Politecnico di Torino

Classification of HTRU2 dataset

Machine Learning and Pattern Recognition

Davide Ariotto
S302489@studenti.polito.it

January 31, 2023

Abstract

Our goal is to compare the results obtained from the HTRU2 dataset using several common Machine Learning (ML) algorithms. The dataset, which contains nearly 18,000 observations of astronomical objects used to identify pulsars, will be first explored to understand data distribution and feature correlation. We will then move on to implementing different classifiers and finally calibrating scores. Ultimately, the results will guide us in selecting linear models as promising solutions for the classification task at hand.

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 inter-stellar 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 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 Test data. 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).
We will apply various classification models to the task, analyse and discuss the results.

2. Feature Analysis

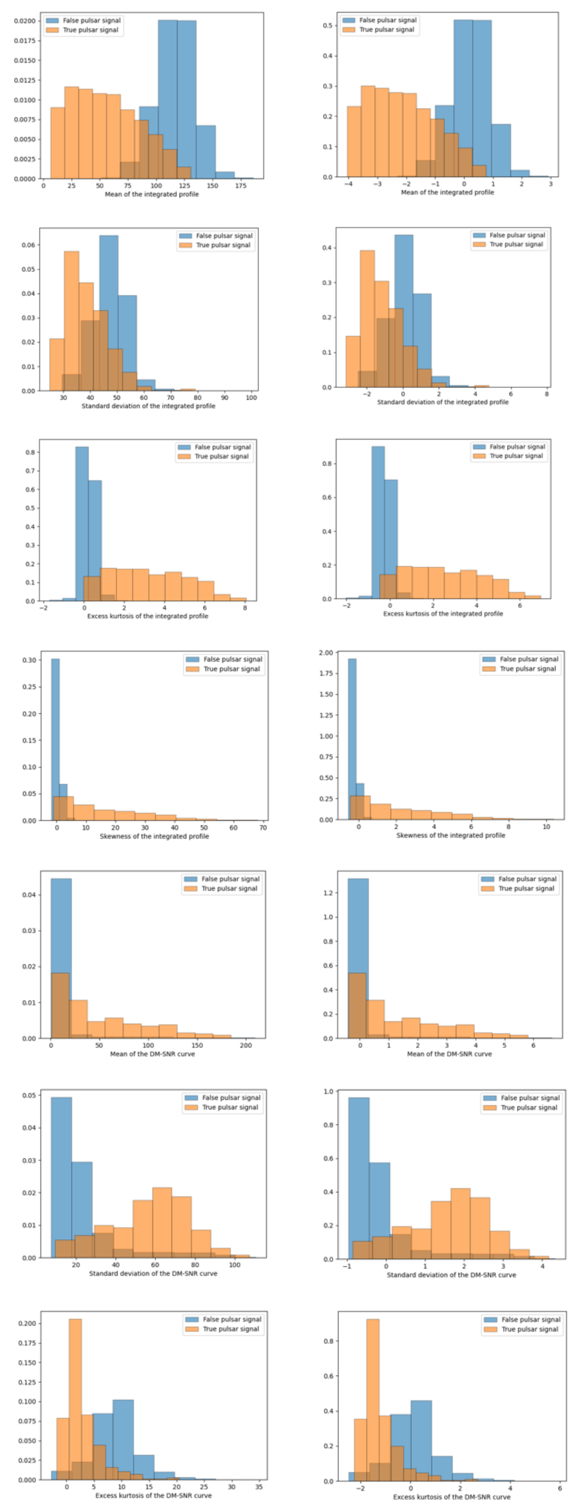
Each candidate is described by 8 continuous variables and a single class variable/label. These are summarised below:

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

We can see from figure 1 that the values are quite high and for this reason I decided to apply Z-normalization to the dataset in order to prevent overflow errors during mathematical operations (especially when dealing with exponential). This operation transforms the dataset such that the mean of all the values is 0 and the standard deviation is 1 (centering and scaling to unit variance in technical terms):

$$z_i = \frac{x_i - \mu}{\sigma} \quad \forall i \ \in \{1, \dots, n\}$$

where $\mathbf{x}_i$ is the observed value for the i-th sample, $\boldsymbol{\mu}$ is the mean vector computed along columns and $\boldsymbol{\sigma}$ is the standard deviation (again along columns).



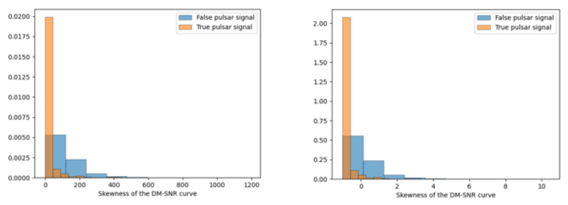
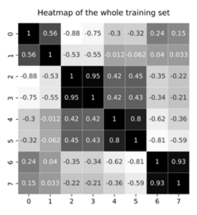


Figure 1: Histograms of the original and z-normalized features

The orange ones refer to the True class while the blue ones to the False class and we can see that the distribution remained very similar. I decided to do no more pre-processing because there's just a small number of outliers, in case of a bigger presence I would have proceeded with a Gaussianization.
Now let's analyse the correlation between features. We'll focus on the Pearson coefficient that is defined as follows:

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



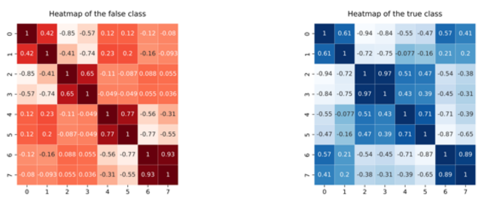


Figure 2: Heatmaps with Pearson correlation coefficient of positive class (blue), negative class (red) and whole training set (grey)

Darker colour (and higher values, the maximum is 1) implies a stronger correlation, while lighter colour the opposite. Firstly, we can see that the heatmaps aren't that different and this means that the correlation between features doesn't depend on the class. Secondly, we notice that features 2-3, 4-5, 6-7 have a quiet big coefficient, especially the last couple. This means that if we substitute features 6 and 7 with a single one we shouldn't lose many information. To do this we can apply PCA: it reduces the original space with n dimension into a m subspace with n << m. I've done it with m=7, m=6 and m=5.

3. Classifying the features

To perform the classification task we can use two approaches: single fold and k-fold.
In the first one we basically split the original data (the training set) into validation data (33,3 %) and training data (66,6 %). This approach is faster, but the model has less data to use. K-folds instead consists of dividing the data into "k" folds, training the model on k-1 of them and evaluating it on the remaining one. This process is repeated k times, with each fold being used as the test set once. The model may not be optimal because it's not trained over the whole training set, but it has more data available compared to the single fold approach. To tackle the first issue we'll get the final classifier by re-training over the whole training set.
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.

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

In order to find the best approach we will measure performances through 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. The evaluation is carried out on the validation subset (extracted from the training set).

3.1 Gaussian classifiers

The first classifiers that have been used are the Gaussian classifiers: multivariate gaussian (MVG) classifier, MVG classifier with Naive Bayes assumption and MVG classifier with tied covariance. All of them assume gaussian distributed data:

$$X \sim N(\mu, \Sigma)$$

where μ is the mean and Σ is the covariance matrix.
Now we measure the min DCF for the three of them:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-normalized Features - no PCA						
Full-Cov	0.116	0.257	0.559	0.142	0.283	0.656
Diag-Cov	0.171	0.264	0.621	0.193	0.312	0.747
Tied Full-Cov	0.091	0.209	0.474	0.112	0.223	0.575
Z-normalized Features - PCA (m = 7)						
Full-Cov	0.115	0.264	0.523	0.139	0.298	0.629
Diag-Cov	0.182	0.485	0.563	0.214	0.505	0.724
Tied Full-Cov	0.091	0.209	0.474	0.112	0.222	0.574
Z-normalized Features - PCA (m = 6)						
Full-Cov	0.120	0.261	0.544	0.152	0.285	0.622
Diag-Cov	0.195	0.506	0.568	0.223	0.525	0.722
Tied Full-Cov	0.109	0.236	0.507	0.140	0.258	0.582
Z-normalized Features - PCA (m = 5)						
Full-Cov	0.129	0.240	0.630	0.148	0.245	0.644
Diag-Cov	0.197	0.425	0.595	0.220	0.453	0.734
Tied Full-Cov	0.124	0.237	0.517	0.150	0.261	0.575

Figure 3: min DCF on the validation set for MVG classifiers

In Figure 3 we can see the min DCF computed on the validation set with different prior probabilities and with some configurations of PCA. We can notice that PCA degraded performances because the min DCF increased in all cases, except for the tied full-cov where we had the same value for "no PCA" and "PCA m=7". Anyway, we've seen that reducing the features from 8 to 7 for the last model didn't cause loss of useful information; on the other hand reducing them to much made the min DCF increase too much. For this reason, from now on we won't consider the PCA m = 5 option anymore.
Comparing the results of the different model we notice that for the tied full-cov the hypothesis of interpreting the covariance matrix as representing noise independent of the class is correct, and the covariance matrices probably are similar for the two classes.
For the diag-cov, the assumption of independent components may be wrong because it had the worst performance.
The full-cov classifier obtained acceptable results, but anyway worse than tied full-cov. This could mean that we don't have enough data to estimate all the parameters.
Finally, we could see that the unbalanced tasks leaded to worse results for all models with respect to the balanced one. All the considerations above are valid both for single-fold and k-fold.
From this analysis we understand that linear models performed better than quadratic ones.
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 and b. The decision rules are assumed to be linear hyperplanes orthogonal to w:

$$w^T x + b$$

The goal is to find the hyperplane that maximizes class probabilities.
Firstly we look for λ, a hyper-parameter that minimize the min DCF. The regularization term $(\lambda/2)*\|w\|^2$ helps obtaining simpler solutions in terms of lower norm of w. In other words, λ that tells us how much we should weigh the cost due to having a high norm with respect to the classification cost. This will help reducing the risk of overfitting the training data. If we select a large value of λ we will obtain a solution that has a small norm but won't be able to separate well the classes. On the other hand, if λ is too small, we will get a solution with a higher norm but poor classification accuracy for test/unseen data.
We'll look for the best value of the parameter with single-fold, k-fold, no PCA and PCA with m=7 and m=6. The results are summarized here:

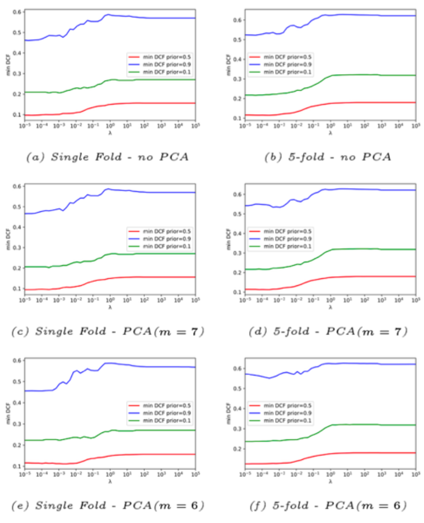
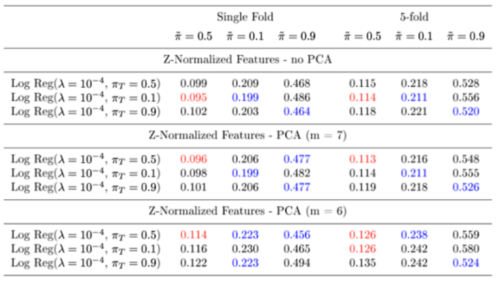


Figure 4: min DCF with different values of λ

From this figure, we can see that there isn't a big difference in the results with the two different types of folds. However, we'll consider mostly the k-fold because it guarantees more reliability and the final model should be more robust due to the bigger number of training samples. The chosen/best value for the hyper-parameter is $\lambda = 10^{-4}$.
Now let's compute again the min DCF:



	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.099	0.209	0.468	0.115	0.218	0.528
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.095	0.199	0.486	0.114	0.211	0.556
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.102	0.203	0.464	0.118	0.221	0.520
Z-Normalized Features - PCA (m = 7)						
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.096	0.206	0.477	0.113	0.216	0.548
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.098	0.199	0.482	0.114	0.211	0.555
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.101	0.206	0.477	0.119	0.218	0.526
Z-Normalized Features - PCA (m = 6)						
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.114	0.223	0.456	0.126	0.238	0.559
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.1$)	0.116	0.230	0.465	0.126	0.242	0.580
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.9$)	0.122	0.223	0.494	0.135	0.242	0.524

Figure 5: min DCF computed using the estimated hyper-parameter

Re-balancing the cost of different classes through π_T didn't improve the model. We got similar min DCF values for all values of π_T but the model that performed better is the one with PCA m = 7 and $\pi_T = 0.5$. We can also see an analogy with MVG classifiers because the results from PCA m = 6 are the worst ones also in this case.

3.3 Support Vector Machines

We now turn our attention to SVMs. We will see linear svm (with and without class balancing) and quadratic svm (with polynomial and rbf kernel), even if from the previous results we expect the linear SVM to perform better.
The main difference between support vector machines and logistic regression is that they try to find the hyperplane that separates the classes with the largest margin.
We want to find the model parameters w and b that maximize the distance of the closest point. We basically compute, for each w and b, the distances from all the points with respect to the separation boundary and select the minimum one (minimum distance).

$$w^*, b^* = \arg\max_{w,b} \min_{i \in \{1 \dots n\}} d(x_i)$$
$$\text{subject to} \quad z_i(w^T x + b) > 0$$

Optimizing this problem is not easy and for this reason we consider a dual formulation:

$$J^D(\boldsymbol{\alpha}) = -\tfrac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T\boldsymbol{1}$$
$$\textit{subject to } 0 \le \alpha_i \le C, \forall i \in \{1,\dots,n\}, \textstyle\sum_{i=1}^{n}\alpha_i z_i = 0$$

where n is the number of training samples, C is a hyper-parameter, 1 is a n-dimensional vector of ones, z_i is the class label for the i-th sample encoded as +1 (true pulsar signal) or -1 (false pulsar signal). H is a matrix whose elements are

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

From the dual formulation we then derive the primal solution.
Since we use the L-BFGS-B algorithm, that it is only able to handle box constraints, we have to modify the formulation to already include the second constraint. This is the modified dual formulation:

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

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

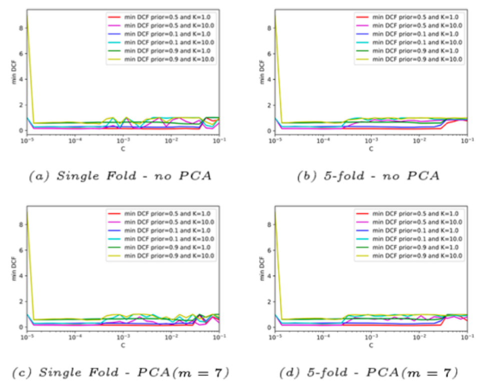
and, since we use the mapping

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

with K=1, the matrix H^ becomes:

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

For linear SVM we need to tune the hyper-parameters C and K. I tried K = 1.0 and K = 10.0 with both single-fold and k-fold cross-validation.



(a) Single Fold - no PCA *(b) 5-fold - no PCA*

(c) Single Fold - PCA(m = 7) *(d) 5-fold - PCA(m = 7)*

Figure 6: min DCF for linear svm without class balancing

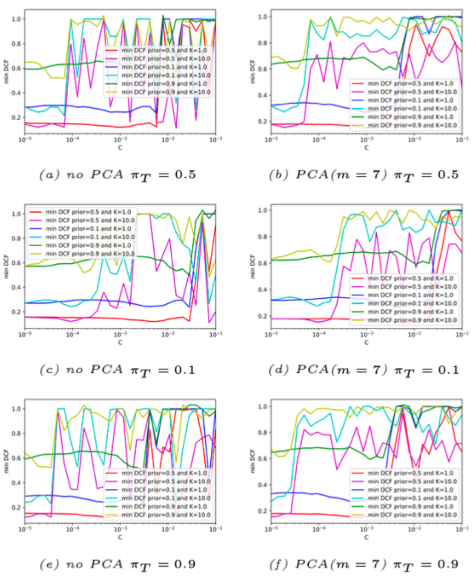
From figure 6 we select as best values K = 1.0 and C = 10^{-2}. We notice that k-fold results are consistent with single-fold results, meaning that our model behaves correctly.
Balancing is done by considering a different value of C for the different classes in the box constraints of the dual formulation:

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

where $C_i = C_T$ for samples of class H_T and $C_i = C_F$ for samples of class H_F. They are defined as follows:

$$C_T = C\frac{\pi_T}{\pi_T^{emp}} \qquad C_F = C\frac{\pi_F}{\pi_F^{emp}}$$

Now let's look at the min DCF for linear svm with class balancing:



(a) no PCA $\pi_T = 0.5$ *(b) PCA(m = 7) $\pi_T = 0.5$*

(c) no PCA $\pi_T = 0.1$ *(d) PCA(m = 7) $\pi_T = 0.1$*

(e) no PCA $\pi_T = 0.9$ *(f) PCA(m = 7) $\pi_T = 0.9$*

Figure 7: min DCF for linear svm with class balancing

Figure 7 shows that K = 1 provides a better and more stable min DCF. Even if prior π~ = 0.5 and K = 10 provide a better min DCF with respect to prior π~ = 0.5 and K=1, the latter is preferred because its graph is more stable for a wide range of C. We decide to choose a different value of the hyper-parameter C for each prior. The table below summarizes the computed min DCF:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Linear SVM($C = 10^{-2}, unbalanced$)	0.130	0.247	0.600	0.158	0.287	0.612
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.121	0.243	0.640	0.151	0.274	0.677
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.122	0.247	0.622	0.151	0.276	0.656
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.122	0.247	0.620	0.152	0.272	0.643
Z-Normalized Features - PCA (m = 7)						
Linear SVM($C = 10^{-2}, unbalanced$)	0.130	0.250	0.605	0.159	0.285	0.618
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.122	0.247	0.623	0.150	0.276	0.658
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.122	0.247	0.621	0.152	0.272	0.651
Z-Normalized Features - PCA (m = 6)						
Linear SVM($C = 10^{-2}, unbalanced$)	0.136	0.260	0.611	0.165	0.293	0.614
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.123	0.253	0.636	0.153	0.282	0.679
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.123	0.250	0.622	0.154	0.286	0.662
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.125	0.253	0.625	0.154	0.289	0.659

Figure 8: min DCF on the validation set for linear svm with/without balancing

The class re-balancing has improved the model and for this reason we will mainly consider the linear model with $\pi_T = 0.5$ and C = 10^{-3}.

Regarding the non-linear formulation in SVMs, the non-linearity is obtained through an implicit expansion of the features in a higher dimensional space. The dual svm formulation depends on the training samples only through dot-products, and we can compute a classification score through scalar products between training and evaluation samples.

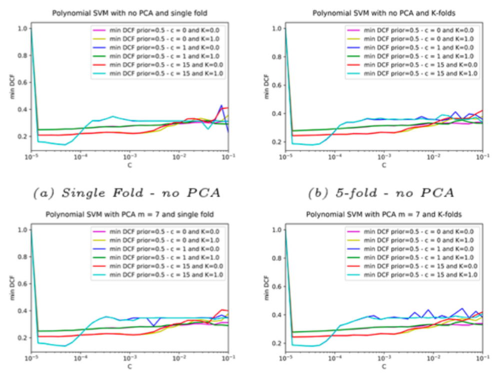
For this reason, it's not required to explicitly compute the feature expansion, it's enough to be able to compute the scalar product between the expanded features, the so-called kernel function k:

$$k(x_1, x_2) = \phi(x_1)^T \phi(x_2)$$

We will test two different kernels:

1. Polynomial kernel of d = 2;
   $k(x_i, x_j) = (x^T_i x_j + c)^d$
2. Radial Basis Function (RBF) kernel:
   $k(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2}$

For the quadratic polynomial svm we need to estimate two hyper-parameters c and C through a cross-validation.

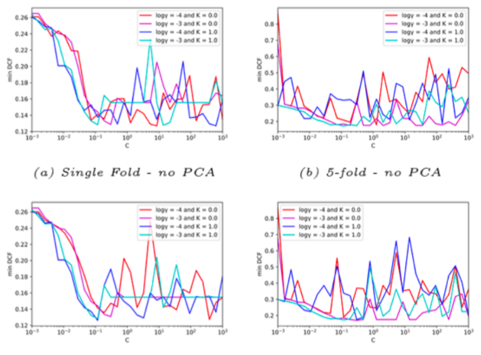


(a) Single Fold - no PCA *(b) 5-fold - no PCA*

(c) Single Fold - PCA(m = 7) *(d) 5-fold - PCA(m = 7)*

Figure 9: min DCF for different values of C (polynomial svm)

From the above figure we see that the minimum min DCF is found with c = 15 and C = $5{*}10^{-5}$.
Now let's estimate the two hyper-parameters γ and C for the RBF svm:

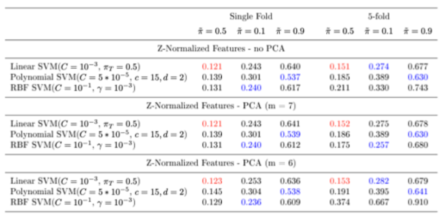


(a) Single Fold - no PCA *(b) 5-fold - no PCA*

(c) Single Fold - PCA(m = 7) *(d) 5-fold - PCA(m = 7)*

Figure 10: min DCF for different values of C (RBF svm)

Looking at figure 10 we selected γ = 10^{-3} and C = 0.1.
Now it's time to look at the performances:



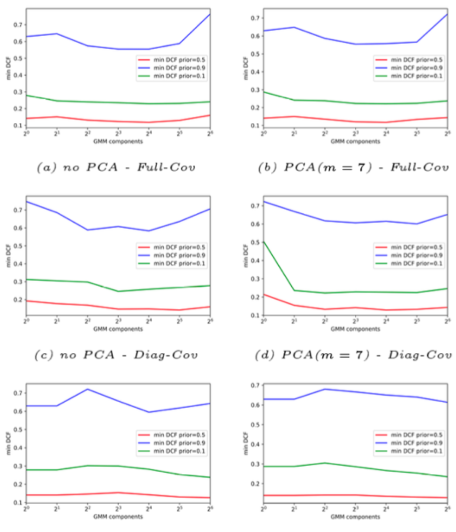
	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.640	0.151	0.274	0.677
Polynomial SVM($C = 5 * 10^{-5}$, $c = 15, d = 2$)	0.139	0.301	0.537	0.185	0.389	0.630
RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$)	0.131	0.240	0.617	0.211	0.330	0.743
Z-Normalized Features - PCA (m = 7)						
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678
Polynomial SVM($C = 5 * 10^{-5}$, $c = 15, d = 2$)	0.139	0.301	0.539	0.186	0.389	0.630
RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$)	0.131	0.240	0.612	0.175	0.257	0.680
Z-Normalized Features - PCA (m = 6)						
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.123	0.253	0.636	0.153	0.282	0.679
Polynomial SVM($C = 5 * 10^{-5}$, $c = 15, d = 2$)	0.145	0.304	0.538	0.191	0.395	0.641
RBF SVM($C = 10^{-1}$, $\gamma = 10^{-3}$)	0.129	0.236	0.609	0.374	0.667	0.910

Figure 11: min DCF on the validation set for the three types of SVMs

We can conclude that the linear approaches fit better the task. The linear svm with C = 10^{-3} and π_T = 0.5 is the best one. As seen before PCA didn't prove to be effective in improving the results quality (but again it helps reducing the complexity).
Anyway, it is important to notice that SVMs performed worse than both gaussian classifiers (tied full-cov) and linear logistic regression.

3.4 Gaussian Mixture Models

Now let's consider an example of a generative model. It is based on training a Gaussian Mixture Model (GMM) over the data of each class. GMM are able to deal better with generalized data and for this reason we expect better results than the Gaussian classifier. We will use GMM with full covariance, full diagonal, and tied covariance. In the tied covariance model, tying takes place at class level, so different classes have distinct covariance matrices.
The likelihood increases if we increase the number of components and for this reason we can't estimate them from it (they'll tend to be infinite). We'll use the cross-validation to evaluate how good the model is on the validation set and based on that we'll select the number of components. Here are the results for cross-validation on k-fold:



(a) no PCA - Full-Cov *(b) PCA(m = 7) - Full-Cov*

(c) no PCA - Diag-Cov *(d) PCA(m = 7) - Diag-Cov*

(e) no PCA - Full Tied-Cov *(f) PCA(m = 7) - Full Tied-Cov*

Figure 12: min DCF for different priors, different models and numbers of GMM components.

It is clear that the models used on the application with π~ = 0.9 show a very unstable behavior and their results are bad especially when the number of components is increased. This means that they're overfitting. There are some signs of overfitting also in the other two applications but much smaller. The best results are achieved with prior π~ = 0.5 .
There isn't a number of components that provides the best results for all three models and for this reason we'll choose three different values. We selected 16 components for the full-cov model, 32 for diag-cov and 64 for tied full-cov. Here's the computed min DCF with both single-fold and k-fold:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - no PCA						
Full-Cov, 16-G	0.104	0.202	0.520	0.118	0.229	0.555
Diag-Cov, 32-G	0.122	0.236	0.477	0.142	0.268	0.636
Tied Full-Cov, 64-G	0.110	0.226	0.560	0.127	0.238	0.643
Z-Normalized Features - PCA (m = 7)						
Full-Cov, 16-G	0.110	0.206	0.548	0.117	0.221	0.558
Diag-Cov, 32-G	0.102	0.199	0.481	0.133	0.225	0.601
Tied Full-Cov, 64-G	0.126	0.253	0.639	0.129	0.235	0.613
Z-Normalized Features - PCA (m = 6)						
Full-Cov, 16-G	0.117	0.209	0.583	0.127	0.217	0.600
Diag-Cov, 32-G	0.118	0.247	0.608	0.148	0.260	0.647
Tied Full-Cov, 64-G	0.128	0.233	0.692	0.135	0.259	0.671

Figure 13: min DCF on validation set for GMM

We conclude that the full-cov model (no pca) with 16 components is the best one followed by the tied full-cov with 64 components. PCA didn't prove to be effective again in general, except for the diag-cov model with single fold.

Now let's do a final comparison between the best models that we've seen in report so far:

	Single Fold			5-fold		
	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov	0.091	0.209	0.474	0.112	0.222	0.574
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.096	0.206	0.477	0.113	0.216	0.548
Linear SVM($C = 10^{-3}$, $\pi_T = 0.5$)	0.121	0.243	0.641	0.152	0.275	0.678
Full-Cov, 16-G	0.110	0.206	0.548	0.117	0.221	0.558

Figure 14: final comparison between different models

The tied full-cov and the logistic regression are the best ones. We'll ignore the linear svm because it's the one with the worst results in this set.

4. Score Recalibration

Until now we have only looked at the min DCF metric. As already discussed, it measures the cost we would pay if we made optimal decisions for the validation set using the scores of the recognizer.
However, the cost that we actually pay depends on the goodness of the threshold we use to perform class assignment. We therefore start considering the actual DCFs.
If scores are well calibrated, the optimal threshold that optimizes the Bayes risk is the theoretical threshold:

$$t = -\log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

We now evaluate the actual dcf to measure how good the models would behave if we were using the theoretical threshold for each application instead of the optimal one:

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov	0.112	0.190	0.222	0.274	0.574	1.422
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$)	0.113	0.115	0.216	0.226	0.548	0.567
Full-Cov, 16-G	0.117	0.121	0.221	0.228	0.558	0.596

Figure 15: comparison between min DCF and actual DCF

For the tied full-cov model the scores are not well calibrated: with π~ = 0.5 we have a loss of ≈ 70% and the loss even exceeds 100% with π~ = 0.9. The other two models provide scores already well-calibrated. This analysis can also be verified through Bayes error plots, which show the DCFs for different applications:

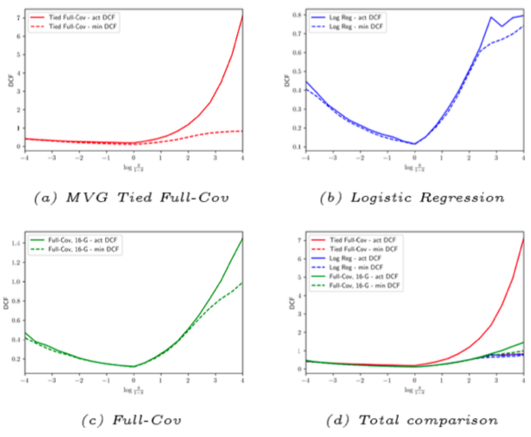


Figure 16: Bayes error plots pre-calibration

Observing the plots (especially the subplot d) we can see that logistic regression and gmm full-cov with 16 components are already well-calibrated for prior log-odds lower than 2. Instead, the mvg tied full-cov has worse performances, especially when prior log-odds is greater than 0.

We will then consider re-calibrating the scores. It consists of transforming the scores so that the theoretical threshold provides values closer to optimal over a wide range of effective prior π~.
More in details, we have to compute a transformation function f that maps the classifiers scores s 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 produce well-calibrated scores, it can be interpreted as the log-likelihood ratio for the two class hypotheses:

$$f(s) = \log \frac{f_{S|C}(s|\mathcal{H}_T)}{f_{S|C}(s|\mathcal{H}_F)} = \alpha s + \beta$$

The class posterior probability for prior π~ corresponds to:

$$\log \frac{P(C = \mathcal{H}_T|s)}{P(C = \mathcal{H}_F|s)} = \alpha s + \beta + \log \frac{\tilde{\pi}}{1 - \tilde{\pi}}$$

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 let:

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

we end up having the same model. We can use the prior-weighted logistic regression model to learn the model parameters α, β′ over our training scores.
To get the calibrated scores f(s) we need to compute:

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

Looking at the formula we see that we have to specify a prior π~ so we are effectively optimizing the calibration for a specific application π~. Anyway, the model will still provide good calibration for different applications. Let's compute DCFs and Bayes error plots:

	$\tilde{\pi} = 0.5$		$\tilde{\pi} = 0.1$		$\tilde{\pi} = 0.9$	
	min DCF	act DCF	min DCF	act DCF	min DCF	act DCF
Z-Normalized Features - PCA (m = 7)						
Tied Full-Cov (cal.)	0.112	0.114	0.222	0.227	0.574	0.612
Log Reg($\lambda = 10^{-4}$, $\pi_T = 0.5$) (cal.)	0.113	0.114	0.216	0.226	0.548	0.562
Full-Cov, 16-G (cal.)	0.117	0.122	0.221	0.250	0.558	0.600

Figure 17: min DCF and actual DCF post-calibration

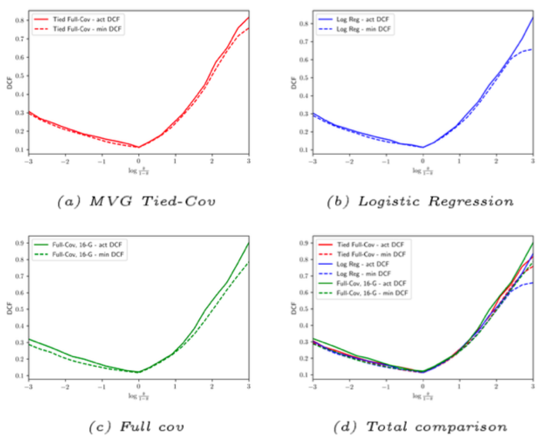


Figure 18: Bayes error plots post-calibration

From figures 17 and 18, we can see that the re-calibration works very well on all models and the values of the actual DCF becomes very similar to the min DCF ones.

From the total comparison (subplot d of figure 18), we can see that the three models provide pretty the same value of min DCF.

5. Experimental Results

In this section we'll observe the behavior of the models on the test set in terms of min DCF. We train the model with the training set and use the test set as evaluation data. Here are the summarized results:

	$\tilde{\pi} = 0.5$	$\tilde{\pi} = 0.1$	$\tilde{\pi} = 0.9$
Z-Normalized Features - no PCA			
MVG Full-Cov	0.140	0.284	0.646
MVG Diag-Cov	0.185	0.331	0.621
MVG Tied Full-Cov	0.110	0.207	0.591
Log Reg($\lambda = 10^{-4}, \pi_T = 0.5$)	0.110	0.199	0.534
Log Reg($\lambda = 10^{-4}, \pi_T = 0.1$)	0.110	0.197	0.534
Log Reg($\lambda = 10^{-4}, \pi_T = 0.9$)	0.115	0.206	0.510
Linear SVM($C = 10^{-2}$, unbalanced)	0.148	0.292	0.603
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.145	0.285	0.599
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.144	0.281	0.613
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.145	0.280	0.605
Polynomial SVM($C = 5 * 10^{-5}, c = 15, d = 2$)	0.204	0.481	0.642
RBF SVM($C = 10^{-1}, \gamma = 10^{-3}$)	0.169	0.249	0.709
GMM Full-Cov, 16-G	0.123	0.230	0.525
GMM Diag-Cov, 32-G	0.147	0.279	0.592
GMM Tied Full-Cov, 64-G	0.130	0.231	0.590
Z-Normalized Features - PCA (m = 7)			
MVG Full-Cov	0.139	0.295	0.574
MVG Diag-Cov	0.200	0.528	0.749
MVG Tied Full-Cov	0.112	0.212	0.570
Log Reg($\lambda = 10^{-4}, \pi_T = 0.5$)	0.108	0.201	0.546
Log Reg($\lambda = 10^{-4}, \pi_T = 0.1$)	0.107	0.199	0.533
Log Reg($\lambda = 10^{-4}, \pi_T = 0.9$)	0.111	0.205	0.555
Linear SVM($C = 10^{-2}$, unbalanced)	0.157	0.319	0.601
Linear SVM($C = 10^{-3}, \pi_T = 0.5$)	0.150	0.290	0.598
Linear SVM($C = 6 * 10^{-3}, \pi_T = 0.1$)	0.144	0.289	0.604
Linear SVM($C = 7 * 10^{-4}, \pi_T = 0.9$)	0.146	0.289	0.610
Polynomial SVM($C = 5 * 10^{-5}, c = 15, d = 2$)	0.226	0.594	0.643
RBF SVM($C = 10^{-1}, \gamma = 10^{-3}$)	0.159	0.250	0.660
GMM Full-Cov, 16-G	0.134	0.293	0.600
GMM Diag-Cov, 32-G	0.121	0.226	0.588
GMM Tied Full-Cov, 64-G	0.131	0.231	0.603

Figure 19: min DCF over test set

The first observation is that the results aren't that different from the ones that we got over the validation set, meaning that the models are consinstent. The similarity between validation and evaluation results also suggests that there are no relevant differences between the evaluation population and the training population.
The best models are: logistic regression with π_T = 0.1 and PCA m = 7, mvg tied full-cov (no PCA), and logistic regression with both π_T =0.5 and π_T =0.1 (no PCA).
It's interesting that the best results of logistic regression for our primary application (π~ = 0.5) are obtained with πT = 0.1, however the difference with respect to models using π_T = 0.5 is barely observable.

We can also compare our best classifiers through a ROC plot considering the models with PCA m = 7.

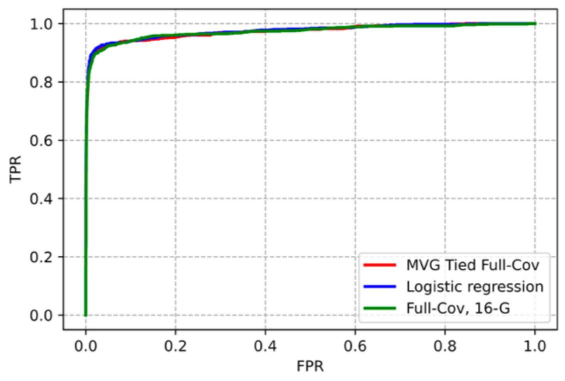


Figure 20: ROC between the three best models

The best classifiers are the ones with the highest AUC (Area Under Curve), in our case we don't have a single classifier that performs much better than the others.

Lastly, we can also verify if the chosen hyper-parameters provided close to optimal results on the test set, understanding if we took the correct values. In the next figures, we have compared the min DCF for different values of the hyper-parameters on the test/evaluation set and validation set.

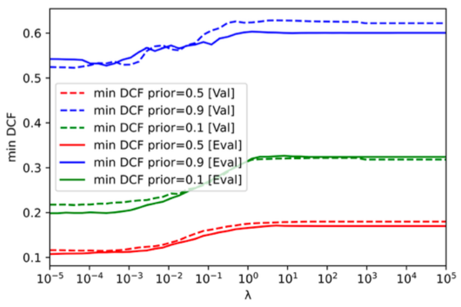


Figure 21: comparison of min DCF for different values of λ

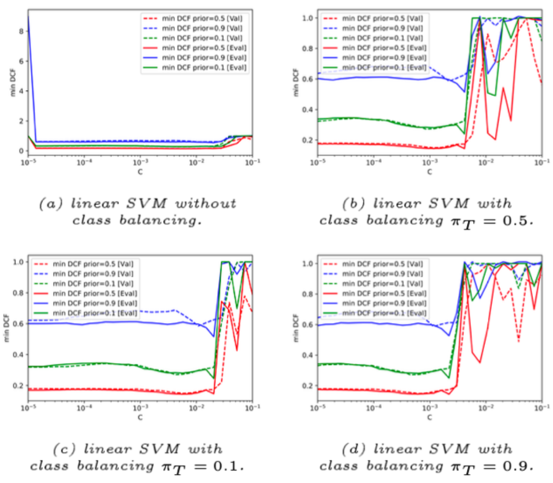


Figure 22: comparison of min DCF for different values of C

We can conclude that the values we chose for the hyper-parameters λ and C were quite accurate.

6. Conclusions

To summarize we have understood that linear models performed better than quadratic ones on the HTRU2 dataset. The best models are the logistic regression (PCA m = 7) with $\lambda = 10^{-4}$ and $\pi_T = 0.5$ and the mvg tied full-cov (PCA m = 7).
The correct choices for the hyper-parameters led us to a min DCF ≈ 0.1 for the target application ($\tilde{\pi} = 0.5$).
We must also notice that the min DCF of the other applications is quite low (respectively ≈ 0.2 for $\tilde{\pi} = 0.1$ and ≈ 0.5 for $\tilde{\pi} = 0.9$).
To end up, the choices made on the training/validation sets proved to be effective also for the evaluation set.

References

[1]. R. J. Lyon, HTRU2, https://figshare.com/articles/dataset/HTRU2/3080389/1
[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
[3]. Cheng Jun Zhang et al., A Review of Research on Pulsar Candidate Recognition Based on Machine Learning