Pulsar Detection - Binary Classification

Omar Ormachea s304811

Beatrice Piras s304812

Abstract

The objective of this work is to find the best-fit model for classifying the HTRU2 dataset. The analysis starts with the study of its features and their distributions, then moving into some common machine learning algorithms and studying their results when applied onto the dataset. After the selection of best candidate models, further techniques such as score calibration and score-level fusion were applied.

Contents

1 Introduction

The object of interest for this analysis are pulsars, which are celestial sources of pulsating electromagnetic radiations (similar to radio waves) classified as a type of Neutron Star. The data set adopted in this project is called HTRU2 and it contains the description of a set of pulsar candidates. It was collected during the High Time Resolution Universe Survey (South)[1]. This project's main objective is to automatically label pulsar candidates, through the use of Machine Learning tools that facilitate the classification and analysis of the data. Since this problem is a binary classification problem, 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 samples, that have been split into a training set with 8929 samples and a test set, also with 8969 samples.

1.1 Attribute description:

Each candidate is described by 8 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 continuous variables 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 last four variables are similarly obtained from the DM-SNR curve.

2 Data preparation and observations

2.1 Z-normalization

The numerical values are quite high, so in order to avoid overflow, the dataset has been pre-processed applying Z-normalization:

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

where x_i = value of the i-th sample, μ = mean vector and σ = standard deviation of the samples. This way, no information is lost and the computations will potentially be carried out more efficiently. The feature histograms show no relevant outliers. The False class' feature distributions show a gaussian profile. Instead, the True class shows more distinct distributions among features, but there are still some cases of seemingly gaussian distributions. For this reason, we assume that a transformation such as Gaussianization wouldn't benefit much. Z-normalization may be sufficient for our analysis.

The Z-normalized features appear to be normally distributed

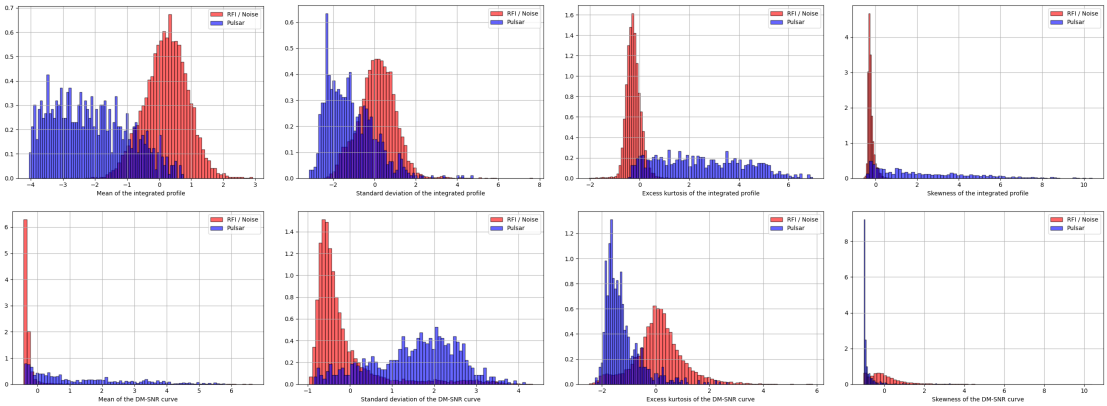

Fig. 1: Z-Normalized feature histogram

2.2 Pearson Correlation

To illustrate the correlation between the different features, the absolute value of the Pearson Correlation coefficient

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

has been used to represent the following heatmaps:

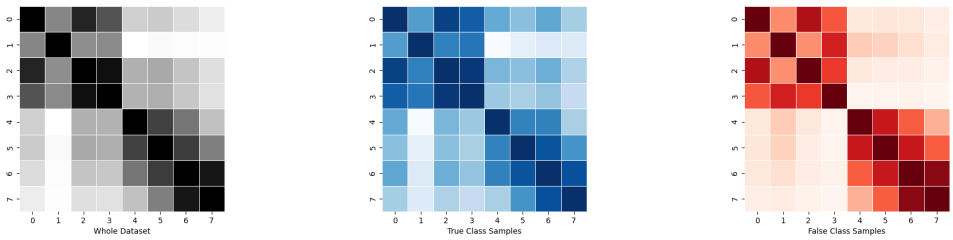

Fig. 2: Feature correlation heatmaps. *Left*: Whole dataset. *Middle*: True class samples. *Right*: False class samples

It can be noticed that the features [0..3] and the features [4..7] are quite uncorrelated, while features 2-0 and features 6-7 are strongly correlated. With this observation, we can deduce that a dimensionality reduction to 7 or even 6 dimensions made with PCA would be convenient to improve some classification results without the loss of meaningful information. A correlation between features 2 and 3 is also noticeable, strongly correlated for the True class and weakly correlated for the False class. We decided to attempt using a further reduction to 5 dimensions with PCA and see what the outcome is.

3 Feature Classification

The various classifiers will be tested using both a single-fold and a K-fold cross-validation with 5 folds over 3 application points:

- (main) balanced application point with : $\tilde{\pi} = 0.5$
- Two unbalanced application points with: $\tilde{\pi} = 0.1$ and $\tilde{\pi}= 0.9$

Furthermore, the classifiers will be run firstly without applying PCA, and then circumstantially applying PCA with $m = 7, 6, 5$ (m representing the number of dimensions after reducing).

The performance of the classifiers will be measured with the calculation of the normalized *Minimum Detection Cost Function* (**minDCF**), in order to select the most convenient approach in terms of "cost to pay" if optimal decisions were taken using the recognizer scores.

3.1 Multivariate Gaussian Classifiers

The first classifiers trained are Multivariate Gaussian classifiers (MVG). Specifically:

- Full Covariance Gaussian
- Diagonal Covariance Gaussian
- Tied Full Covariance Gaussian
- Tied Diagonal Covariance Gaussian

They are all generative models and they all assume Gaussian distributed data, given the class:

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

MVG Classifiers - minDCF on the validation set

	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
Without PCA						
Full Covariance	0.11	0.281	0.63	0.141	0.284	0.669
Diagonal Covariance	0.183	0.28	0.619	0.193	0.314	0.746
Tied Full Covariance	0.108	0.216	0.522	0.112	0.223	0.574
Tied Diagonal Covariance	0.156	0.257	0.578	0.161	0.265	0.58
With PCA m= 7						
Full Covariance	0.134	0.281	0.62	0.1732	0.298	0.708
Diagonal Covariance	0.178	0.592	0.693	0.219	0.631	0.82
Tied Full Covariance	0.119	0.268	0.566	0.152	0.279	0.61
Tied Diagonal Covariance	0.136	0.279	0.606	0.172	0.303	0.624
With PCA m = 6						
Full Covariance	0.142	0.292	0.562	0.17	0.315	0.657
Diagonal Covariance	0.179	0.604	0.677	0.219	0.645	0.806
Tied Full Covariance	0.131	0.263	0.585	0.156	0.285	0.596
Tied Diagonal Covariance	0.137	0.28	0.578	0.172	0.31	0.623
With PCA m = 5						
Full Covariance	0.151	0.386	0.624	0.18	0.428	0.731
Diagonal Covariance	0.172	0.566	0.711	0.201	0.604	0.784
Tied Full Covariance	0.154	0.279	0.586	0.178	0.312	0.601
Tied Diagonal Covariance	0.163	0.291	0.585	0.187	0.343	0.61

Analyzing the data from the table above we can observe how the *5-fold* generally seems to perform worse than the *single fold*, but the results for the 5-folds are ultimately more **realistic**, since when using *single fold* there is the probability to choose a very well performing set that does not represent the reality of the general set.
For this reason, from now on we will focus on the *5-fold* results.

Comparing the results between classifiers, we notice that the *diagonal covariance* model and the *tied diagonal* model give the worse results. This is because we have strong correlations that need to be taken into account, but the diagonal covariance classifier will not consider them. The correlations are captured in the elements outside of the diagonal of the covariance matrix and for this reason, relevant data will be lost.
We notice, instead, that the *tied full covariance* model gives the best results. This could be due to various reasons:

- The distributions are fairly similar between classes.
- When fewer samples are present (like in this case for the False class), the tied full covariance model is able to construct covariance matrices which, compared to Full covariance model, are more robust to noise.

In regard of the use of **PCA**, as expected, the best performing results are obtained without it. This is because even though we have some features that appear to be strongly correlated when considering the whole dataset, they are not as similarly correlated when taking in consideration the distinct classes (True and False). In particular, even removing just one dimension the results are far worse since each feature contributes to the results greatly, and as said before the correlations are not as strong as we initially thought in out first assumptions.

We also noticed that going from PCA m=7 to PCA m=6 there is not a relevant performance drop. We assume this is because the two pairs of "strongly" correlated features we identified before are similar in their degree of correlation. Instead, with PCA m=5 we noticed an important (but expected) performance drop since the corresponding correlated feature pair holds a strong correlation for the True class only (and it is also the less abundant class). For these reasons, from this point on we will focus our attention on the results without applying PCA and applying a reduction to 7 dimensions (1 dimension reduced).

3.2 Logistic Regression Classifiers

In this section we will focus on discriminative models, in particular we will analyze the results obtained with Linear Logistic Regression and Quadratic Logistic regression.

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

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

Where (w,b) are the model parameters. For both linear and quadratic logistic regression models we set $\pi_T = 0.5$ for selecting the most appropriate value of the hyperparameter λ.

3.2.1 Linear Logistic Regression

As a first step we will tune the hyperparameter λ, to find the best possible value in terms of minDCF.

Linear logistic Regression Lambda Tuning:

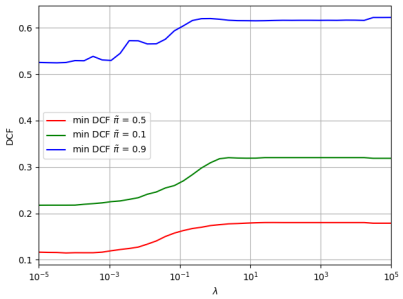

Fig. 3: No PCA

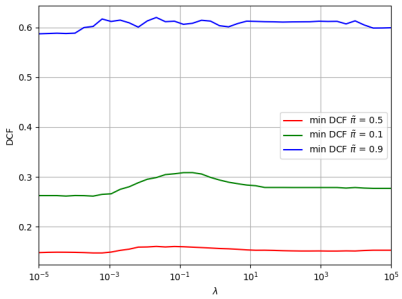

Fig. 4: PCA m=7

From the above figures, we convey that our best candidate is $\lambda = 10^{-5}$. In the table below the results for different prior π_T are shown.

minDCF for Z-Normalized data, 5-Fold			
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
MGV- Tied Full Covariance	**0.112**	0.223	0.574
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.116	0.218	0.526
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	**0.113**	0.21	0.559
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.119	0.219	0.515
PCA m=7			
MGV- Tied Full Covariance	0.152	0.279	0.61
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.148	0.263	0.587
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.144	0.259	0.62
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.156	0.267	0.575

As we can evince from this comparison, Linear logistic regression generally works slightly better compared to our previous best candidate (**Tied Full** covariance MVG model), and the parameter that seems to be working best is $\pi_T = 0.1$. This is consistent with the actual class proportions of our data.

3.2.2 Quadratic Logistic Regression

In some cases, a linear separation rule might not be able to properly classify the data. For this reason, we implemented quadratic logistic regression, which projects the data into a higher-dimensional 2D space (expanded feature space). It performs a linear separation in this space which corresponds to a quadratic separation rule in the original space.

We performed hyperparameter tuning also for this model. The results are the following:

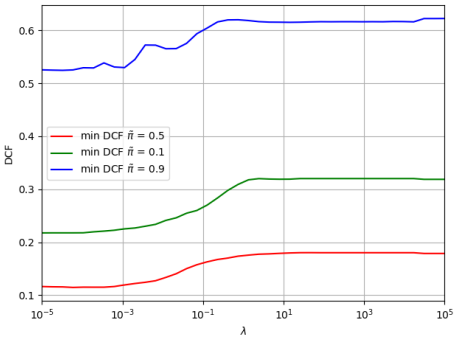

Fig. 5: Quadratic Logistic Regression Lambda Tuning: No PCA

As in the linear case, we select $\lambda = 10^{-5}$.

minDCF for Z-Normalized data, 5-Fold			
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
MVG- Tied Full Covariance	0.112	0.223	0.574
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.113	0.21	0.559
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.114	0.218	0.47
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.11	0.206	0.503
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.123	0.235	0.514
PCA m=7			
MVG- Tied Full Covariance	0.152	0.279	0.61
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.156	0.267	0.575
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.126	0.215	0.562
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.122	0.219	0.55
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.13	0.235	0.582

We can observe that the results are fairly similar to the ones observed for the **Linear LogReg** model, even slightly better with a prior $\pi_T = 0.1$.

Since using a quadratic kernel lead to slightly better results, we could expect similar performances on the upcoming kernel SVM models.

3.3 Support Vector Machines

We will now deploy three different SVM models:

- Linear SVM
- Polynomial quadratic kernel SVM
- Radial Basis Function kernel SVM

The dual formulation of the SVM problem is a maximization problem w.r.t. α:

$$J^D(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{1} \qquad \text{subject to} \quad 0 \leq \alpha_i \leq C, \forall i \in \{1, \ldots, n\}, \sum_{i=1}^{n} \alpha_i z_i = 0$$

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

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

and H is the matrix:

$$\boldsymbol{H}_{ij} = z_i z_j \boldsymbol{x}_i^T \boldsymbol{x}_j$$

Due to the high class imbalance, the SVM implementation considered class balancing. Balancing is done by considering a different value of C for the different classes in the box constraint of the dual formulation:

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

where $C_i = C_T$ for samples in the True class and $C_i = C_F$ for the ones in the False class.
We choose $C_T = C\frac{\pi_T}{\pi_T^{emp}}$ and $C_F = C\frac{\pi_F}{\pi_F^{emp}}$, π_T^{emp} and π_F^{emp} being the empirical priors for the classes computed over the training set.

We employed 5-fold cross validation for choosing the best value of C in the 3 models proposed, without using class rebalancing.

3.3.1 Linear SVM

The first step is the selection of the hyperparameter C:

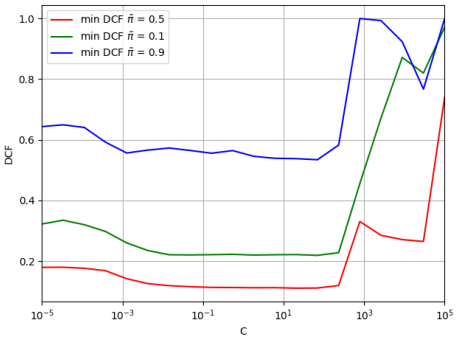

Fig. 6: Linear SVM C Tuning: No PCA

As we can see from the image, the best value that we can select appears to be $C = 10^{-1}$ as it is the lowest in the 3 application points with a stable margin.

minDCF for Z-Normalized data, 5-Fold			
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
MVG- Tied Full Covariance	0.112	0.223	0.574
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.113	0.21	0.559
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	**0.11**	0.206	0.503
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	**0.111**	0.211	0.569
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	0.115	0.221	0.563

From the table above we can evince that the Linear SVM model doesn't perform better than Quadratic LogReg model, but it performs similarly.

3.3.2 Polynomial (quadratic) kernel SVM

Now we will focus on the non-linear SVM models. The non-linearity can be obtained via an explicit expansion of the feature space into a higher dimensional space as in Quadratic Logistic Regression, which is very computationally expensive. A much better option is to exploit the dual formulation of the SVM problem that allows us to work with only the dot product in the expanded space. This is achieved by using the so called *kernel function*:

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

In this first case we will make use of the polynomial kernel function, defined as:

$$k\left(\hat{x}_i, \hat{x}_j\right) = \left(\hat{x}_i^T \hat{x}_j + c\right)^d$$

Also in this case, the tuning of the hyper parameter C is needed. As opposed to the previous case, the C hyper-parameter is dependent on the parameter c and d (degree set to 2). Thus, a joint optimization of C and c is performed:

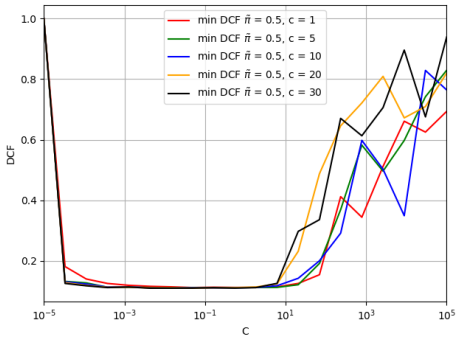

Fig. 7: Quadratic kernel SVM C Tuning: No PCA

As we can see from the image, the best value that we can select appears to be $C = 10^{-1}$ and $c = 1$, since this line is the best performing in general.

minDCF for Z-Normalized data, 5-Fold

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	**0.11**	0.206	0.503
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	0.111	0.211	0.569
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	0.115	0.221	0.563
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	0.113	0.205	0.553
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	**0.11**	0.218	0.498
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.1$)	0.112	0.206	0.551
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.9$)	0.135	0.274	xxx

As we can see from the table above, the improvement over the linear SVM model is small but the results are to the Quadratic LogReg, which suggests that quadratic discriminative methods seem to work well in our model.
It can also be noted that class balancing does not improve the performance significantly for the application points $\tilde{\pi} = 0.5$ and $\tilde{\pi} = 0.1$, while for $\tilde{\pi} = 0.9$ there is a vast improvement, so a class balancing of $\pi_T = 0.5$ seems to be the best option.

3.3.3 Radial Basis Function kernel SVM

In this case, the tuning of the hyper-parameter C is done jointly with the parameter γ:

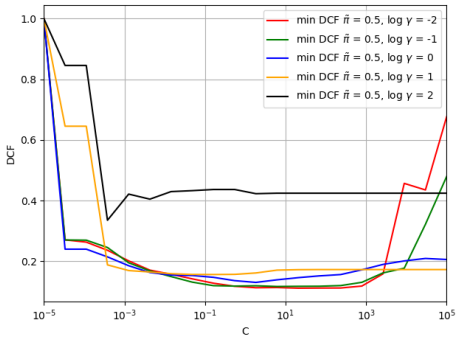

Fig. 8: RBF SVM C Tuning: No PCA

As we can see from the image, the best value that we can select appears to be $C = 10^{-1}$ and the corresponding best value of γ seems to be $\gamma = 10^{-1}$.
Also this time we will try both balanced and unbalanced versions of the RBF kernel.

minDCF for Z-Normalized data, 5-Fold

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	0.111	0.211	0.569
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	0.115	0.221	0.563
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	0.113	0.205	0.553
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	0.11	0.218	0.498
RBF kernel SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	0.125	0.232	0.727
RBF kernel SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	0.123	0.229	0.561

The table suggests that RBF kernel performs significantly worse than the previous models and that class balancing improves performance for the application point $\tilde{\pi} = 0.9$ only.

3.4 Gaussian Mixture Models

At last, we will consider Gaussian Mixture models (GMM), generative models based on the approximation of generic distributions by "fitting" MVG distributions and associating each sample to one of the "clusters" defined by each of the MVG components. These are obtained by iteratively increasing the number of Gaussian components by performing splits (2, 4, 8 components etc...).
Using the 5-fold approach, we will take into consideration:

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

For the Tied models, the covariance tying is done at class level, meaning that different classes (True and False) have different covariance matrices. It is each of the MVG components inside a class that share the same covariance matrix.

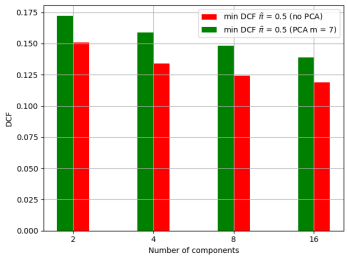

Fig. 9: Full covariance

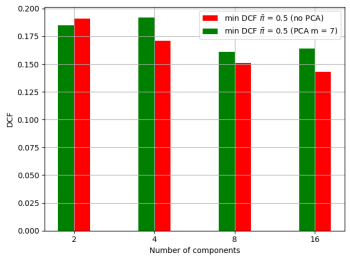

Fig. 10: Diagonal Covariance

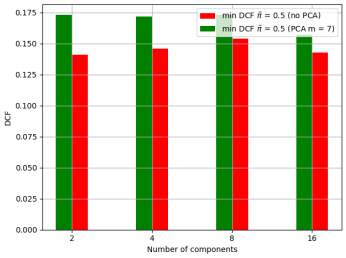

Fig. 11: Tied covariance

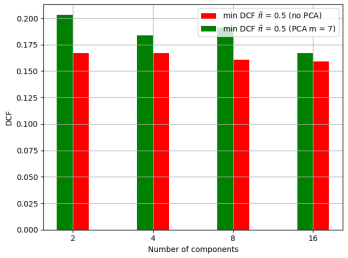

Fig. 12: Tied Diag Covariance

As we can see from the Figures above, not using PCA is consistently better when compared to the PCA m=7 case. We can also notice a trend where, except for the Tied Covariance, an increasing number of splits seems to improve the results of the minDCF.

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
Full Covariance GMM (n_components = 16)	0.119	0.237	0.568
Diag Covariance GMM (n_components = 32)	0.143	0.253	0.581
Tied Covariance GMM (n_components = 2)	0.141	0.283	0.644
Tied Diag Covariance GMM (n_components = 16)	0.159	0.297	0.634

The best results are obtained by the Full Covariance GMM model with 16 components, even though it does not outperform the previous best score registered by the Quadratic SVM model.

The table below summarizes the best combination of parameters for each family of models analyzed so far.

minDCF - best candidates so far

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
MVG - Tied Full Covariance	0.112	0.223	0.574
Linear LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.113	0.21	0.559
Quadratic LogReg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	**0.11**	0.206	0.503
Linear SVM ($K = 1$, $C = 0.1$, $\pi_T =$ unb)	0.111	0.211	0.569
Quadratic SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	**0.11**	0.218	0.498
RBF kernel SVM ($K = 1$, $C = 0.1$, $\pi_T = 0.5$)	0.123	0.229	0.561
Full Covariance GMM (n_components = 16)	0.119	0.237	0.568

3.5 Scores calibration

We have discussed about **minDCF** metric and costs for the validation set using scores of the recognizer. The **minDCF** is an optimistic metric that assumes the selection of the optimal threshold for performing class assignment. However, the cost that we actually pay depends on the actual threshold used for performing class assignment, which is the theoretical threshold:

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

Due to the lack of probabilistic interpretation for the recognizer scores, the theoretical threshold usually does not correspond to the optimal one. For this reason, we introduce the **actualDCF** metric, allowing us to measure the preciseness of the actual decisions that will be made by the recognizer. We decided to follow an approach to re-calibrate the scores, transforming them so that the theoretical threshold provides close to optimal values over a wide range of effective priors $\tilde{\pi}$.
For this approach, we need a transformation function f that maps the classifiers scores into well-calibrated scores $s_{cal} = f(s)$. There are several 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 be an affine function:

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

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

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

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

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

Interpreting scores as features, we have a quite 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:

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

We selected as our candidate models the Tied Full covariance MVG and the Quadratic Kernel SVM models with class balancing $\tilde{\pi} = 0.5$, since these provided good results in terms of $minDCF$ and we were interested in combining a generative and a discriminative classifier.

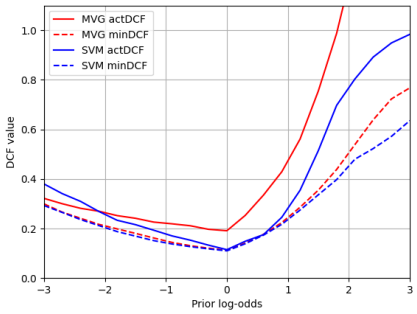

Fig. 13: Bayes error plot for uncalibrated MVG and quadratic SVM scores

minDCF and actualDCF for uncalibrated scores

	$\tilde{\pi}$=0.5		$\tilde{\pi}$=0.1		$\tilde{\pi}$=0.9	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
MVG - Tied Full Covariance	0.112	0.191	0.223	0.273	0.574	1.422
Quadratic SVM (K=1, C =0.1, π_T=0.5)	0.11	0.114	0.218	0.286	0.498	0.837

In the figures above it can be observed that for the main application point $\tilde{\pi}$=0.5, the scores already have an acceptable level of calibration. Instead for the remaining range of application points the miscalibration is very negatively impactful.
The score calibration was done taking the 5-fold scores of the full training set and training a pseudo Linear Logistic Regression model with a 1D dataset (consisting on the scores only), then applying the transformation pair *(w, b)* onto itself. These two transformations arrays were stored for later applying them onto the test set in the final part of the analysis. After applying the score calibration to the training scores we obtained the following improved results:

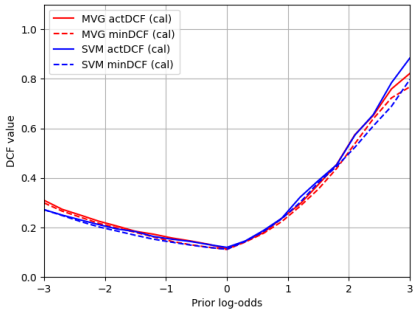

Fig. 14: Bayes error plot for **calibrated** MVG and quadratic SVM scores

As we can see the actualDCF now provides closer to optimal values. An important detail to note is that the Logistic Regression for calibration was trained with a $\pi_T = 0.5$ (representing the main application point $\tilde{\pi}$=0.5) and it is still able to provide well-calibrated scores over a wide range of applications.

minDCF and actDCF for **calibrated scores**

	$\tilde{\pi}$=0.5		$\tilde{\pi}$=0.1		$\tilde{\pi}$=0.9	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
MVG - Tied Full Covariance	0.112	0.113	0.223	0.228	0.574	0.616
Quadratic SVM (K=1, C =0.1, π_T=0.5)	0.11	0.114	0.218	0.286	0.498	0.837

3.6 Fusion Model

Following a similar approach, we are able to perform a *score level* fusion of two models. We assume that the fused score is a function of the outputs of the different classifiers. If $s_{t,A}$ and $s_{t,B}$ are the scores of classifiers A and B for sample x_t, then the fused score for sample x_t will be

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

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

$$f\left(s_{t,A}, s_{t,B}, s_{t,C}, \ldots\right) = \alpha_A s_{t,A} + \alpha_B s_{t,B} + \alpha_C s_{t,C} + \cdots + \beta$$

We will train a new model taking as the "dataset" the two sets of scores stacked vertically, as if the dataset contained 2-dimensional samples. After the training, the new scores should hopefully be more accurate and benefit from the re-calibrating effect of this method.
Our candidate models *MVG - Tied Full Covariance* and *Quadratic SVM (K=1, C =0.1, π_T=0.5)* were combined following this approach. Here are the results:

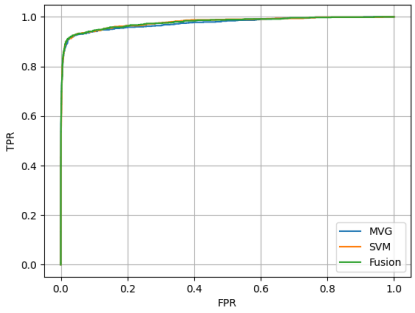

Fig. 15: ROC curve

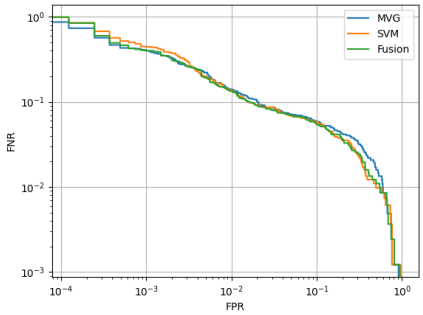

Fig. 16: DET curve

The figures above show that the fusion model performs similarly in regards of the error trade-offs through all the tested thresholds.

Fusion [MVG Tied Full Cov, Quad SVM (C = 0.1, π_T=0.5)] on training set

	minDCF		
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
	No PCA		
Fusion	0.108	0.216	0.499
MVG Tied Full Cov	0.112	0.223	0.574
Quad SVM (C = 0.1, π_T = 0.5)	0.11	0.218	0.498
	PCA m= 7		
Fusion	0.13	0.248	0.565
MVG Tied Full Cov	0.152	0.279	0.61

In terms of minDCF, though, there is a clear improvement over the candidate models, especially for the $\tilde{\pi} = 0.9$ application point. The transformation pair, as in the calibration model, was stored for later use in the evaluation phase.

4 Experimental Results

In this last section we will repeat the analyses on the evaluation set to study all the models previously taken into consideration, in order to confirm or discredit the results observed while applying the 5-fold cross-validation. We will take into consideration the most relevant results in terms of minDCF as well as the hyperparameter tunings.

4.1 MVG classifiers

MVG Classifiers - minDCF on test set			
	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
Without PCA			
Diagonal Covariance	0.185	0.33	0.621
Tied Covariance	0.11	0.207	0.591
With PCA m= 7			
Diagonal Covariance	0.217	0.626	0.678
Tied Covariance	0.147	0.268	0.622
With PCA m = 6			
Diagonal Covariance	0.215	0.648	0.687
Tied Covariance	0.147	0.268	0.607

During the training phase, the best and worst variants were the Tied Full and Diagonal covariance models, respectively. As it can be seen in the table above, the results prove to be consistent for all the application points. Also, as observed before, the use of PCA (m=7) is not effective as it decreases significantly the performance. From m=7 to m=6, again, there is a negligible decrease in performance. The 5-fold cross-validation was effective.

4.2 Logistic Regression

4.2.1 Linear Logistic Regression

As for Linear Logistic Regression, the same trends can be observed.

We proceed with the tuning of the hyper-parameter λ:

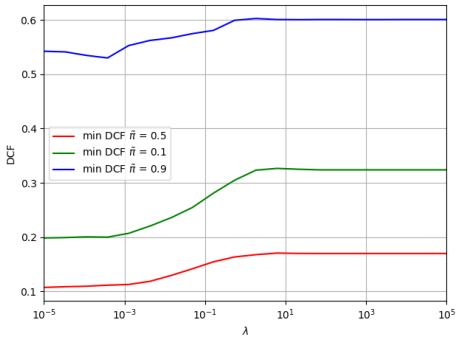

Fig. 17: Linear LogReg λ tuning

For bigger values of the hyper-parameter λ the performance is decreased. The choice of $\lambda = 10^{-5}$ was appropriate.

minDCF - Linear Log Reg on test set

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
Without PCA			
Linear Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.107	0.199	0.542
Linear Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.11	0.199	0.531
Linear Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.115	0.201	0.51
With PCA m= 7			
Linear Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.14	0.245	0.605
Linear Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.14	0.246	0.605
Linear Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.145	0.251	0.582

Class balancing seems to not have meaningful effects, as the minDCF values are almost equal. Nevertheless, the "best" model is obtained by applying class-balancing with $\pi_T = 0.1$, as seen with cross-validation. This model, as expected, slightly outperforms the candidate MVG model.

4.2.2 Quadratic Logistic Regression

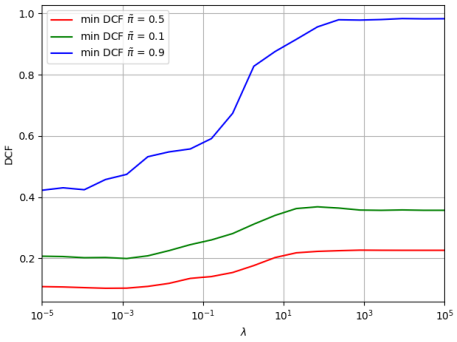

Fig. 18: Quadratic LogReg λ tuning

Now, considering Quadratic Logistic Regression, the choice of λ again seems to be accurate.

minDCF - Quadratic Log Reg on test set

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
Without PCA			
Quadratic Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.108	0.207	0.422
Quadratic Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.1	0.206	0.457
Quadratic Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.115	0.215	0.441
With PCA m= 7			
Quadratic Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.5$)	0.112	0.208	0.481
Quadratic Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.1$)	0.115	0.203	0.492
Quadratic Log Reg ($\lambda = 10^{-5}$, $\pi_T = 0.9$)	0.114	0.224	0.473

As in Linear LogReg, class balancing doesn't affect much the minDCF. The unbalanced applications $\pi_T = 0.5$ and $\pi_T = 0.1$ perform better than the $\pi_T = 0.9$ case. Also in this case, the 5-fold cross validation seems to be accurate enough with respect to the evaluation set result.

4.3 Support vector machines

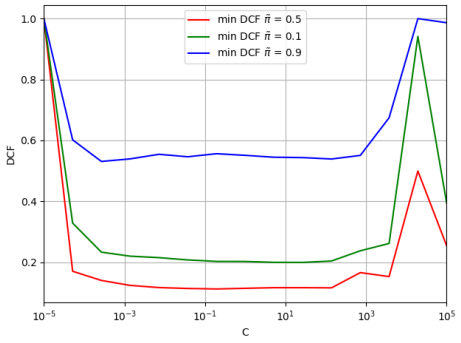

Fig. 19: Linear SVM C tuning

Considering the Linear SVM models, we again tested multiple values of the coefficient C in order to find the best performing one. The selected value on the validation phase is consistent with the evaluation data ($C = 0.1$).

minDCF - Support Vector Machines on test set

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
Without PCA			
Linear SVM (K = 1, C = 0.1, π_T = unb)	0.112	0.205	0.552
Linear SVM (K = 1, C = 0.1, π_T = 0.5)	0.11	0.203	0.567
Quadratic Kernel SVM (K = 1, C = 0.1, π_T = unb)	0.102	0.204	0.552
Quadratic Kernel SVM (K = 1, C = 0.1, π_T = 0.5)	0.098	0.205	0.451
RBF Kernel SVM (K = 1, C = 0.1, π_T = unb)	0.119	0.222	0.648
RBF Kernel SVM (K = 1, C = 0.1, π_T = 0.5)	0.115	0.223	0.498
With PCA m= 7			
Linear SVM (K = 1, C = 0.1, π_T = unb)	0.141	0.247	0.614
Linear SVM (K = 1, C = 0.1, π_T = 0.5)	0.136	0.248	0.617
Quadratic Kernel SVM (K = 1, C = 0.1, π_T = unb)	0.123	0.215	0.559
Quadratic Kernel SVM (K = 1, C = 0.1, π_T = 0.5)	0.112	0.218	0.477

Class balancing doesn't affect much, as expected.

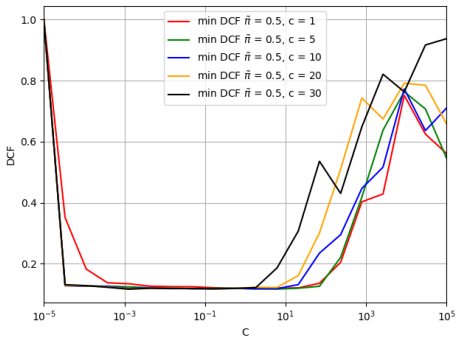

Fig. 20: Quadratic SVM C tuning

As for the Quadratic Kernel SVM model, a joint optimization was made for the C coefficient and the c parameter. The value pair ($C = 0.1$, $c = 1$) was, again, the most robust and better performing. In terms of class balancing, the best choice is $\pi_T = 0.5$, since it performs consistently better, mostly

in the application point $\tilde{\pi} = 0.9$. This confirms the selection of this model as the best one during the validation phase.

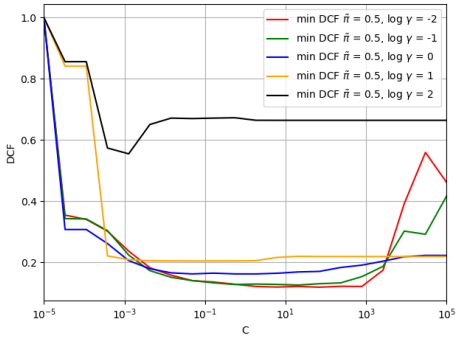

Fig. 21: RBFc SVM C tuning

RBF kernel model performs worse than the previously analyzed models, as it did in the validation phase. Several values for the γ parameter were tested with cross-validation and the reported results are consistent with the trends observed during the validation phase. The best pair of values is again ($C = 0.1$, $\gamma = 0.1$)

4.4 Gaussian Mixture Models

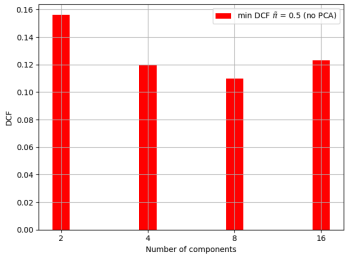

Fig. 22: Full Cov GMM

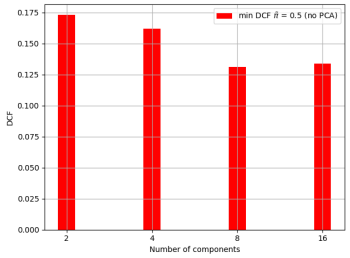

Fig. 23: Diagonal Cov GMM

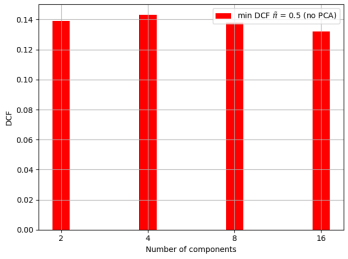

Fig. 24: Tied Full Cov GMM

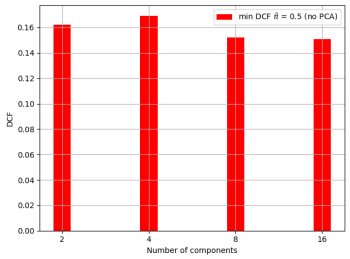

Fig. 25: Tied Diagonal Cov GMM

Turning our attention to the Gaussian Mixture Models, we again observed a worse performance than the previous models, as seem on the table and the plots. Interestingly, the Full Covariance GMM model now shows clear signs of overfitting from n_components = 16 on, making the best candidate the model with 8 components, instead of the 16 components one as observed in the validation phase.

minDCF - GMM whole dataset

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
Full Covariance GMM (n_comp = 2)	0.156	0.256	0.764
Full Covariance GMM (n_comp = 4)	0.12	0.239	0.529
Full Covariance GMM (n_comp = 8)	0.11	0.218	0.586
Full Covariance GMM (n_comp = 16)	0.123	0.23	0.525
Diag Covariance GMM (n_comp = 8)	0.131	0.26	0.553
Tied Covariance GMM (n_comp = 2)	0.139	0.276	0.58
Tied Diag Covariance GMM (n_comp = 16)	0.151	0.326	0.581

4.5 Score calibration

As previously done, we will show the actualDCF of the candidate models.

minDCF and actualDCF for uncalibrated scores (test set)

	$\tilde{\pi}$=0.5		$\tilde{\pi}$=0.1		$\tilde{\pi}$=0.9	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
MVG - Tied Full Covariance	0.11	0.202	0.207	0.278	0.591	1.338
Quadratic SVM (K=1, C =0.1, π_T=0.5)	0.098	0.101	0.205	0.282	0.451	0.847

The actDCF values are far from optimal, so they will need re-calibration. Training the LogReg meta-model for calibration now would be "cheating", because we cannot assume that the test labels are at our disposal when deploying a model for testing on real unknown data. The transformation pair (w, b) obtained back in the validation phase with the training scores were stored for this purpose. The

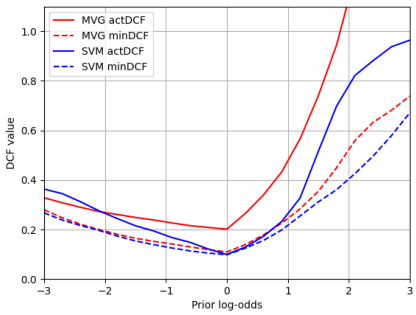

Fig. 26: Bayes error plot for uncalibrated MVG and quadratic SVM scores

calibration in this phase will only consist on applying the transformation onto the test scores. Below are the results:

minDCF and actDCF for **calibrated scores**

	$\tilde{\pi}$=0.5		$\tilde{\pi}$=0.1		$\tilde{\pi}$=0.9	
	minDCF	actDCF	minDCF	actDCF	minDCF	actDCF
MVG - Tied Full Covariance	0.11	0.115	0.207	0.222	0.591	0.611
Quadratic SVM (K=1, C =0.1, π_T=0.5)	0.098	0.101	0.205	0.215	0.451	0.476

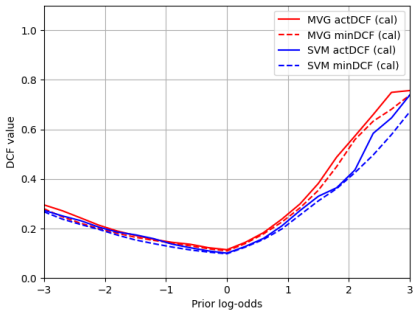

Fig. 27: Bayes error plot for calibrated MVG and quadratic SVM scores

The calibration was successful as the test scores are well calibrated over a wide range of applications.

4.6 Fusion Model

The approach for obtaining the Fusion model in this stage is to train each of the candidate models with all the training data and keep the scores. As in the calibration procedure, we don't train the meta-model again but we just stack the scores vertically and apply the transformation (w, b) obtained and stored during the validation phase.

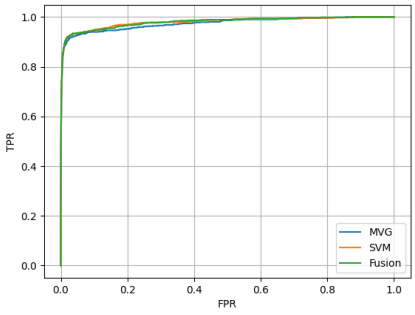

Fig. 28: ROC curve

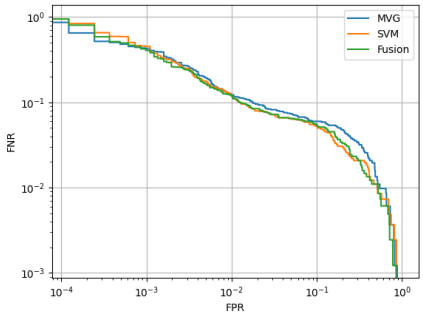

Fig. 29: DET curve

In terms of error trade-offs along the varying thresholds, the Fusion model performs slightly better than the candidate models through most of the tested thresholds, as it can be seen on the DET curve.

minDCF - Fusion Model on test set

	$\tilde{\pi}$=0.5	$\tilde{\pi}$=0.1	$\tilde{\pi}$=0.9
No PCA			
Fusion	0.101	0.198	0.465
MVG - Tied Full Covariance	0.11	0.207	0.591
Quadratic SVM (K=1, C =0.1, π_T=0.5)	0.098	0.205	0.451
PCA m= 7			
Fusion	0.114	0.226	0.523
MVG Tied Full Cov	0.147	0.268	0.622
Quadratic Kernel SVM (K = 1, C = 0.1, π_T = 0.5)	0.112	0.218	0.477

The fusion model performs similar the candidate models, as in the validation phase and is provides well-calibrated scores over a wide range of applications.

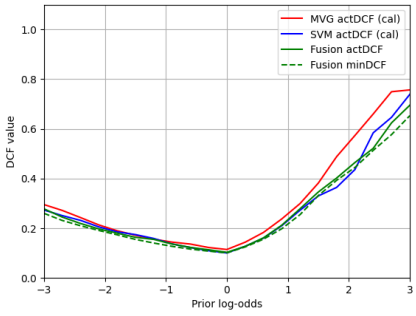

Fig. 30: Bayes error plot for calibrated Fusion modelMVG and quadratic SVM scores

5 Conclusion

After the analysis we ended up with well performing models. We are able to achieve a minDCF cost of around 0.1 for the main application point ($\tilde{\pi} = 0.5$), and minDCF cost of around 0.2 and 0.46 for the application points $\tilde{\pi} = \{0.1, 0.9\}$, respectively. The choices made on our training / validation sets proved to be effective also for the evaluation data.

6 References

1. 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.

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