	0.634	0.303
GMM - Tied Cov- 8 components	0.137	0.594	0.281
GMM - Tied Cov- 16 components	0.131	0.595	0.264
Z-normalized data - PCA (m=7)			
MVG Full-Cov	0.139	0.569	0.292
MVG Diag-Cov	0.196	0.765	0.504
MVG Tied-Cov	0.109	0.580	0.203
Log Reg ($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.109	0.545	0.198
Linear SVM ($C=10^{-2}$)	0.152	0.601	0.303
Polynomial SVM ($C=5*10^{-5}$, c=10, d=2)	0.183	0.607	0.348
RBF SVM ($C=10^{-1}$, $\gamma=10^{-3}$)	0.174	0.820	0.256
GMM - Full Cov- 2 components	0.153	0.727	0.267
GMM - Full Cov- 4 components	0.122	0.551	0.246
GMM - Full Cov- 8 components	0.116	0.519	0.239
GMM - Full Cov- 16 components	0.125	0.585	0.252
GMM - Diag Cov- 2 components	0.156	0.635	0.250
GMM - Diag Cov- 4 components	0.130	0.608	0.224
GMM - Diag Cov- 8 components	0.118	0.539	0.236
GMM - Diag Cov- 16 components	0.127	0.607	0.246
GMM - Tied Cov- 2 components	0.137	0.562	0.278
GMM - Tied Cov- 4 components	0.140	0.612	0.306
GMM - Tied Cov- 8 components	0.132	0.654	0.278
GMM - Tied Cov- 16 components	0.132	0.600	0.260

We can now proceed with score calibration and apply the same process discussed in 3.5, so that our system can make use of the theoretical threshold to perform optimal decision.
We can compare our classifiers through a ROC plot, which is shown in Figure 21.
The best classifiers are the ones with the highest AUC (Area Under Curve), in our case the MVG Tied-Cov PCA with m = 7 and the Linear Logistic Regression PCA with m = 7. Their curves show a high slope in the left part, which means that the models are able to correctly classify true pulsar signals while keeping the false positives to a minimum.

Figure 21: *ROC plot for models comparison: MVG Tied-Cov PCA with $m = 7$, Linear Logistic Regression PCA with $m = 7$, GMM Full-Cov 8 GMM components no PCA*



5 Conclusions

Our analysis led us to two different and well performing models, the MVG Tied-Cov PCA with m = 7 and the Linear Logistic Regression PCA with m = 7, $\lambda = 10^{-4}$ and $\pi_T = 0.5$. Therefore, linear models appear to be more suited for the HTRU2 classification task with respect to quadratic models.
We are able to achieve a DCF of ≈ 0.1 for the target application $\widetilde{\pi} = 0.5$, altough also for the other two considered applications with $\widetilde{\pi} = 0.1$ and $\widetilde{\pi} = 0.9$ results are acceptable, with DCFs of ≈ 0.2 and ≈ 0.6, respectively.
The choices we made on our training/validation sets proved to be effective also for the evaluation set.

References

[1] R. J. Lyon. Htru2. https://doi.org/10.6084/m9.figshare.3080389.v1.

[2] M. J. Keith et al. The high time resolution universe pulsar survey - i. system configuration and initial discoveries. *Monthly Notices of the Royal Astronomical Society, vol. 409, pp. 619-627.*, 2010. https://doi.org/10.1111/j.1365-2966.2010.17325.x.