Pulsar Detection

295515 Davide Pinti
295474 Giorgio Orioles

Contents

List of Figures

Abstract

This analysis aim to find the better ML algorithm for the classification of HTRU2 dataset. We will start to analyze the features and their correlation, then we will move to the analysis of different classifiers.

1 Introduction

Data Set Information:

HTRU2[1] is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South). 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 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. At present multi-class labels are unavailable, given the costs associated with data annotation. The data set shared here contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. These examples have all been checked by human annotators.

Attribute Information:

Each candidate is described by 8 continuous variables, and a single class variable. The first four are simple statistics obtained from the integrated pulse profile (folded 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. 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.

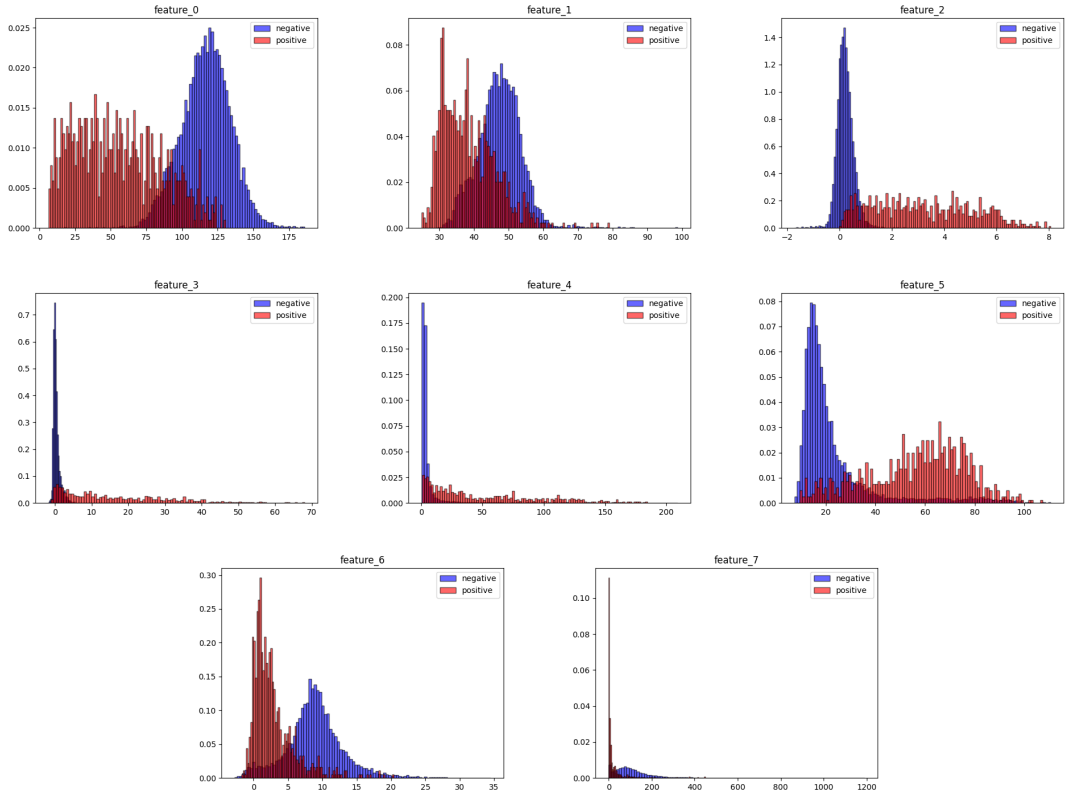
8. Skewness of the DM-SNR curve.

9. Class

HTRU 2 Summary
17,898 total examples.
1,639 positive examples.
16,259 negative examples

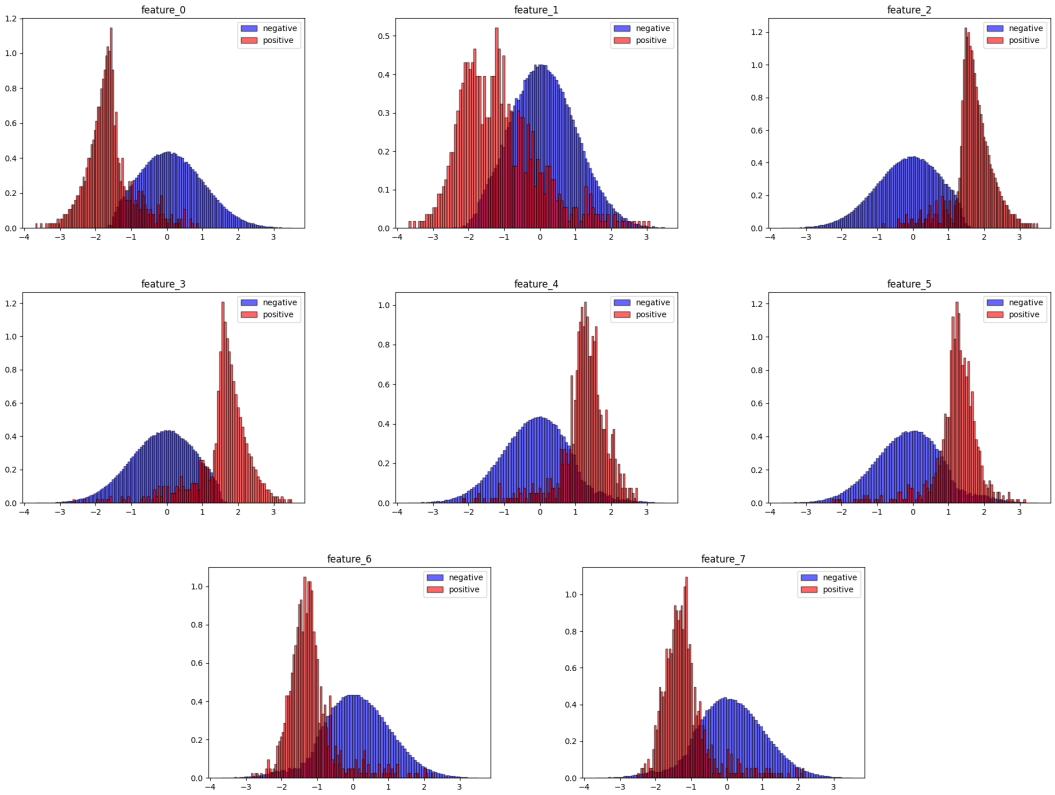
2 Features

Histogram of training dataset features. Red histograms refer to positive features, blue histograms to negatives..



2.1 Gaussinezed

A preliminary analysis shows that due to presence of outliers it can be useful gaussianize the features, especially for gaussian-based classifier. Gaussianization is a procedure that allows mapping a set of features to values whose empirical cumulative distribution function is well approximated by a gaussian cumulative distribution function

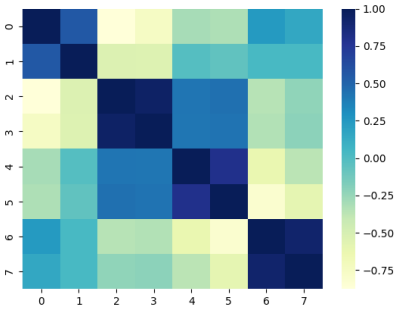


This method make all the features more gaussian-like, not the single class, in fact the overlapping remain.

2.2 Heat Map

Now we analyze the correlation between features using the Heat map. The heat map showing the absolute value of the Pearson correlation coefficient

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



The graphic shows a correlation between feature 2 and 3, and 6 and 7; we may benefit from using PCA = 6, however due to the results of the classifiers and due to the reduction of the analyzed data, we will see that PCA don't have benefits.

3 Classifiers's Introduction

Start from now we employ the K-Fold cross-validation(K-Fold=5), this method allows to have more data for training and validation. The 5-Fold split the dataset in 5 partitions, 4 for training and 1 for evaluation, this process is done 5 times in order to train the classifiers over the whole dataset. Data has been shuffled before splitting. Although the K-Fold is not useful for a huge dataset, we decide to use it, instead of single fold, because produce more accurate results.

We consider three different application, one uniform prior and two unbalanced applications:

$$(\widetilde{\pi}, C_{fp}, C_{fn}) = (0.5, 1, 1) \tag{1}$$
$$(\widetilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1) \tag{2}$$
$$(\widetilde{\pi}, C_{fp}, C_{fn}) = (0.9, 1, 1) \tag{3}$$

We measure performance in terms of normalized minimum detection costs, *minDCF*, which evaluate the classifier with the cost we would pay if we made optimal decisions. We will evaluate performances on the validation set.

4 Gaussian Classifiers

We start considering gaussian classifiers: MVG, MVG with Naive Bayes assumption, MVG wiht tied covariance and MVG with both Naive and tied assumption.
The Gausssian classifiers assume that our data, given the class, can be described by a Gaussian distribution:

$$X|C = c \sim N(\mu_c, \Sigma_c) \tag{4}$$

Tied: the tied covariance model assume that each class has its own mean μ_c, but the covariance matrix is same for all classes

Naive: the Naive Bayes assumption supposes that each component are indipendent. The naive Bayes Gaussian classifier corresponds to a Multivariate Gaussian classifier with diagonal covariance matrices.

CLASSIFIERS	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Gaussianezed features - no PCA			
MVG	0.247	0.154	0.710
TiedCovariance	0.514	0.139	0.760
NaiveBayes	0.278	0.152	0.604
NaiveBayesTied	0.490	0.161	0.819
Gaussianezed features - PCA(6)			
MVG	0.448	0.161	0.697
TiedCovariance	0.546	0.169	0.879
NaiveBayes	0.433	0.200	0.623
NaiveBayesTied	0.484	0.169	0.920
Gaussianized features - PCA(7)			
MVG	0.428	0.162	0.707
TiedCovariance	0.534	0.175	0.897
NaiveBayes	0.412	0.199	0.627
NaiveBayesTied	0.515	0.175	0.930
No Gaussianized features - PCA(6)			
MVG	0.315	0.169	0.659
TiedCovariance	0.498	0.180	0.852
NaiveBayes	0.646	0.219	0.826
NaiveBayesTied	0.339	0.212	0.826
No Gaussianized features - no PCA			
MVG	0.286	0.141	0.672
TiedCovariance	0.262	0.163	0.880
NaiveBayes	0.315	0.193	0.747
NaiveBayesTied	0.372	0.248	0.888

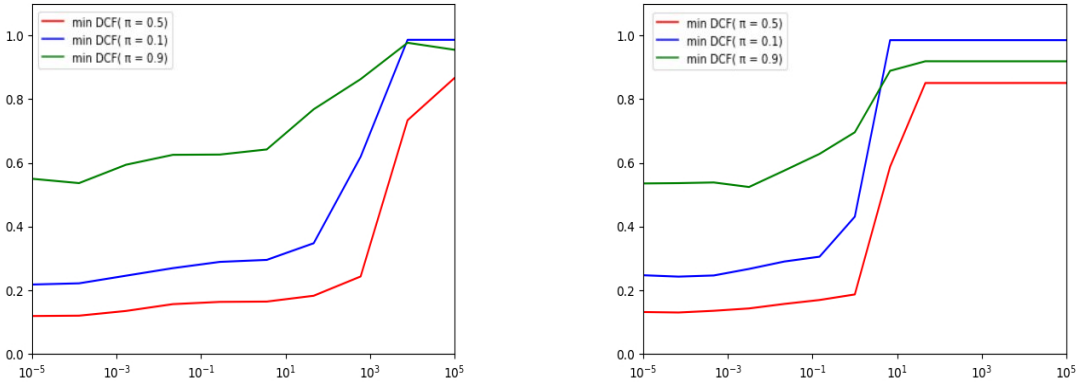
The best minDCF is the gaussinized TiedCovariance model with $\widetilde{\pi}$= 0.5 without PCA. Overall, the best results for the three applications, belong to gaussinized features without PCA. By comparing the result in the table, PCA is not effective, because for the No-Tied classifiers the difference between minDCF is huge. In addition, despite the reduction of dimensionality, the PCA results have an obvious difference with the No-PCA results.
The Full-Covariance models (*MVG and TiedCovariance*) perform better than the Diag-Cov models, since the features are correlated, so the off-diagonal values of Naive Bayes's covariance matrix can't be approximated to zero, so the Naive Bayes assumption don't get optimal results.
On the other hand, the Full-Covariance classifiers handle the correlation between features and get better results.

5 Linear Logistic Regression

Since classes are not balanced, we re-balance the costs of the different classes, minimizing

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

The first term is introduced because if the classes are linearly separable, the logistic regression solution is not defined. In fact, we can increase $\|w\|$, and changing accordingly the value of b. This objective function is less prone to overfitting. λ is a hyper-parameter that allows specifying the relative weight of the regularization term. The selection of good values for λ should be based on cross-validation.



(a) Raw features

(b) Gaussianized features

Figure 1: Linear Log-Reg with $\pi_T = 0.1$ changing λ and calculating the corresponding minDCF

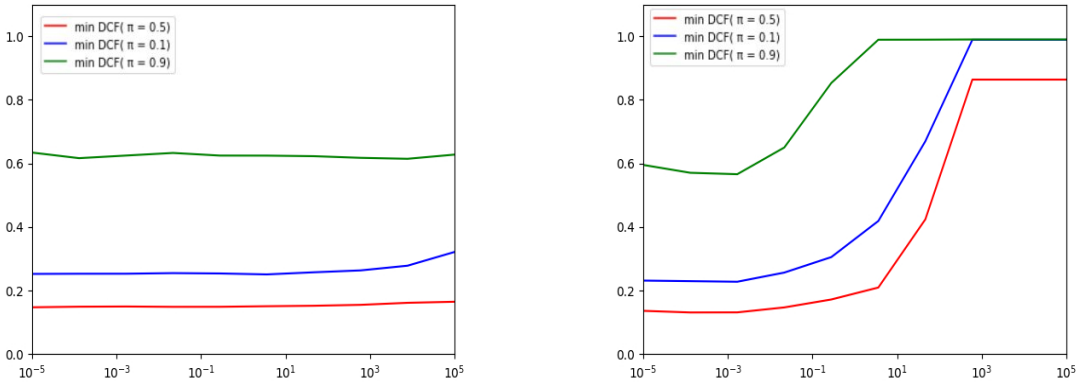
As we can see from the graphic the best λ is 10^-5

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Raw Features			
LogReg ($\lambda = 10^-5, \pi_T = 0.1$)	0.218	0.119	0.549
LogReg ($\lambda = 10^-5, \pi_T = 0.5$)	0.243	0.145	0.683
LogReg ($\lambda = 10^-5, \pi_T = 0.9$)	0.281	0.185	0.733
Gaussianezed features			
LogReg ($\lambda = 10^-5, \pi_T = 0.1$)	0.247	0.131	0.535
LogReg ($\lambda = 10^-5, \pi_T = 0.5$)	0.337	0.148	0.563
LogReg ($\lambda = 10^-5, \pi_T = 0.9$)	0.454	0.188	0.607

Overall, as we can see from the figure 1 and from the table, the Linear LogReg with $\pi_T = 0.1$ provide an improvement over the MVG Tied.
Since MVG correspond to quadratic separation rules, we repeat the analysis for Quadratic Logistic Regression; however, since the MVG Tied-Covariance and the Linear LogReg, which are linear model, perform better than the MVG model we expect that the Quad LogReg won't get good results.

6 Quadratic Logistic Regression

We can train LR model using feature vectors $\Phi(x)$ rather than x. We will obtain a model that has linear separation surface in the space defined by the mapping Φ, this space is called expanded feature space. Since expressions $w^T\Phi(x)+c$ correspond to quadratic forms in the original feature space, we are actually estimating quadratic separation surfaces.



(a) Raw features

(b) Gaussianized features

Figure 2: Quadratic Log-Reg changing λ and calculating the corresponding minDCF

As we can see from the graphic the best λ is 10^-5, and, despite the values of minDCF increase for bigger λ, in this case the gaussianization is more helpful. For the raw features graphic the choice of λ doesn't affect the minDCF.

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Raw Features			
QuadLogReg $(\lambda = 10^-5, \pi_T = 0.1)$	0.251	0.144	0.608
QuadLogReg $(\lambda = 10^-5, \pi_T = 0.5)$	0.251	0.144	0.608
QuadLogReg $(\lambda = 10^-5, \pi_T = 0.9)$	0.251	0.144	0.608
Gaussianezed features			
QuadLogReg $(\lambda = 10^-5, \pi_T = 0.1)$	0.213	0.130	0.530
QuadLogReg $(\lambda = 10^-5, \pi_T = 0.5)$	0.213	0.130	0.530
QuadLogReg $(\lambda = 10^-5, \pi_T = 0.9)$	0.213	0.130	0.530

The rebalancing of the model doesn't change the results, as we expect the Linear LogReg get better result than this model. From the results obtain until now, we will give more attention to the balance application $\widetilde{\pi} = 0.5$.

7 SVM

We now consider a classifier that allows us to give a geometrical interpretation of the regularization term: *Support Vector Machine*. SVM provide a natural way to achive non-linear separation without the need for an explicit expansion of our features. The output of SVMs cannot be directly interpreted as class posteriors.
The difference between SVM and Linear LogReg is that SVM has the concept of the margin, which is the distance of the closest point w.r.t. the separation hyperplane. SVM aim to maximize the margin and minimize the number of points that lie inside the margin.
There are two SVMs methods: *Linear*; *Kernel*, which it can be *Polynomial* or *Radial Basis Function*.

7.1 Linear

We start considering a linear model, to solve the SVM we consider the dual problem:

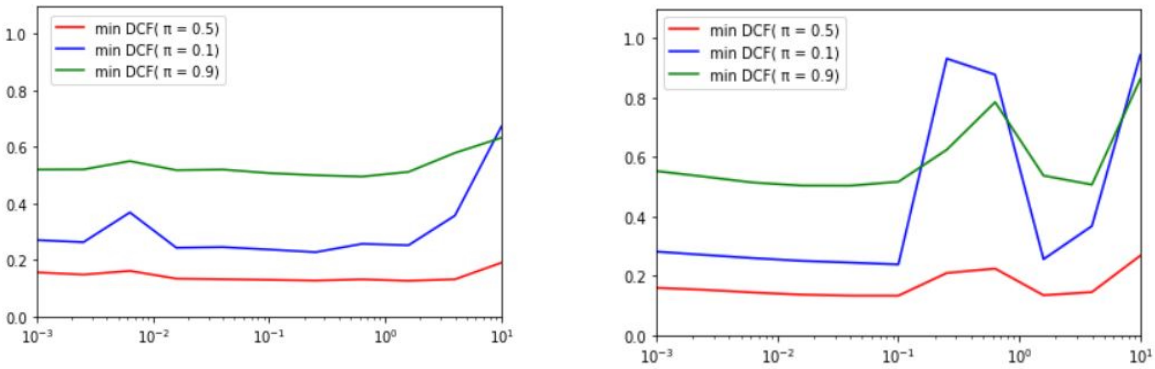
$$L_D(\alpha) = \alpha^T * 1 - \frac{1}{2}\alpha^T H \alpha$$
$$\text{s.t. } 0 \leq \alpha_i \leq C, \ \forall i = 1, ...n, \quad \sum_{i=1}^{n} a_i z_i = 0$$

where n is the number of training samples, C is an hyper-parameter, 1 is a n-dimensional vector of ones,

z_i is the class label for the i_{th} sample encoded as $z_i = \begin{cases} 1, & \textit{if xi belongs to class 1 (true pulsar signal)} \\ -1, & \textit{if xi belongs to class 0 (false pulsar signal)} \end{cases}$

and H is the matrix $H_{ij} = z_i z_j x_i^T x_j$
The dual formulation is differentiable, however it contains two constraints, as the L-BFGS algorithm can handle only box constraints, $0 \leq \alpha_i \leq C$.
We will have to modify the dual formulation, now we use the mapping $\hat{x} = \begin{bmatrix} x_i \\ K \end{bmatrix}$
with K = 0, the modified matrix $\hat{H} = z_i z_j (x_i^T x_j + 1)$

To re-balance the classes we use a different value of C, 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 $C_F = C\frac{\pi_F}{\pi_F^{emp}}$.
π_T^{emp} and π_F^{emp} are the empirical priors (i.e. sample proportions) for the two classes computed over the training set.



(a) Balancing $\pi_T = 0.5$ Gaussianized

(b) No Balancing Gaussianized

Figure 3: SVM linear changing C and calculating the corresponding minDCF

We can see from 3a that the best choice is C $\in [0.01, 1]$, instead form 3b, the best is C $\in [0.01, 0.1]$. From comparing the two graphic we notice that the rebalancing with $\pi_T = 0.5$ is the best.

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Raw features			
SVM Linear (C = 1, NoRebalancing)	0.242	0.146	0.818
SVM Linear (C = 0.01, $\pi_T = 0.1$)	0.321	0.181	0.870
SVM Linear (C = 0.01, $\pi_T = 0.9$)	0.895	0.411	0.900
SVM Linear (C = 0.01, $\pi_T = 0.5$)	0.241	0.134	0.515
Gaussianized features			
SVM Linear (C = 0.01, $\pi_T = 0.1$)	0.262	0.147	0.527
SVM Linear (C = 0.01, $\pi_T = 0.5$)	0.247	0.137	0.515
SVM Linear (C = 0.01, $\pi_T = 0.9$)	0.796	0.436	0.885
SVM Linear (C = 1, $\pi_T = 0.5$)	0.261	0.130	0.514

The best result is the Gaussianized SVM Linear (C = 1, $\pi_T = 0.5$), in fact the most accurate minDCF are calculated with Gaussianization. In addition the $\pi_T = 0.5$ rebalancing, which we use for the figure 3a, give the best result both for Guassianized and No-Gaussianized features

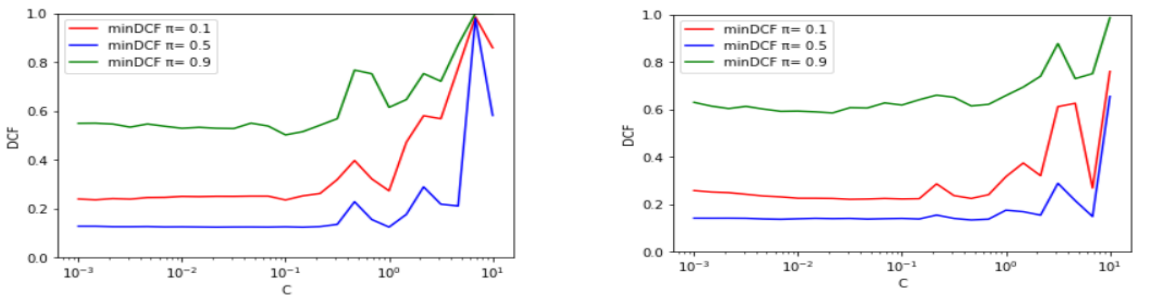
7.2 Kernel polynomial

We now analyse the non-linear formulations.
If we have a function that efficiently computes the dot-products in the expanded space

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

then both training and scoring can be performed by using only k.
Matrix H is given $H_{ij} = z_i z_j k(x_i, x_j)$
In practice, we are computing a linear separation surface on the expanded space, which corresponds to a non-linear separation surface in the original feature space.
In the case of polynomial kernels we define

$$k(x_1, x_2) = (x_1^T x_2 + c)^d$$



(a) Balancing $\pi_T = 0.5$ Gaussianized c = 10 (b) No Balancing No Gaussianized c = 10

Figure 4: SVM Kernel poly changing C and calculating the corresponding minDCF

We notice that a good value for C is $\in [0.001, 0.1]$ and for the application $\widetilde{\pi} = 0.5$ the balancing with $\pi_T = 0.5$ and gaussianized is slightly better.

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Gaussianized features			
SVM Poly (C = 0.1, $c_{small} = 1$, $\pi_T = \pi_{emp}$)	0.290	0.163	0.936
SVM Poly (C = 0.1, $c_{small} = 1$, $\pi_T = 0.5$)	0.306	0.165	0.895
SVM Poly (C = 0.01, $c_{small} = 10$, $\pi_T = 0.5$)	0.240	0.123	0.531
SVM Poly (C = 0.01, $c_{small} = 10$, $\pi_T = 0.1$)	0.236	0.152	0.544

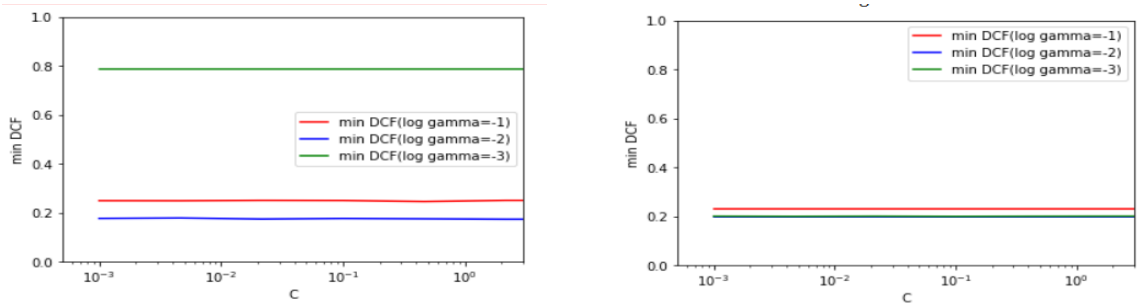
The best result is SVM Kernel polynomial ($C = 0.01, c_{small} = 10, \pi_T = 0.5$); we reject the result of PCA and of No-Gaussinized feature because the values are not optimal. After computing cross-validation to decide the value of c_{small}, we select 10 as the best choice.

7.3 Kernel RBF

In the case of Radial Basis Function the function kernel is

$$k(x_1, x_2) = e^{-\gamma ||x_1 - x_2||^2}$$

γ defines the width of the kernel, with small γ the kernel is wide (S.V. influences most other points), with a large one the kernel is narrow (S.V. has very small influence on points that are not close)



(a) Balancing $\pi_T = 0.5$ Gaussianized $\widetilde{\pi} = 0.5$ (b) NoBalancing No-Gaussianized $\widetilde{\pi} = 0.5$

Figure 5: SVM Kernel RBF changing C and calculating the corresponding minDCF

We can see from 5b that the values for γ=0.01 and γ=0.001 are overlapped, from 5a we decide to choose $\gamma = 0.01$ because give the best minDCF. As there are constants trends, the choice of C doesn't effect the best value of minDCF

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Raw features			
RBF Kernel (C = 1000, $\gamma = 0.1$, $\pi_T = 0.1$)	0.326	0.232	0.819
RBF Kernel (C = 1000, $\gamma = 0.1$, $\pi_T = 0.5$)	0.326	0.232	0.819
RBF Kernel (C = 1000, $\gamma = 0.1$, $\pi_T = 0.9$)	0.326	0.232	0.819
RBF Kernel (C = 1000, $\gamma = 0.01$, $\pi_T = 0.5$)	0.298	0.193	0.907
Gaussianized features			
RBF Kernel (C = 100, $\gamma = 0.01$, $\pi_T = 0.1$)	0.306	0.173	0.585
RBF Kernel (C = 100, $\gamma = 0.01$, $\pi_T = 0.9$)	0.359	0.198	0.630
RBF Kernel (C = 1000, $\gamma = 0.01$, No Balancing)	0.506	0.250	0.612
RBF Kernel (C = 1000, $\gamma = 0.01$, $\pi_T = 0.5$)	0.250	0.143	0.768
RBF Kernel (C = 1000, $\gamma = 0.01$, $\pi_T = 0.9$)	0.511	0.250	0.623

The best result is SVM Kernel RBF (C = 1000, $\gamma = 0.01$, $\pi_T = 0.5$).
We decide to use only the balanced model because it provide smaller results; at the end, analyzing the minDCF, the Poly SVM achieves slightly better results than SVM Kernel RBF.

8 Gaussian Mixture Model

Now we turn our attention to the last classifier: *GMM*. A GMM is a density model obtained as a weighted combination of Gaussians:

$$X \sim GMM(M, \Sigma, \Pi) \Rightarrow f_X(x) = \sum_{c=1}^{k} w_c N(x; \mu_c, \Sigma_c) \tag{5}$$

Expression 5 is an example of a K-components Gaussian Mixture Model.
This model is a generative approach based on training a GMM over the data of each class. Although our training set cannot be well modeled by a Gaussian distribution, we can imagine that the set can be partitioned into subset, in such a way that the distribution of the points of each component can be modeled by a Gaussian, so we want to estimate the cluster assignments and model parameters as to maximize the *marginal distribution* of the data.
We consider both full covariance and diagonal model, with and without covariance tying. For tied covariance models, tying takes place at class level (i.e. different classes have different covariance matrices)

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Full model Gaussianized			
GMM 4Gau	0.258	0.131	0.527
GMM 8Gau	0.252	0.128	0.528
GMM 16Gau	0.247	0.128	0.565
GMM 32Gau	0.255	0.129	0.602
GMM 64Gau	0.277	0.156	0.701
Full Tied Gaussianized			
GMM 4Gau	0.249	0.144	0.679
GMM 8Gau	0.313	0.156	0.711
GMM 16Gau	0.330	0.167	0.720
GMM 32Gau	0.326	0.171	0.733
GMM 64Gau	0.333	0.172	0.734
Diagonal model Gaussianized			
GMM 4Gau	0.287	0.157	0.638
GMM 8Gau	0.285	0.160	0.659
GMM 16Gau	0.279	0.157	0.658
GMM 32Gau	0.286	0.151	0.639
GMM 64Gau	0.284	0.146	0.641
Diagonal tied Gaussianized			
GMM 4Gau	0.302	0.189	0.723
GMM 8Gau	0.297	0.187	0.688
GMM 16Gau	0.291	0.167	0.636
GMM 32Gau	0.292	0.149	0.653
GMM 64Gau	0.292	0.151	0.656

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
Full model Raw feature			
GMM 4Gau	0.279	0.162	0.660
GMM 8Gau	0.251	0.144	0.644
GMM 16Gau	0.250	0.140	0.718
GMM 32Gau	0.272	0.158	0.796
GMM 64Gau	0.284	0.176	0.951
Full Tied Raw feature			
GMM 4Gau	0.298	0.159	0.710
GMM 8Gau	0.313	0.156	0.711
GMM 16Gau	0.330	0.167	0.719
GMM 32Gau	0.326	0.171	0.712
GMM 64Gau	0.333	0.172	0.734
Diagonal model Raw feature			
GMM 4Gau	0.309	0.191	0.763
GMM 8Gau	0.298	0.184	0.719
GMM 16Gau	0.285	0.168	0.669
GMM 32Gau	0.271	0.154	0.739
GMM 64Gau	0.283	0.162	0.775
Diagonal tied Raw feature			
GMM 4Gau	0.297	0.165	0.736
GMM 8Gau	0.300	0.169	0.760
GMM 16Gau	0.311	0.163	0.722
GMM 32Gau	0.319	0.162	0.737
GMM 64Gau	0.321	0.164	0.762

We can see that the best model is GMM 16 Gau Full model Gaussianized, we can predicted that diagonal doesn't perform well because the feature are correlated, so the off-diagonal values of covariant matrix is not zero.
As we are using Guassianized models, Gaussianization has a better effect on the minDCF, it reduce the dynamic range of sample far from the data mean and thus reduce the separability of some clusters. Starting from 64 gaussians, the trend of the minDCF increase drastically, due to overfitting.

9 Comparisons

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
TiedCovariance Gauss	0.514	0.139	0.760
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.218	0.119	0.549
QuadLogReg Gauss ($\lambda = 10^-5, \pi_T = 0.1$)	0.213	0.130	0.530
SVM Linear Gauss (C = 1, $\pi_T = 0.5$)	0.261	0.130	0.514
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.240	0.123	0.531
RBF Kernel Gauss (C = 1000, $\gamma = 0.01$, $\pi_T = 0.5$)	0.250	0.143	0.768
GMM 16Gau Gauss	0.247	0.128	0.565

We are searching the two best results of minDCF, as we can see from the table, we select SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$) and LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$).

10 Score Calibration

Until now we consider only the minDCF, which means don't deal with the choice of optimal threshold. The cost that we actually pay, however, depends on the goodness of the threshold we use to perform class assignment. We therefore start considering actualDCF. If the score are well calibrated, the optimal threshold is the theoretical threshold

$$t = -log\frac{\widetilde{\pi}}{(1-\widetilde{\pi})}.$$

Otherwise there are two options:

- Taking the threshold corresponding to the minimum of the DCF on validation set.
- Compute a transformation function that maps the classifiers scores s to well-calibrated score $s_{cal} = f(s)$.

The first approach consist in:

- Pool the K-fold scores of the classifier.
- Shuffle the scores and then split the shuffled scores into 2 partitions.
- Estimate the threshold on one partition, apply the threshold on the remaining one and compare minimum and actual DCFs over the latter.

CLASSIFIER	*min DCF*	*act DCF* $t = -log\frac{\widetilde{\pi}}{(1-\widetilde{\pi})}$	*act DCF* t^*
$\widetilde{\pi} = 0.1$			
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.244	0.499	0.260
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.206	0.228	0.248
$\widetilde{\pi} = 0.5$			
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.129	0.137	0.136
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.116	0.128	0.119
$\widetilde{\pi} = 0.9$			
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.507	0.963	0.573
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.539	0.608	0.594

The new threshold looks accurate for all the applications. We can see that the actDCF with the theoretical threshold are worse for the unbalanced applications.

For the second we assume that f is linear in s $f(s) = \alpha s + \beta$. Since $f(s)$ should produce well-calibrated scores, the function can be interpreted as the log-likelihood ratio for the two class hypotheses

$$f(s) = log\frac{f_{S|C}(s|H_T)}{f_{S|C}(s|H_F)} = \alpha s + \beta \tag{6}$$

and the class posterior probability for prior $\widetilde{\pi}$ corresponds to

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

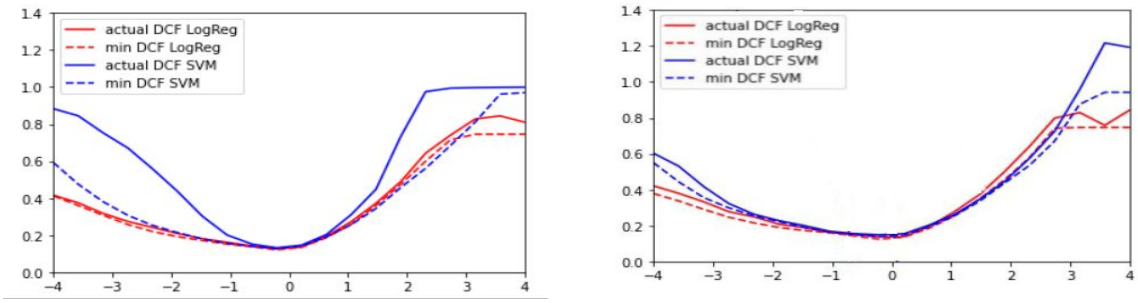
The log posterior ratio of the Logistic regression model has a very similar expression ($w^T x + \beta$) to the (6), interpreting the scores as features.
If we let $\beta' = \beta + log\frac{\tilde{\pi}}{1-\tilde{\pi}}$, then we have exactly the same model and we can exploit the prior-weighted Logistic Regression model to learn the model parameters α, β over our training scores.
To recover the calibrated score $f(s)$ we will need to compute:

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

	min DCF ($\tilde{\pi}$ = 0.1)	*min DCF ($\tilde{\pi}$ = 0.5)*	*min DCF ($\tilde{\pi}$ = 0.9)*
SVM Poly	0.244	0.129	0.507
LogReg	0.206	0.116	0.539
	LogReg act DCF ($\tilde{\pi}$ = 0.1)	*LogReg act DCF ($\tilde{\pi}$ = 0.5)*	*LogReg act DCF ($\tilde{\pi}$ = 0.9)*
Log-Reg π_T=0.1	0.233	0.128	0.620
Log-Reg π_T=0.5	0.341	0.195	0.962
	SVM Poly act DCF ($\tilde{\pi}$ = 0.1)	*SVM Poly act DCF ($\tilde{\pi}$ = 0.5)*	*SVM Poly act DCF ($\tilde{\pi}$ = 0.9)*
Log-Reg π_T=0.1	0.253	0.139	0.538
Log-Reg π_T=0.5	0.487	0.182	0.912

As we can see from the results, for both the classifiers, the actDCFs calculated with calibration of the score don't provide optimal values. On the other hand, the first approach produce better results, for the two classifiers.
Now we check our assumption using **bayes error plot** and **DET plot**.

10.1 Bayes Error Plot



(a) Non calibrated scores

(b) Calibrated scores

Figure 6: Bayes Error Plot of SVM Poly and Log-Reg

How we expect from the data, the calibrated plot shows an improvement, in fact the curve have similar values of minDCF and actDCF. Particularly in the middle of the graphs, the minDCF trend and the actDCF trend, for both classifiers, are overlapped. Instead, for external values of theoretical threshold, the two curves tend to separate; this phenomenon is worse in the first graph.

10.2 DET plot

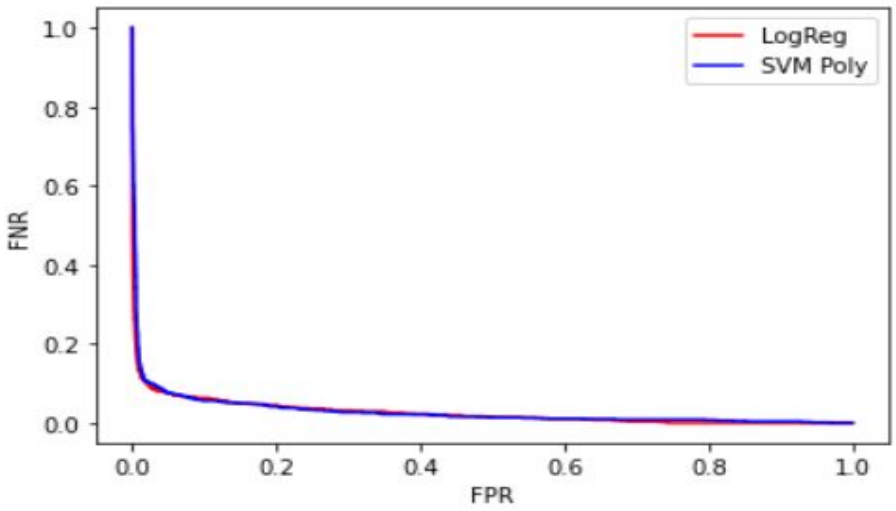


Figure 7: DET plot of LogReg and SVM Poly

The trends are really close, because we chose the two best classifiers, which give similar results of minDCF.
However, if we had to choose the best classifier, even if only slightly, the Log-Reg model is better.

11 Fusion

In order to increase the classification model, we will combine our two best classifier.
An effective fusion approach consist in performing score-level fusion, rather than decision-level voting. We assume that the fused score is a function of the scores of the different classifiers.
If $s_{t,A}$ and $s_{t,B}$ are the scores of the classifiers A and B for sample x_t, the the fused score for sample x_t will be

$$s_t = f(s_{t,A}, s_{t,B})$$

We will then use the fused score s_t to perform the decision.

As for calibration, we can assume a simple linear form for f:

$$f(s_{t,A}, s_{t,B}, s_{t,C}, ...) = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \alpha_C s_{t,C} + ... + \beta$$

Again, we can use prior-weighted logistic regression to train the model parameters. The resulting score will, hopefully, have higher accuracy and benefit from re-calibrating effects of the model.
We will use SVM Polynomial Gaussinized (C = 0.01, $c_{small} = 10$, $\pi_T = 0.5$) and LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$). For training of model parameters we choose the prior-weighted logistic regression with $\pi_T = 0.1$ for both the classifiers.

11.1 DET plot

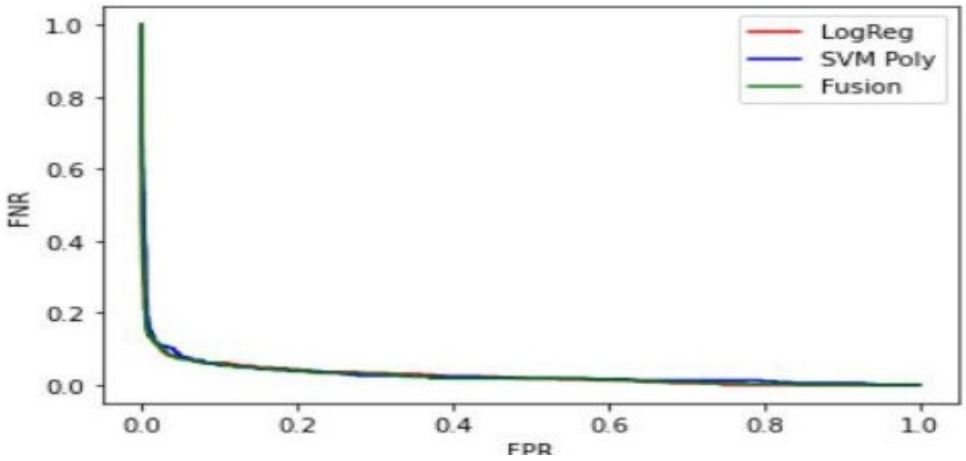


Figure 8: DET plot of LogReg and SVM Poly and Fusion

The plot differs from the previous DET plot since we are evaluating on different subset of samples (we use the validation portion to plot the DET). The fusion was done using the training portion of the two classifiers.
From the Figure 8, we can see that the Fusion has a similar trend, but slightly better than the others.

11.2 Bayes Error Plot

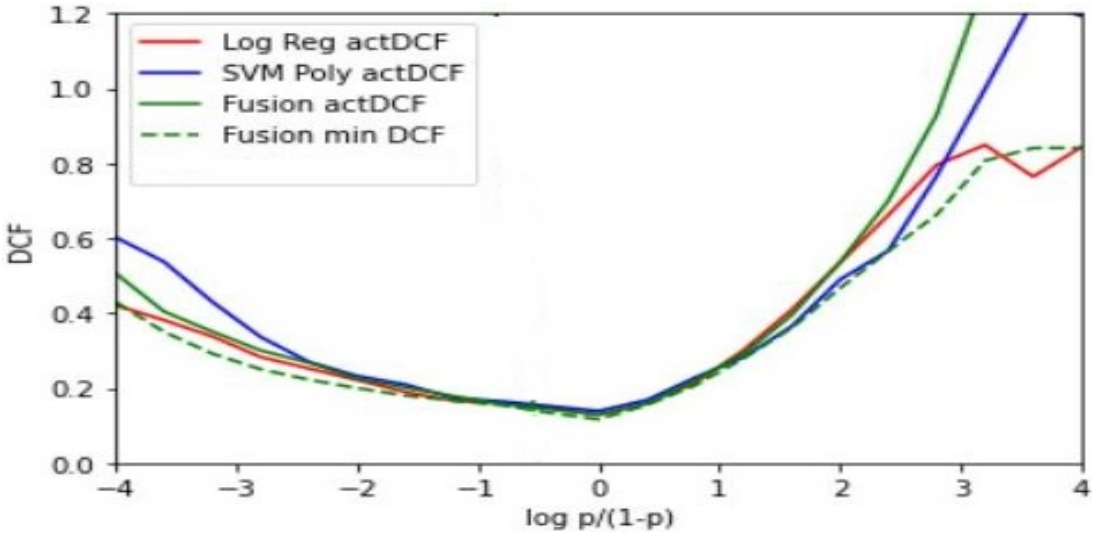


Figure 9: Bayes Error Plot of LogReg and SVM Poly and Fusion

	$\widetilde{\pi} = 0.1$		$\widetilde{\pi} = 0.5$		$\widetilde{\pi} = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
SVM Poly	0.244	0.253	0.129	0.136	0.507	0.538
LogReg	0.206	0.233	0.116	0.119	0.539	0.594
Fusion	0.212	0.242	0.116	0.129	0.520	0.645

As we can see from the table and from Figure 9 the fusion model doesn't provide optimal result for minDCF.
Finally, this approach is not effective in reducing minimum costs, because for the unbalanced applications the two classifiers perform better than the fusion, and for the balanced application doesn't provide an improvement.
Untill now, our best classifier is the Log-Reg model.

12 Experimental results

We now need to assess the quality of our model on test set in terms of minDCF. This allows veryfing whether the proposed solutions are consistent on unseen data. Differently from the previous data set we decide not to use a K-Fold validation, but re-train the data with the whole training set and evaluate with the test set.

CLASSIFIER	$\widetilde{\pi} = 0.1$	$\widetilde{\pi} = 0.5$	$\widetilde{\pi} = 0.9$
MVG Gauss	0.241	0.151	0.622
TiedCovariance Gauss	0.478	0.166	0.818
NaiveBayes Gauss	0.285	0.154	0.574
NaiveBayesTied Gauss	0.478	0.166	0.818
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.198	**0.111**	0.529
LogReg Gauss ($\lambda = 10^-5, \pi_T = 0.1$)	0.211	0.125	0.504
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.5$)	0.207	0.122	0.562
QuadLogReg Gauss ($\lambda = 10^-5, \pi_T = 0.1$)	0.204	0.117	0.464
QuadLogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.246	0.141	0.611
SVM Linear Gauss (C = 1, $\pi_T = 0.5$)	0.214	0.124	0.517
SVM Linear Raw Feature (C = 1, $\pi_T = 0.5$)	0.272	0.160	0.747
SVM Linear Gauss (C = 1, $\pi_T = 0.1$)	0.219	0.123	0.525
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.224	**0.112**	0.485
RBF Kernel Gauss (C = 1000, $\gamma = 0.01$, $\pi_T = 0.5$)	0.324	0.169	0.571
GMM Full 16Gau Gauss	0.253	0.126	0.550
GMM Full Tied 8Gau Gauss	0.249	0.143	0.615
GMM Diagonal 8Gau Gauss	0.277	0.155	0.649
GMM Diagonal Tied 16Gauss	0.299	0.175	0.643
GMM Full 4Gau Gauss	0.243	0.122	0.494

Table 1: Table with minDCFs over all test set.

We can see from Table 1 that our results are consistent with the ones obtained before. Therefore, there are no relevant differences between the evaluation and the training population, except for the Guassian classifiers, which have a worsening of minDCF.
As we decided for the training set, we analyze only no-PCA models for the same reason said before.

12.1 Score Calibration

As we said before, the minimum DCF measure the potential capabilities of our system to produce good decision, so for a most realistic approach we need actDCF. We will use the two approaches (chapter 10): calibrated score, calculation of best threshold.

Our best model, like before, are LogReg Raw Feature($\lambda = 10^-5, \pi_T = 0.1$) and SVM Poly Gaussianized(C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)

CLASSIFIER	*min DCF*	*act DCF* $t = -log\frac{\widetilde{\pi}}{(1-\widetilde{\pi})}$	*act DCF* t^*
$\widetilde{\pi} = 0.1$			
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.222	0.541	0.226
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.198	0.232	0.211
$\widetilde{\pi} = 0.5$			
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.103	0.103	0.119
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.103	0.106	0.105
$\widetilde{\pi} = 0.9$			
SVM Poly Gauss (C=0.01,$c_{small} = 10$,$\pi_T = 0.5$)	0.426	0.974	0.491
LogReg Raw Feature ($\lambda = 10^-5, \pi_T = 0.1$)	0.444	0.504	0.478

	min DCF ($\widetilde{\pi} = 0.1$)	*min DCF ($\widetilde{\pi} = 0.5$)*	*min DCF ($\widetilde{\pi} = 0.9$)*
SVM Poly	0.222	0.103	0.426
LogReg	0.198	0.103	0.444
	LogReg act DCF ($\widetilde{\pi} = 0.1$)	*LogReg act DCF ($\widetilde{\pi} = 0.5$)*	*LogReg act DCF ($\widetilde{\pi} = 0.9$)*
Log-Reg π_T=0.1	0.234	0.106	0.468
	SVM Poly act DCF ($\widetilde{\pi} = 0.1$)	*SVM Poly act DCF ($\widetilde{\pi} = 0.5$)*	*SVM Poly act DCF ($\widetilde{\pi} = 0.9$)*
Log-Reg π_T=0.1	0.226	0.104	0.529

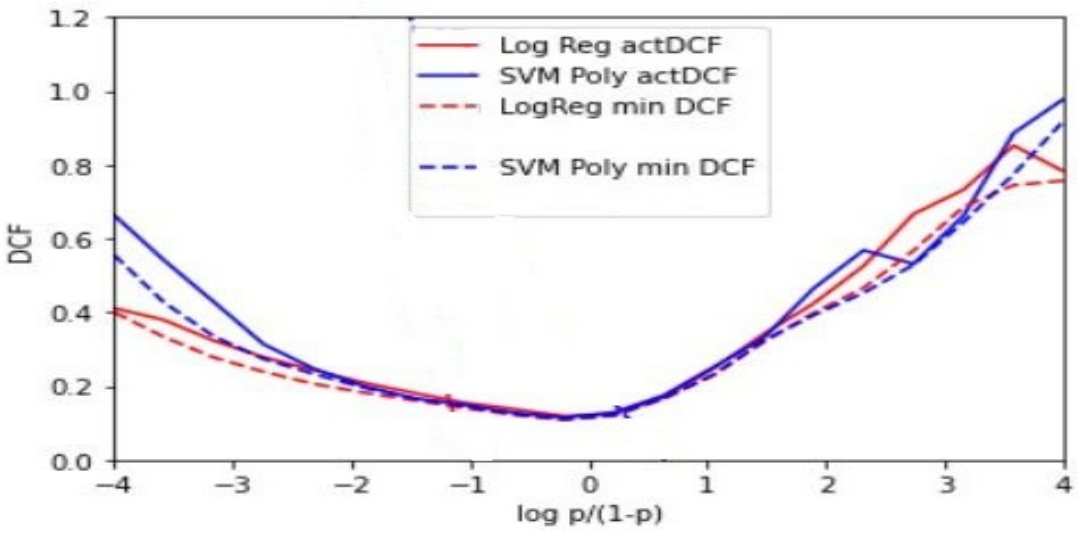
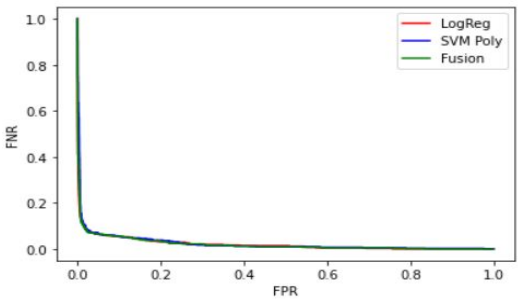


Figure 10: Bayes Error Plot of LogReg and SVM Poly

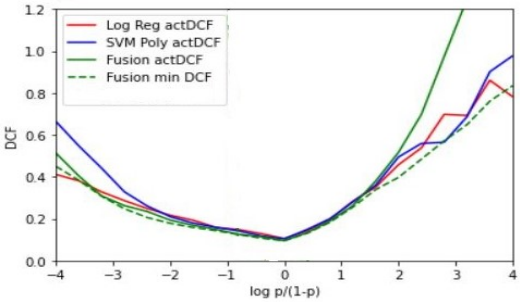
As we expect from the first part of the project, for the unbalanced applications the first approach is better, on the contrary for the uniform prior application the second approach produce optimal results.
This assumption is supported by the fact that, as we can see from Figure 10, for external values of threshold the actDCF's curves are breaking away from minDCF's curves.

12.2 Fusion

Although the fusion calculated on the training set provide not-optimal result, we choose to computing anyway the fusion, in order to confirm our hypothesis.



(a) DET plot



(b) Bayes error plot

Figure 11: Bayes Error Plot and DET plot of SVM Poly, Log-Reg and fusion

	$\widetilde{\pi} = 0.1$		$\widetilde{\pi} = 0.5$		$\widetilde{\pi} = 0.9$	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
SVM Poly	0.222	0.226	0.103	0.119	0.426	0.491
LogReg	0.198	0.211	0.103	0.105	0.444	0.478
Fusion	0.191	0.216	0.095	0.096	0.436	0.600

The comparisons between the three classifiers for different application is the same computed before, particularly the balanced application provide an optimal result, on the other hand the others application don't improve the minDCF.
In the case of $\widetilde{\pi} = 0.5$ the actDCF is very close to the minDCF, which means that we are able to select an optimal threshold.

13 Conclusion

Concluding, the fusion of two subsystem is effective and produce well-calibrated scores for the primary application $\widetilde{\pi} = 0.5$; we are able to achieve a DCF cost of ≈ 0.1.
However, the model are relatively ineffective for applications with imbalanced costs and priors proportions. For this reason, we select the Logistic Regression Raw Feature ($\lambda = 10^{-}5, \pi_T = 0.1$) model , computed before, thanks to which we are able to achieve a DCF cost of ≈ 0.1 and ≈ 0.45, respectively for $\widetilde{\pi} = 0.1$ and $\widetilde{\pi} = 0.9$ applications.
Overall, the similarity between validation and evaluation results suggests that the choice we made on training sets proved effective also for the evaluation data.
These results led us to think we have selected suited model for HTRU2 dataset.

References

[1] A D Cameron et al. "The High Time Resolution Universe Pulsar Survey – XVI. Discovery and timing of 40 pulsars from the southern Galactic plane". In: *Monthly Notices of the Royal Astronomical Society* 493.1 (Jan. 2020), pp. 1063–1087. DOI: 10.1093/mnras/staa039. URL: https://doi.org/10.1093%2Fmnras%2Fstaa039.