# Binary Classification of Pulsar Detection

Francesco Sorrentino
*Polytechnic of Turin*
*Artificial Intelligence and Data Analytics*
s301665

Francesco Di Gangi
*Polytechnic of Turin*
*Artificial Intelligence and Data Analytics*
s301793

**Abstract**

In this report is shown how some of the most common Machine Learning (ML) algorithms in order to classify and compare the results of the HTRU2 dataset. The starting point will be the feature analysis and feature correlation, then it will show the result of different classifiers using normalization techniques (Gaussian and Z Score normalization) in order to lead us to choose the best-fit model.

## 1 Problem Overview

The data set used in this project is called *HTRU2*, and it describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [1]. 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.

In this report it is possible to see how **Machine Learning** tools are now being used automatically label pulsar candidates to facilitate rapid analysis, especially in the *classification* field.

This report is about binary classification problems, reason why the data set is divided in two classes:

- 1,639 legitimate pulsar examples are a minority positive class, real pulsar examples
- 16,259 spurious examples the majority negative class, caused by RFI/noise

for a total of 17,898 examples, that are split into training set (8929 samples) and test set (8969 samples).

Also, each candidate is described by eight continuous variables:

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.

The first four are simple statistics obtained from the integrated pulse profile (folded profile), that is an array of continuous variables describing a longitude-resolved version of the signal that has been averaged in both time and frequency, meanwhile the remaining four variables are similarly obtained from the DM-SNR curve.

The data are normalized following the *Z-Score Normalization*

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

where $x_i$ indicates the observed value for the i-th sample, and $\mu$ is the mean vector of samples, ending with *sigma* that represents the standard deviation.

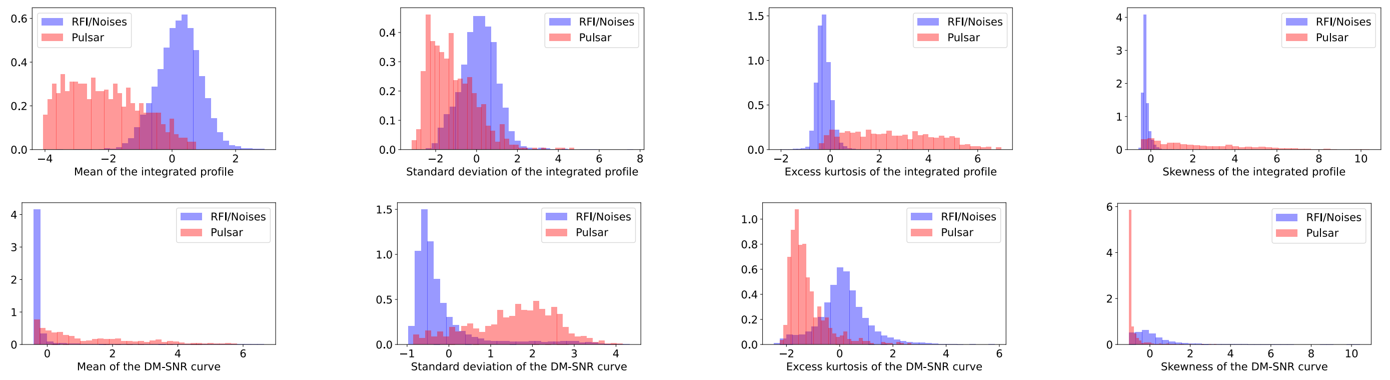


Figure 1: Z-Score Normalized Features

The feature with the *Z-Score Normalization* seem already well distributed, as shown on figure 1. Since we'll use Gaussian classifiers we can try to apply a Gaussianization over the features, but we expect worse result because of the already normalized distribution with no evident outliers.

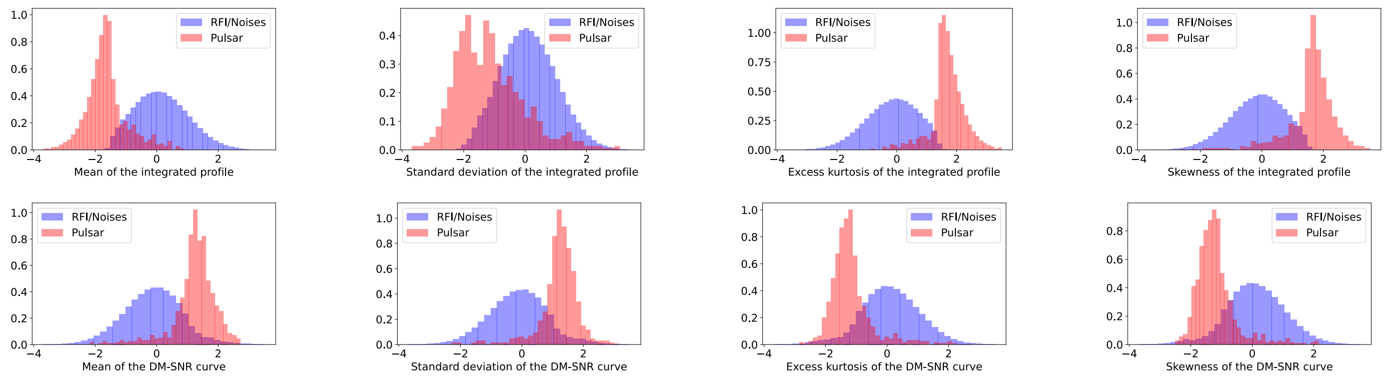


Figure 2: Gaussianized features (quantile transformation)

The heatmap over, calculated with the absolute value of the Pearson correlation coefficient on *Z-Score Normalization*

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

illustrates the correlation between features:

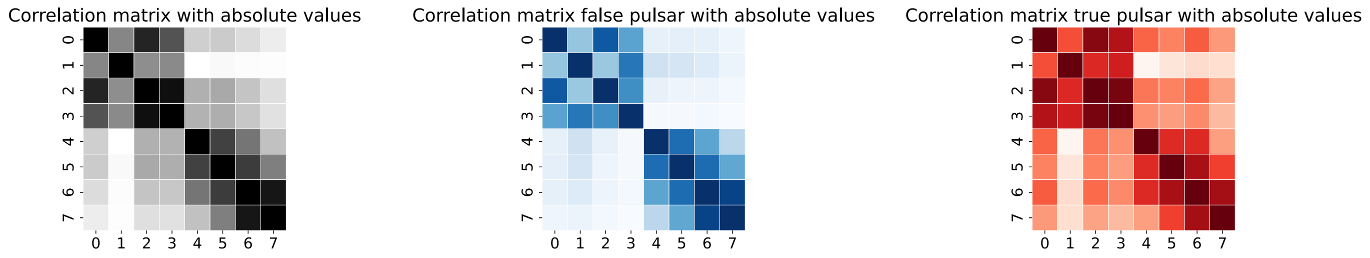


Figure 3: Heatmap - Pearson correlation

As we can see, the first 4 features (0,1,2,3) and the other 4 (4,5,6,7) are not so correlated, this can be connected to the fact that are extracted from 2 different curves (integrated profile and DM-SNR). There is strong correlation between features 6-7 and between features 2-0, a dimensionality reduction technique such as PCA could be useful to reduce features to 7 dimensions or 6, however it is possible to see that there is a moderate correlation between features 2-3(weak for the false, strong for the true), reason why it could be possible to proceed with a PCA of 5 dimensions.

## 2 Classifying HTRU2 features

The approach that is followed on the following cases is *K Fold* with 3 folds, where there will be considered three applications.

- a balanced one:

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

- two unbalanced:

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

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

Since the application target is to choose the most promising approach, we will measure performance in terms of normalized **minimum** Detection Cost Function (DCF) - it measures the cost we would pay if we made optimal decisions using the recognizer scores.

### 2.1 Gaussian classifiers

We start considering the MultiVariate Gaussian (MVG) classifiers: Full Covariance, Diagonal Covariance and Tied Covariance. These are generative model that assume Gaussian distributed data, given the class:

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

The Tied MVG classifier assumes that each class has its own $\mu_c$, yet the covariance is shared.

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

The Diagonal, instead, assumes that the covariance matrices are diagonal.

**MVG Classifiers - minDCF on the evaluation set**

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - no PCA | | | |
| MVG - Full Covariance | 0.142 | 0.285 | 0.660 |
| MVG - Diag Covariance | 0.191 | 0.315 | 0.735 |
| MVG - Tied Covariance | 0.112 | 0.224 | 0.580 |
| Z-Norm features - PCA (m = 7) | | | |
| MVG - Full Covariance | 0.140 | 0.304 | 0.628 |
| MVG - Diag Covariance | 0.213 | 0.510 | 0.715 |
| MVG - Tied Covariance | 0.112 | 0.223 | 0.574 |
| Z-Norm features - PCA (m = 6) | | | |
| MVG - Full Covariance | 0.151 | 0.290 | 0.635 |
| MVG - Diag Covariance | 0.222 | 0.534 | 0.726 |
| MVG - Tied Covariance | 0.138 | 0.254 | 0.584 |
| Gaussianized features - no PCA | | | |
| MVG - Full Covariance | 0.153 | 0.243 | 0.696 |
| MVG - Diag Covariance | 0.154 | 0.280 | 0.600 |
| MVG - Tied Covariance | 0.132 | 0.233 | 0.541 |

We used, as said above, three different applications, and applied PCA on *Z Score Normalized* data. The Tied Covariance obtains the best performance in *K fold cross validation protocol* both with and without PCA.

It is possible to notice that PCA is not quiet effective in none of the three classifiers, but in the Tied one with m=7 we can see that it preserves the performance for our target application obtaining slightly better results for the unbalanced applications. Applying the PCA with 6 dimensions, it is also possible to notice that we have a worse performance, so we stopped at 6 dimensions.

As we have assumed before, applying Gaussianization of the features led to worse results.

The Diag model performs, in general, worse. We expected that because features are correlated in group of fours, and the Naive Bayes assumption led to bad results. In its case, the Gaussianization improve performances, but it is still the worse in all the four cases

## 2.2 Logistic Regression Classifiers

It is a discriminative model. We will employ two types of Logistic Regression: *Linear Logistic Regression* and *Quadratic Logistic Regression.*

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

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

Where $(w, b)$ are the model parameters. The model assumes that decision rules are linear hyper-planes orthogonal to $w$.

To estimate a proper value for the hyper-parameter $\lambda$ we also used a 3-fold cross-validation; we consider our main application, so we set $\tilde{\pi_T} = 0.5$.

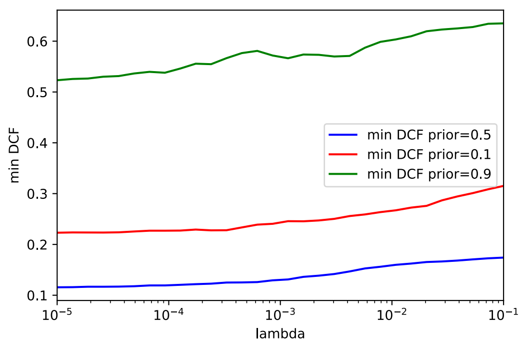


Figure 4: Linear Logistic Regression lambda tuning

As it is possible to see from the figure above, the proper value for the hyper-parameter $\lambda$ would be $\lambda = 10^{-5}$. We can also consider training using a different prior $\pi_T$ to see the effects on the other application, as we did over.

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - no PCA | | | |
| MVG - Tied Covariance | 0.112 | 0.224 | 0.580 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.116 | 0.223 | 0.523 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.114 | 0.217 | 0.547 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.9$) | 0.123 | 0.226 | 0.523 |
| Z-Norm features - PCA (m = 7) | | | |
| MVG - Tied Covariance | 0.112 | 0.223 | 0.574 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.116 | 0.220 | 0.532 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.115 | 0.218 | 0.551 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.9$) | 0.120 | 0.222 | 0.541 |
| Gaussianized features - no PCA | | | |
| MVG - Tied Covariance | 0.132 | 0.233 | 0.541 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.124 | 0.232 | 0.517 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.128 | 0.230 | 0.519 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.9$) | 0.127 | 0.235 | 0.510 |

We tested the application first with *Z-Score Normalization*, without applying PCA, data, then with *Z-Score Normalization* with a PCA of 7 dimensions applied, and at the end with *Gaussianized features*.

Overall, the MVG Tied Covariance performs better than the logistic regression.

Balancing the training priors, we can see that the Logistic Regression performs better than the MVG Tied.

surprisingly Gaussianized features does improve the unbalance application $\pi = 0.9$, even if the *Logistic Regression* does not require specific assumption on data distribution.

We will now focus on the Quadratic Logistic Regression. We still have to tune a value for the hyper-parameter $\lambda$:

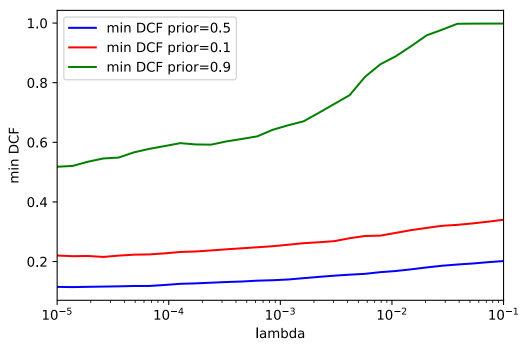


Figure 5: Linear Logistic Regression lambda tuning

Also in this case, the selected value will be $\lambda = 10^{-5}$.

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - no PCA | | | |
| MVG - Tied Covariance | 0.112 | 0.224 | 0.580 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.116 | 0.223 | 0.523 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.114 | 0.217 | 0.547 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.115 | 0.220 | 0.518 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.114 | 0.213 | 0.553 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.9$) | 0.118 | 0.232 | 0.558 |
| Z-Norm features - PCA (m = 7) | | | |
| MVG - Tied Covariance | 0.112 | 0.223 | 0.574 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.116 | 0.220 | 0.532 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.115 | 0.218 | 0.551 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.113 | 0.220 | 0.532 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.1$) | 0.113 | 0.215 | 0.546 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.9$) | 0.117 | 0.231 | 0.548 |
| Gaussianized features - no PCA | | | |
| MVG - Tied Covariance | 0.132 | 0.233 | 0.541 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.124 | 0.232 | 0.517 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.128 | 0.232 | 0.520 |

As we did for the *Linear Logistic Regression*, we try with different $\pi_T$.

We can notice that the *Quadratic Logistic Regression* performs worse than the *Linear Logistic Regression.*

Things change, tho, if the apply PCA with 7 dimensions. In fact, also if the MVG Tied Covariance is still the best, we can see that the *Quadratic Logistic Regression* performs similar to it, gaining better results in the unbalanced applications.

Using different value of $\pi_T$ improves the classification performance for the specific application (at least for $\pi = 0.1$)

Gaussianizing features is not relevant, as we expected.

## 2.3 Support Vector Machines

We now turn our attention to SVMs, where we will consider:

- Linear Support Vector Machine
- Polynomial quadratic kernel
- Radial Basis Function kernel formulations

The formulation of SVM can be expressed as:

$$\max_{\alpha} \alpha^T 1 - \frac{1}{2}\alpha^T \hat{H} \alpha$$

Subject to:

$$0 \leq \alpha_i \leq C_i, i = 1, ..., n$$

where $n$ is the number of training samples, **C** is a hyper-parameter that we need to tune, **1** is a n-dimensional vector of ones, $z_i$ is the class label for the i-th sample encoded as:

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

and **H** is a matrix whose elements are

$$H_{i,j} = z_i z_j x_i^T x_j$$

Since we have an hyper-parameter, we need to tune and choose the best value. We used again a 3-fold cross-validation with:

- Z-Score Normalization
- Z-Score Normalization and PCA with 7 dimensions
- Gaussianized features

We will first analyze the *Linear Support Vector Machine.*

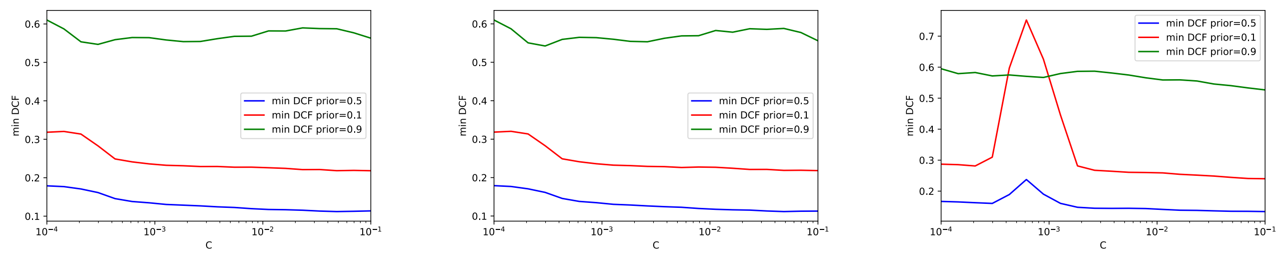


Figure 6: Linear SVM C tuning: Z - Score, Z - Score with PCA 7, Gaussianized features

Given the analysis of the figure above, we choose $C = 0.1$ since it seems to provide a lower minDCF along our main application (blue line). Chosen $C$, we can analyze the results that we obtained with a 3-fold cross-validation.

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - no PCA | | | |
| MVG - Tied Covariance | 0.112 | 0.224 | 0.580 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.116 | 0.223 | 0.523 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.115 | 0.220 | 0.518 |
| Linear SVM (C = 0.1) | 0.114 | 0.218 | 0.563 |
| Z-Norm features - PCA (m = 7) | | | |
| MVG - Tied Covariance | 0.112 | 0.223 | 0.574 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.116 | 0.220 | 0.532 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.113 | 0.220 | 0.532 |
| Linear SVM (C = 0.1) | 0.113 | 0.218 | 0.556 |
| Gaussianized features - no PCA | | | |
| MVG - Tied Covariance | 0.132 | 0.233 | 0.541 |
| Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.124 | 0.232 | 0.517 |
| Quad Log Reg($\lambda$=$10^{-5}$, $\pi_T = 0.5$) | 0.128 | 0.232 | 0.520 |
| Linear SVM (C = 0.1) | 0.134 | 0.240 | 0.527 |

*Linear SVM* produces slightly worse results than MVG. With PCA m=7 it aligns with MVG and Quadratic LogReg.

As we expected, Gaussianization does not improve our model.

We can now focus on the **Non-Linear Support Vector Machines**. The formulation of SVM can be also expressed as:

$$\hat{x} = \begin{bmatrix} X \\ K \end{bmatrix}$$

It is needed then to find a function $k(\hat{x_i}, \hat{x_j}) = \phi(\hat{x_i})^T \phi(\hat{x_j})$ that allows us to compute the dot product. We can thus define:

$$k(\hat{x_i}, \hat{x_j}) = (\hat{x_i}^T \hat{x_j} + c)^d$$

We will employ $d = 2$ for *Quadratic Support Vector Machine* and tune again to find an optimal value for the hyper-parameter $C$.

The technique used to estimate $C$ is still a 3-fold cross-valiation.

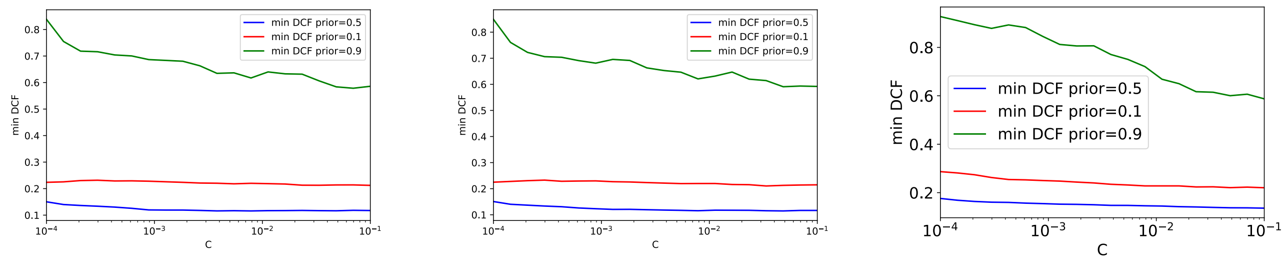


Figure 7: Quadratic SVM C tuning: Z - Score, Z - Score with PCA 7, Gaussianized features

Given the results of the figure above, the choice of C isn't too critical, we can employ $C = 0.1$ .

Choosen $C$, we can again see the results of the calculation:

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - no PCA | | | |
| Linear SVM (C = 0.1) | 0.114 | 0.218 | 0.563 |
| Quadratic SVM(C=0.1) | 0.117 | 0.212 | 0.586 |
| Z-Norm features - PCA (m = 7) | | | |
| Linear SVM (C = 0.1) | 0.113 | 0.218 | 0.556 |
| Quadratic SVM(C=0.1) | 0.117 | 0.215 | 0.592 |
| Gaussianized features - no PCA | | | |
| Linear SVM (C = 0.1) | 0.134 | 0.240 | 0.527 |
| Quadratic SVM(C=0.1) | 0.135 | 0.220 | 0.603 |

We can notice that, confronted to the *Linear SVM*, the **Quadratic SVM** performs slightly worse, in fact its minDCF is 0.117 in both application with and without PCA with 7 dimensions. As seen before, the *Linear SVM* slightly improve with the applicaiton of the PCA.

When we try to Gaussianize features, instead, we obtain a worse result, even if *Linear SVM* and *Quadratic SVM* are almost the same: in fact the *Linear SVM* obtains a minDCF of 0.134, on the other hand the *Quadratic SVM* obtains a minDCF of 0.135.

We can also use a *different* kernel function:

$$k(\hat{x_i}, \hat{x_j}) = e^{-\gamma||\hat{x_i} - \hat{x_j}||}$$

That is also called **Radial Basis Function Support Vector Machine**.

We tune again for the best value of the hyper-parameters $C$ and $\gamma$.

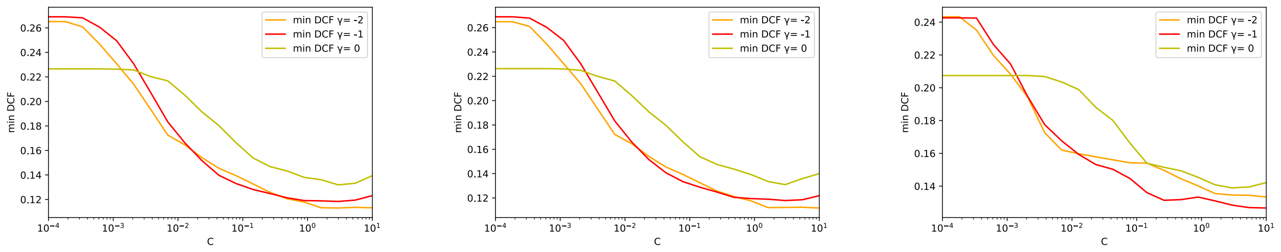


Figure 8: RBF SVM C tuning

As Figure 8 shows, good values for the model could be $C = 10$ and $\gamma = 0.01$. For Gaussianize features we choose $\gamma = 0.1$ instead.

At this point we can again see the results for the **Radial Basis Function**.

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - no PCA | | | |
| Linear SVM (C = 0.1) | 0.114 | 0.218 | 0.563 |
| Quadratic SVM(C=0.1) | 0.117 | 0.212 | 0.586 |
| RBF SVM(C=10, $\gamma$=0.01) | 0.113 | 0.210 | 0.553 |
| Z-Norm features - PCA (m = 7) | | | |
| Linear SVM (C = 0.1) | 0.113 | 0.218 | 0.556 |
| Quadratic SVM(C=0.1) | 0.117 | 0.215 | 0.592 |
| RBF SVM(C=10, $\gamma$=0.01) | 0.112 | 0.209 | 0.564 |
| Gaussianized features - no PCA | | | |
| Linear SVM (C = 0.1) | 0.134 | 0.240 | 0.527 |
| Quadratic SVM(C=0.1) | 0.135 | 0.220 | 0.603 |
| RBF SVM(C=10, $\gamma$=0.1) | 0.127 | 0.215 | 0.798 |

It is possible to see that the **RBF**, compared to the *Linear* and *Quadratic Support Vector Machine* slighlty improve in both the application, with and without applying the PCA with a reduction to 7 dimensions.

Gaussianizing features, we can notice that there are worse performances but the **RBF** would still be the better to perform.

The **Radial Basis Function Support Vector Machine** seems to be the best model among the *SVMs*, anyway we can try to rebalance the classes, using the dual formulation subject to:

$$0 \leq \alpha_i \leq C_i,\ i = 1, ..., n$$

Where $C_i = C_T$ for samples of class 1 and $C_i = C_F$ for samples of class 0. We select $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$.

For $\pi_T = 0.5$ we obtain:

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features - PCA (m = 7) | | | |
| Linear SVM (C = 0.1) | 0.113 | 0.218 | 0.556 |
| Quadratic SVM(C=0.1) | 0.117 | 0.215 | 0.592 |
| RBF SVM(C=10, $\gamma$=0.01) | 0.112 | 0.209 | 0.564 |
| RBF SVM(C=10, $\gamma$=0.01, $\pi_T$=0.5) | 0.110 | 0.220 | 0.528 |

RBF with $\pi_T = 0.5$ obtains slightly better results than the previous best models (MVG and Quadratic LogReg).

## 2.4 Gaussian Mixture Models

The last model considered is a Gaussian Mixture Model (GMM) over the data of each class. The hyper-parameter to tune, in this case, is the number of Gaussian components. It will be tuned both for *Z-Score Normalized* and *Gaussianized* features.

In this report we will consider:

- Diagonal covariance model
- Full covariance model
- Tied covariance model
- Tied Diagonal covariance model

We will use again the 3-folds approach to choose the number of components and to compare different models.

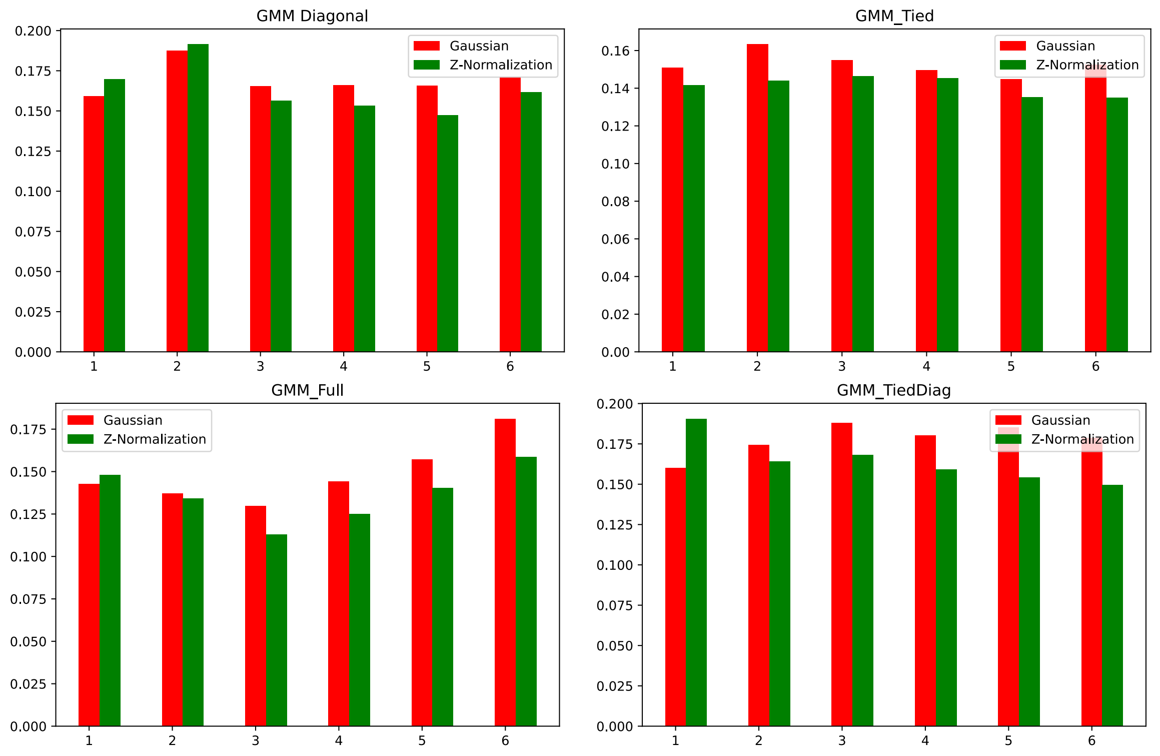


Figure 9: Number of components for GMM models. $x$ is the number of slits. $\pi = 0.5$

The GMM model that performs better is the Full Covariance, in fact is possible to see that with a number of 3 splits, so $2^3 = 8$ components, the minDCF is the lowest, reaching a value of 0.113.

| | 3-fold | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z-Norm features | | | |
| GMM - Full Covariance (8 components, Z Score Normalization) | 0.113 | 0.235 | 0.559 |
| GMM - Tied Covariance (32 components, Z Score Normalization) | 0.135 | 0.244 | 0.640 |
| GMM - Diagonal Covariance (32 components, Z Score Normalization) | 0.147 | 0.266 | 0.645 |
| GMM - Tied Diagonal Covariance (64 components, Z Score Normalization) | 0.149 | 0.298 | 0.628 |

In the case of the GMM model the Tied is different because the Tied MVG assumes that the covariance matrix is shared between class; yet in the case of a GMM the covariance matrix is shared between *components* of the same class.

Based on Figure 9, we selected the best four models:

- GMM - Full Covariance with 8 components and Z Score Normalization
- GMM - Tied Covariance with 32 components and Z Score Normalization
- GMM - Diagonal Covariance with 32 components and Z Score Normalization
- GMM - Tied Diagonal Covariance with 64 components and Z Score Normalization

Analyzing these results, it is possible to notice that the best model appears to be the Full Covariance GMM with 8 components and *Z-Score Normalized* features, without applying PCA.

Now we consider the three model that obtains similar results:

- MVG Tied PCA (m=7)
- Quadratic LogReg($= 10^{-5}$, $\pi_T = 0.5$) PCA (m=7)
- RBF SVM($C = 10$, $\gamma = 0.01$, $\pi_T = 0.5$) PCA (m=7)

for this application we will select two candidate: RBF SVM($C = 10$, $\gamma = 0.01$, $\pi_T = 0.5$) PCA (m=7) as a first system, and MVG Tied PCA (m=7) as a secondary system. Quadratic LogReg could be a substitute for the MVG because we have seen it gives better results on unbalanced application but we will not consider it.

## 2.5 Score Calibration

We have discussed about minDCF metric and costs for the *validation set* using scores of the recognizer. However, the cost we actually pay, depends on a threshold we uso to perform class assignment, reason why we introduce the **actualDCF**.

We decided to follow an approach based on re-calibrating the scores, transforming them so that the theoreical threshol

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

provides close to optimal values over a wide range of effective priors $\tilde{\pi}$.

In this approach we need a transformation function $f$ that maps the classifiers scores into well-calibrated scores $s_{cal} = f(s)$.

There are more than one way to estimate $f$, such as:

- Isotonic regression
- Discriminative score models (such as Logistic Regression)
- Generative score models

We will focus on the second approach, in fact our function $f$ will have a simple form such as:

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

$f(s)$ can be thus interpreted as the log-likelihood ratio for two class hypotheses:

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

The class posterior for $\tilde{\pi}$ corresponds then to:

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

Interpreting scores as features, we have a quiet similar expression as the log posterior ratio of the logistic regression model.

In fact, if we let

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

we have the exact same model. This, can be employed as a model parameter over our training scores. To recover the calibrated score $f(s)$ we will need then:

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

For the **SVM RBF** with calibrated scores, the results are:

| | minDCF ($\pi = 0.5$) | minDCF $\pi = 0.1$) | minDCF ($\pi = 0.9$) |
|---|---|---|---|
| | 0.110 | 0.220 | 0.528 |
| | actDCF ($\pi = 0.5$) | actDCF ($\pi = 0.1$) | actDCF ($\pi = 0.9$) |
| Log Reg ($\pi_T$ =0.5) | 0.112 | 0.225 | 0.558 |

Meanwhile for the **MVG Tied**, the results with calibrated scores are:

| | minDCF ($\pi = 0.5$) | minDCF ($\pi = 0.1$) | minDCF ($\pi = 0.9$) |
|---|---|---|---|
| | 0.112 | 0.223 | 0.574 |
| | actDCF ($\pi = 0.5$) | actDCF ($\pi = 0.1$) | actDCF ($\pi = 0.9$) |
| Log Reg ($\pi_T$ =0.5) | 0.113 | 0.231 | 0.604 |

We selected these two algorithms because their results showed the best performances. Both *minDCF* are with PCA with 7 dimensions applied.

We can see from the tables above that the **actualDCF** is slightly higher than the minDCF, due to the calibration of the scores.

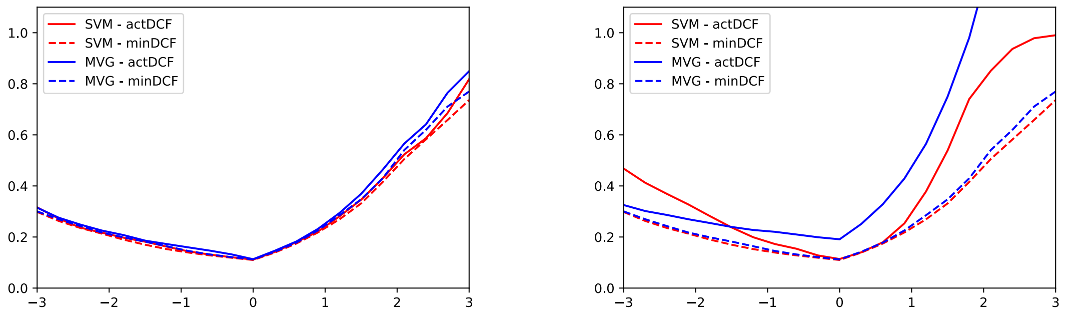


Figure 10: Calibrated (left) and non calibrated scores (right) of SVM and MVG

We can observe that the Logistic Regression approach obtains accurate results. It is also possible to see from the Bayes error plots for the calibrated and non-calibrated scores, the model is trained with $\tilde{\pi} = 0.5$.

### 2.5.1 Fusion

Two classifiers can agree and disagree on some decision, we can try to combine the decisions at **score-level**, fusing the output of the two classifiers to train a meta-model over the scores.

The technique that we use is the stacking ensemble: we take the output of the two classifiers (scores), we build a new dataset where each feature is the output of each calssifier and then we train a meta-model (in our case a balanced LogReg) to produce new score balanced with the model parameters.

We perform a fusion of the **SVM** and **MVG**, hopefully the new scores will have higher accuracy and benefit from the re-calibrating effect of the model.

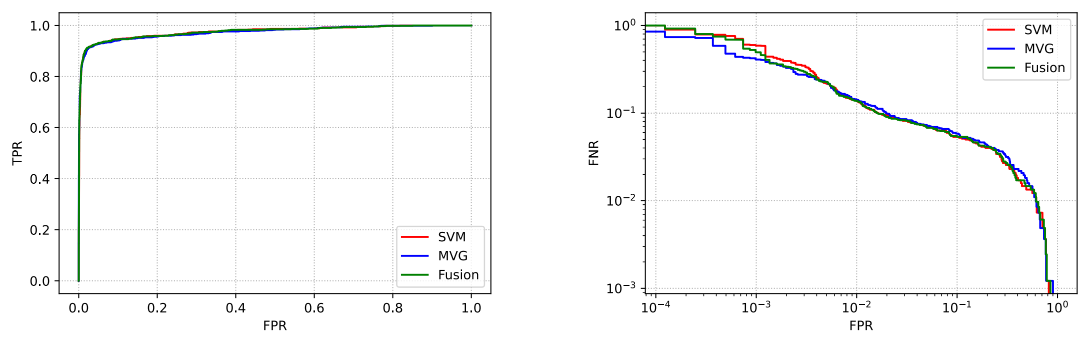


Figure 11: ROC and DET plots

| | $\pi = 0.5$ | | $\pi = 0.1$ | | $\pi = 0.9$ | |
|---|---|---|---|---|---|---|
| | minDCF | actDCF | minDCF | actDCF | minDCF | actDCF |
| SVM | 0.110 | 0.112 | 0.220 | 0.225 | 0.528 | 0.558 |
| MVG | 0.112 | 0.113 | 0.223 | 0.231 | 0.574 | 0.604 |
| Fusion | 0.109 | 0.109 | 0.218 | 0.220 | 0.540 | 0.560 |

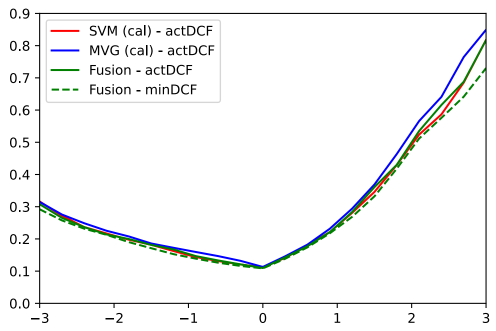


Figure 12: Bayes Error plot, fused system

Fusion does not really improve the scores, it still reduce very slightly the costs but it's aligned with the original models.

Scores are well-calibrated.

We expect the fusion to obtain results close to the RBF SVM or lower.

## 3 Experimental results

We now focus on analyzing the different models performances on the test set, in terms of minDCFs, to see how our choices affected performance for unseen data.

**MVG models:**

| | All data | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z - Norm features | | | |
| MVG - Full Cov | 0.140 | 0.283 | 0.646 |
| MVG - Tied Cov | 0.110 | 0.207 | 0.591 |
| MVG - Diag Cov | 0.185 | 0.330 | 0.621 |
| Z-Norm features - PCA (m = 7) | | | |
| MVG - Full Cov | 0.139 | 0.293 | 0.574 |
| MVG - Tied Cov | 0.110 | 0.208 | 0.587 |
| MVG - Diag Cov | 0.201 | 0.514 | 0.755 |
| Z-Norm features - PCA (m = 6) | | | |
| MVG - Full Cov | 0.139 | 0.293 | 0.574 |
| MVG - Tied Cov | 0.110 | 0.208 | 0.587 |
| MVG - Diag Cov | 0.201 | 0.514 | 0.755 |
| Gaussianized features | | | |
| MVG - Full Cov | 0.151 | 0.241 | 0.622 |
| MVG - Tied Cov | 0.125 | 0.221 | 0.540 |
| MVG - Diag Cov | 0.154 | 0.285 | 0.574 |

**LR models:**

| | All data | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z - Norm features | | | |
| LR ($\lambda$=1e-5, $\pi_T$ =0.5) | 0.109 | 0.199 | 0.537 |
| LR ($\lambda$=1e-5, $\pi_T$ =0.1) | 0.109 | 0.197 | 0.529 |
| LR ($\lambda$=1e-5, $\pi_T$ =0.9) | 0.115 | 0.205 | 0.508 |
| Z-Norm features - PCA (m = 7) | | | |
| LR ($\lambda$=1e-5, $\pi_T$ =0.5) | 0.108 | 0.203 | 0.538 |
| LR ($\lambda$=1e-5, $\pi_T$ =0.1) | 0.109 | 0.199 | 0.532 |
| LR ($\lambda$=1e-5, $\pi_T$ =0.9) | 0.115 | 0.203 | 0.532 |
| Gaussianized features | | | |
| LR ($\lambda$=1e-5, $\pi_T$ =0.5) | 0.121 | 0.216 | 0.485 |
| LR ($\lambda$=1e-5, $\pi_T$ =0.1) | 0.122 | 0.209 | 0.499 |
| LR ($\lambda$=1e-5, $\pi_T$ =0.9) | 0.121 | 0.217 | 0.475 |

**Quadratic LR:**

| | All data | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z - Norm features | | | |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.5$) | 0.103 | 0.203 | 0.470 |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.1$) | 0.105 | 0.202 | 0.519 |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.9$) | 0.108 | 0.207 | 0.451 |
| Z-Norm features - PCA (m = 7) | | | |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.5$) | 0.104 | 0.202 | 0.470 |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.1$) | 0.104 | 0.202 | 0.515 |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.9$) | 0.108 | 0.208 | 0.467 |
| Gaussianized features | | | |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.5$) | 0.116 | 0.215 | 0.479 |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.1$) | 0.119 | 0.209 | 0.507 |
| Quad LR ($\lambda$=1e-5, $\pi_T = 0.9$) | 0.116 | 0.223 | 0.483 |

**Support Vector Machines:**

| | All data | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z - Norm features | | | |
| SVM C=0.1 | 0.112 | 0.205 | 0.552 |
| QSVM C=0.1 | 0.102 | 0.204 | 0.522 |
| RBF C=10, $\gamma$=0.01 | 0.106 | 0.200 | 0.465 |
| Z-Norm features - PCA (m = 7) | | | |
| SVM C=0.1 | 0.112 | 0.205 | 0.552 |
| QSVM C=0.1 | 0.104 | 0.204 | 0.532 |
| RBF C=10, $\gamma$=0.01 | 0.106 | 0.201 | 0.483 |
| RBF C=10, $\gamma$=0.01, $\pi_T = 0.5$ | 0.098 | 0.200 | 0.480 |
| Gaussianized features | | | |
| SVM C=0.1 | 0.129 | 0.222 | 0.499 |
| QSVM C=0.1 | 0.125 | 0.209 | 0.541 |
| RBF C=10, $\gamma$=0.1 | 0.114 | 0.209 | 0.687 |

**GMM models:**

| | All data | | |
|---|---|---|---|
| | $\pi = 0.5$ | $\pi = 0.1$ | $\pi = 0.9$ |
| Z - Norm features | | | |
| GMM Full-Cov 8c | 0.100 | 0.220 | 0.578 |
| GMM Tied 32c | 0.128 | 0.241 | 0.607 |
| GMM Diag 32c | 0.142 | 0.267 | 0.616 |
| GMM Tied-Diag 64c | 0.145 | 0.282 | 0.683 |

We can see that the results are consistent with the one showed during the *validation*, with a few exception: in fact the LogReg models here have better performance than the MVG Tied model (candidate system that we selected).

GMM Full-Cov 8c it's the second best model for this experiment.

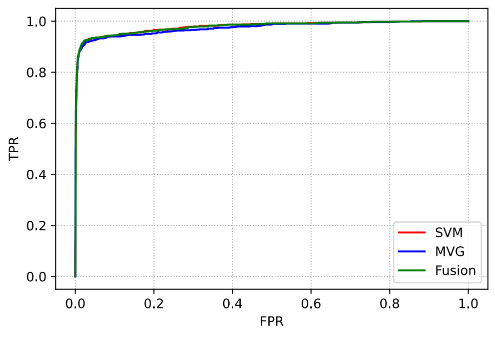
Overall PCA (m=7) in some cases is effective but in others it worsen the results (Quad SVM).

The first system that we selected is still the best one.

Finally, we can evaluate the effectiveness of our model fusion.

**Fusion:**

| | $\pi = 0.5$ | | $\pi = 0.1$ | | $\pi = 0.9$ | |
|---|---|---|---|---|---|---|
| | minDCF | actDCF | minDCF | actDCF | minDCF | actDCF |
| SVM - Calibrated | 0.098 | 0.099 | 0.200 | 0.210 | 0.480 | 0.499 |
| MVG - Calibrated | 0.110 | 0.114 | 0.208 | 0.220 | 0.587 | 0.606 |
| Fusion | 0.098 | 0.099 | 0.199 | 0.206 | 0.494 | 0.534 |



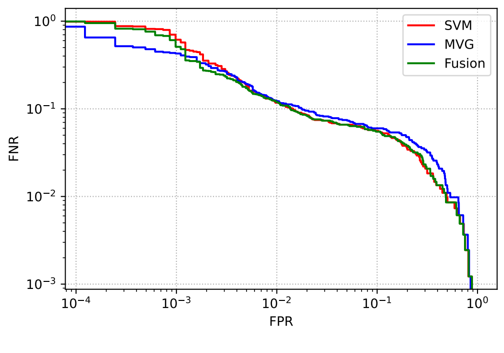


Figure 13: ROC and DET plots on evaluation set

As we expected, the fusion aligns with the SVM model, giving similar results.

## 4 Conclusion

Concluding, our approach, consisting of the fusion of two subsystems, it's effective: we could've achieved better results fusing different system, but the one we analyzed produce the best results among all the models we made and produce well calibrated scores.

We are able to achieve a DCF cost of $\approx 0.1$ for our main application ($\pi = 0.5$). The model is good also for applications with imbalanced prior 0.1 (DCF cost of $\approx 0.2$) and for the one with prior 0.9. (DCF cost of $\approx$ 0.5).

The choices we made on our training / validation sets proved effective also for the evaluation data

## References

[1] R. J. Lyon, B. W. Stappers, S. Cooper, J. M. Brooke, J. D. Knowles, Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach, Monthly Notices of the Royal Astronomical Society 459 (1), 1104-1123, DOI: 10.1093/mnras/stw656

[2] R. J. Lyon, HTRU2, DOI: 10.6084/m9.figshare.3080389.v1.