Index

[Abstract 2](#_Toc121933597)

[HTRU2 Data Set 2](#_Toc121933598)

[Histograms 4](#_Toc121933599)

[Correlation 8](#_Toc121933600)

[Validation approach 10](#_Toc121933601)

[Multivariate Gaussian Classifiers (MVG) 12](#_Toc121933602)

[Logistic Regression 14](#_Toc121933603)

[Linear Logistic Regression 14](#_Toc121933604)

[Quadratic Logistic Regression 16](#_Toc121933605)

[Support Vector Machine (SVM) 18](#_Toc121933606)

[Linear SVM 18](#_Toc121933607)

[Quadratic SVM 20](#_Toc121933608)

[RBF SVM 21](#_Toc121933609)

[Gaussian Mixture Models 24](#_Toc121933610)

[Model selection 26](#_Toc121933611)

[Score calibration 28](#_Toc121933612)

[Combining classifiers(fusion approach) 33](#_Toc121933613)

[Experimental results on evaluation set 35](#_Toc121933614)

[MVG 35](#_Toc121933615)

[Linear Logistic Regression 36](#_Toc121933616)

[Quadratic Logistic regression 37](#_Toc121933617)

[Linear SVM 38](#_Toc121933618)

[Quadratic SVM 39](#_Toc121933619)

[RBF SVM 40](#_Toc121933620)

[GMM 40](#_Toc121933621)

[Score Calibration 42](#_Toc121933622)

[Fusion 43](#_Toc121933623)

[References 43](#_Toc121933624)

# Abstract

This report analyzes the results of different machine learning algorithms applied to the HTRU2 Data Set [[1]](../../../../../UCI%20Machine%20Learning%20Repository_%20HTRU2%20Data%20Set.html). The final goal is to individuate which models obtain best performance in classification of HTRU2 samples.

Initially an analysis of the features of the dataset is carried out. Then different classifiers are considered: Multivariate Gaussian Models (MVG), Logistic Regression (linear and quadratic), Support Vector Machine (linear, RBF and quadratic), Gaussian Mixture Models and Fusion. To select best hyperparameters for each model a **validation dataset** is extracted from training data set using K-fold cross-validation. First the results of classifiers are measured in terms of minimum Detection Cost Function (min DCF), then also actual DCF is taken in account with a discussion about **score calibration**.

In the end the best models with the selected hyperparameters are tested on the **evaluation set** to verify if the models works well on unseen (evaluation) data.

We will show that all the classifiers provide similar results on HTRU2 dataset, though quadratic classifiers performs slightly better. A Fusion model between Quadratic Logistic Regression and RBD SVM will be selected as best model.

# HTRU2 Data Set

HTRU2 is a data set which describes a sample of pulsar candidates collected during the High Time Resolution Universe Survey (South) [2].

**Pulsars** are a rare type of Neutron star that produce radio emission detectable here on Earth.

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 sample in the HTRU2 Data set is a **'candidate'**, a potential signal detection known 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 (actual pulsar) hard to find.

HTRU2 contains a set of pulsar candidates, the actual pulsar are labeled with 1, while the others samples are labeled with 0. More in details the HTRU2 data set contains 16,259 spurious examples caused by RFI/noise, and 1,639 real pulsar examples. These examples have all been checked by human annotators. The HTRU2 is thus **highly imbalanced**: the legitimate pulsar examples are a minority positive class, and spurious examples the majority negative class.

**Classification systems** which treat the candidate data sets as **binary classification problems** can be employed to automatically label pulsar candidates to facilitate rapid analysis. Results of these classifiers are reported in the next sections of this document.

Each candidate is described by **8 continuous variables**. 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 (see [3] for more details). The remaining four variables are similarly obtained from the DM-SNR curve (again see [3] for more details). Each samples has the features (where the last one is the label) summarized below:

0. Mean of the integrated profile.

1. Standard deviation of the integrated profile.

2. Excess kurtosis of the integrated profile.

3. Skewness of the integrated profile.

4. Mean of the DM-SNR curve.

5. Standard deviation of the DM-SNR curve.

6. Excess kurtosis of the DM-SNR curve.

7. Skewness of the DM-SNR curve.

8. Class

The HTRU2 contains 17,898 total examples: 1,639 positive examples (labeled with 1) and 16,259 negative examples (labeled with 0).

The dataset has been split into **Train** and Evaluation (**Test**) data.

The training set contains 8108 negative examples and 821 positive examples.

The test set contains 8151 negative examples and 818 positive examples.

HTRU2 contains 8 continuous variables. The mean (**µ**) and standard deviation (**σ**) of each features for the **training dataset** are:

It can be observed that features of the data set have different scales, they have large differences between their ranges. So, in this case, **Z-normalization** on the data-set to bring all the features on the same scale could be useful. Z-normalization centers the feature columns at mean 0 with standard deviation 1. Thus before applying any operation each sample of the training set has been transformed through the expression:

Where is the sample after the Z-score normalization, while is the original sample in the original data set.

In next sections what will be referred as “**raw features**” (or raw data) are actually the original data Z-normalized.

## Histograms

It is useful to analyze the distribution of the features, since some of the classifiers applied in the following make assumptions on the distribution of data. First the histogram of features of different classes for the entire training data set are plotted:



The 8 features result easy separable. The histograms shows that in the most of the cases features of class 0 are concentrated around a certain value. The shape of the histograms of features of class 0 is Gaussian like. On the other hand the features of class 1 results more spread and irregular.

Since one of the technique we will apply is Multivariate Gaussian Classifier (which assumes that data are drawn from a Gaussian distribution), we will apply **Gaussianization** pre-processing stage which tries to transform our features sample as if they behave more likely as a Gaussian distribution.

Gaussianization allows to transform features in such a way that the empirical cumulative distribution of the transformed features is the same as that of a Gaussian distribution. In particularly we will consider as a target a Gaussian distribution with unit variance and zero mean. To perform Gaussianization we will compute empirical cumulative distribution function (c.d.f.) on the features, we will map this c.d.f. to a uniform distribution and then we will transform it to Gaussian c.d.f.

For each feature x we have to compute the **rank** as:

the rank basically counts the number of samples of the features that have a lower value than and divide it by the number of samples . The rank expression adds 1 to the counts the number of samples of the features that have a lower value than and adds 2 to number of samples for numerical reasons: rank(x)=0 should be avoided because it would lead to a gaussianized sample mapped to minus infinity.

is basically the empirical cumulative distribution for the data. Each sample is replaced with a sample that has the same value for the cumulative distribution but the new sample will have a cumulative distribution assumed to be taken from a Gaussian density. The purpose of Gaussianization is to find a value such that the cumulative distribution of is equal to the rank of .

We assume that is the normal gaussian cumulative distribution. Solving with respect to y we obtain:

Where is called **percent point function (p.p.f)** and is the inverse of the cumulative distribution function of the standard normal distribution.

Scipy library’s implementation of p.p.f is used in the code [4]

Gaussianization should be applied to both training and evaluation sets. The ranking is computed using only the training data; evaluation data are single samples, ranking of the evaluation samples is computed comparing them to the training samples.

Below histograms of each feature after Gaussianization are showed.



Gaussianization process ignore the classes, it works considering the whole dataset, it makes the **whole dataset** more Gaussian like, so samples of one class could be quite different from a Gaussian distribution.

It can be observed that HTRU2 features of class 0 after Gaussianization have a shape very similar to a Gaussian distribution. Though features of class 1 after Gaussianization they are still quite asymmetrical. So, we don’t expect that Gaussianization will greatly improve the results on MVG and Gaussian Mixture Models (GMM). We can expect that Gaussianization will be even less effective on other classifiers. However in the following we will compare the results of classifiers applied to the samples Gaussianized and not Gaussianized (**raw features**) to check which is the best choice.

## Correlation

To analyze the interaction of features we can compute the **correlation** of features. This is useful because it suggest if PCA could be useful and how many features can be discarded.  
**Pearson correlation coefficient** is considered to measure correlation, it can be computed as:

Actually absolute value of Pearson correlation is considered because we are only interested to understand if there is correlation or not

The abs Pearson correlation coefficient can take value between 0 and 1. If 0 it means that the 2 features are uncorrelated, while 1 means that the features are completely correlated: actually one feature is the scaled version of the other.  
To plot the correlation between features we use heatmap where darker color means high correlation between 2 features, while lighter color means low correlation between 2 features.  
We plot the correlation considering: the whole data training samples (left); training samples belonging to class 0 (right); training samples belonging to class 1 (center).

Correlation of features without Gaussianization. On the left all the data set is considered; in the center only samples of class 1 are considered; on the right only the samples of class 0 are considered.



The strongest correlation can be observed on couples of features 6-7, 5-6, 0-1, 2-3. It may be worth to map the features to a space with lower dimensionality than 8.  We will use PCA, which is a statistical technique for reducing the dimensionality of a dataset. PCA finds a subspace, that is a set of directions over which to project our data set points. More in details PCA finds map projection that minimize the average reconstruction error (i.e. for which the reconstructed point is as close as possible to the original point).

In the following will be considered result of the classifiers after applying PCA up to m = 5.  
Furthermore these heatmaps suggest that Gaussian classifier with diagonal covariance matrix won’t perform better than full covariance Gaussian classifier since there is high correlations between features; anyway we will evaluate also the Gaussian classifiers with diagonal covariance matrix to compare them with other Gaussian classifiers.   
We can also plot the correlation of Gaussianized features:

The observation made for the non-Gaussianized features are still valid and even strengthened for the Gaussianized features because the correlation of the previous quoted couple of features seems to be increased.

Correlation of features after Gaussianization. On the left all the data set is considered; in the center only samples of class 0 are considered; on the right only the samples of class 1 are considered.



In particular, after Gaussianization the correlation between features that were strongly correlated before Gaussianization has increased, while the correlation between features that were weakly correlated before Gaussianization has decreased.

# Validation approach

In the following different models for classification will be considered. For each of them training data is used to first train and then validate the models. The goal of this job is to identify which kind of classifier performs better on HTRU2 data set and which is the best combination of hyperparameters. Costs of different classifiers type are compared. For the binary problem the **cost function** can be separated in 2 components: the goodness of classifier itself and the goodness of the threshold chosen. At the beginning the threshold problem is not considered, we just need to know which classifier better discriminate the scores, for this reason **minimum detection cost (min DCF)** will be used to measure the performance.

The problem of selecting an optimal threshold will be considered afterward.

*Explain more what min DCF is and how is it computed?*

To validate the models a **validation set** can be extracted from training set. Models are trained using training set (without validation set samples) then scores and min DCF is computed on validation samples. The model can be trained with different combinations of hyperparameters, results (min DCF computed on the scores of validation samples) of the same model with different hyperparameters are compared to find the optimal ones.

A validation set can be extracted with two different methods:

* Split the training set in 2 different set: one used for training the models actually; the other set used for validation (**single split**)
* Employ **K-fold cross-validation**

With the single split approach the final classifier will be the same evaluated on the validation set. In this case not all the samples available on of training data set will be used to build the final classifier. The big advantage of single split is that only one model need to be trained, but on the other hand fewer data for training and evaluation are used and this could lead to less robust results.  
The K-fold cross validation consists of splitting the training data set in K folds: K-1 folds are used for training the model, while the last one is used for validation. This process is repeated K times, at each iteration is computed the score for the validation samples. At the end of K iterations, scores has been computed for each sample of training set and min DCF can be computed starting from this set of scores.

After hyperparameters have been selected, the final classifier will be obtained re-training over the whole training data set. So the K-fold cross validation has the advantage of having more data available for training and evaluation (at the end all the samples of training set have been used as training and validation samples). On the other hand decisions are made over the validation set for the models trained using folds and for this reason they may not be optimal for the mode learned from all training data. Furthermore K-fold cross-validation has a big con: require to train and validate models K+1 time and this could require lot of computational time.

Generally if feasible the K-fold cross-validation leads to more robust results rather than single split, for this reason in the following K-fold cross-validation is considered rather than single split. It is desirable to have a high value of K to have access to big amount of training data at each iteration. Thought the amount of time required to estimate model parameters increase with higher value of K, so in the following K= 5 will be considered. Previous attempts with higher value of K have been tried but they required too much computation time especially for the Support Vector Machine and Gaussian Mixture Models.

The results with single split are not reported in the following because it doesn’t make sense to compare the result of different validation approaches.

For each iteration of K-fold cross validation the scores relative to the evaluation set are computed. At the end these scores are put all together and a single min DCF is computed.

Furthermore data are shuffled before splitting, so that the data of different folds are likely homogeneous.

It has not been specified which application should be used and which conditions the models will be operating. In the next sections of this document different applications are considered, results of validation and evaluation with different prior (*π*) are reported, in particularly the applications considered are:

* **Main target**: uniform prior application with (*π, Cfp, Cfn*) = (0.5, 1, 1)
* Unbalanced application with (*π, Cfp, Cfn*) = (0.1, 1, 1)
* Unbalanced application with (*π, Cfp, Cfn*) = (0.9, 1, 1)

# Multivariate Gaussian Classifiers (MVG)

First models considered are Gaussian classifiers. These are **generative models** (they try to model the class distribution of observed samples). Gaussian classifier assumes that both the training set and evaluation samples are **independent and identically distributed (i.i.d.)** given a set of parameters . In particular Gaussian distribution for a samples given the class is assumed:

The model parameters are estimated using Maximum Likelihood estimation approach. This lead to find:

Where for the HTRU2 dataset there are only 2 classes and . is the number of samples belonging to class .

Explain how to compute score and why for gaussian classifiers

We will consider different models with different assumptions to compute covariance matrix:

* **Full covariance matrix**(Full-cov): compute covariance matrix () without any simplifications. This model is expected to be robust if the number of training samples is far bigger than number of dimensions of samples. For the HTRU2 dataset good results are foreseen.
* **Diagonal covariance matrix** (Diag-cov): Multivariate Gaussian classifier with **diagonal covariance matrices** where the diagonal element of row-*i* are the variances of the

feature-*i* of the training samples. Diag-cov is a diagonal version of the original full covariance matrix (same elements on the diagonal, 0 in the other elements).This model work well in scenario where features are more or less **independent** (not correlated). In HTRU2 There is significant correlation between features, so the Gaussian classifier with diagonal hypothesis is expected to have poorer results than full covariance Gaussian classifier.

* **Tied covariance matrix**: assume that Gaussian parameters of different classes are related in one to each other. In particularly this model assumes that covariance matrices of different classes are the same, there is only one covariance matrix. This single covariance matrix will somehow corresponds to a weighted average empirical covariance matrix for each class.  
  Both full tied-covariance (**Full Tied-cov**) matrix and diagonal tied-covariance (**Diag Tied-cov**) are considered.

This model works well when there are classes with few samples. For HTRU2 The tied hypothesis is expected to have good performance because the training data set is highly unbalanced and the class 1 of pulsar samples have fewer samples.

Min DCF results for MVG (with different assumptions on covariance matrix) with K-fold cross validation (K=5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** |  | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.141 | 0.284 | 0.669 | 0.153 | 0.246 | 0.697 |
| Diag-Cov | 0.193 | 0.314 | 0.746 | 0.153 | 0.276 | 0.606 |
| Tied Full-Cov | 0.112 | 0.223 | 0.574 | 0.131 | 0.232 | 0.534 |
| Tied Diag-Cov | 0.161 | 0.265 | 0.580 | 0.163 | 0.291 | 0.614 |
| Raw features – PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.139 | 0.301 | 0.643 | 0.153 | 0. 245 | 0.690 |
| Diag-Cov | 0.214 | 0.505 | 0.724 | 0.163 | 0.246 | 0.660 |
| Tied Full-Cov | 0.112 | 0.222 | 0.573 | 0.134 | 0.243 | 0.533 |
| Tied Diag-Cov | 0.138 | 0.270 | 0.599 | 0.137 | 0.253 | 0.562 |
| Raw features – PCA m = 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.153 | 0.287 | 0.648 | 0.154 | 0.241 | 0.695 |
| Diag-Cov | 0.223 | 0.525 | 0.721 | 0.156 | 0.239 | 0.643 |
| Tied Full-Cov | 0.140 | 0.258 | 0.582 | 0.136 | 0.247 | 0.545 |
| Tied Diag-Cov | 0.164 | 0.297 | 0.590 | 0.141 | 0.255 | 0.588 |
| Raw features – PCA m = 5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.150 | 0.250 | 0.642 | 0.152 | 0.246 | 0.698 |
| Diag-Cov | 0.220 | 0.453 | 0.733 | 0.156 | 0.240 | 0.633 |
| Tied Full-Cov | 0.150 | 0.261 | 0.574 | 0.136 | 0.247 | 0.542 |
| Tied Diag-Cov | 0.171 | 0.311 | 0.604 | 0.141 | 0.256 | 0.594 |

Tied Full covariance MVG obtains the best results both in raw and Gaussianized features with and without PCA; this is due to the fact that class 1 samples are fewer than class 0 samples.

Diagonal covariance matrix MVG performs worse than full covariance matrix since, as shown before, there is strong correlation between features and the diagonal hypothesis is not accurate.

Finally, the results related to unbalanced application with are far from results of other applications since the HTRU2 dataset actually has much more samples belonging to class 0 rather than samples belonging to class 1. Bad outcomes for unbalanced application with are expected also for the other classifier models.

Gaussianization doesn’t improve the performance at all, in fact the best results are obtained with raw features. PCA with m = 7 provide the best results, even though they are quite similar to results without applying PCA. While outcomes with PCA with m<7 are even worse than outcomes on data without PCA.

So, best MVG model is Tied Full covariance MVG trained on raw features with PCA m=7.

# Logistic Regression

The attention is now turned to **discriminative models** starting from Logistic Regression. Discriminative models try to directly model the class posterior distribution, rather than modelling the distribution of observed samples (generative models).

It can be expected that PCA has limited effects on Logistic Regression models, as well as Gaussianization, since discriminative models does not require specific assumptions on data distribution.

## Linear Logistic Regression

First we consider a **regularized Linear Logistic Regression.** This model has 2 parameters that can be found minimizing the expression reported here:

expression correspond to the average **cross-entropy** between the distribution of observed and predicted labels plus a **regularization term**  .is a hyper-parameter that allows specifying the relative weight of the regularization term.

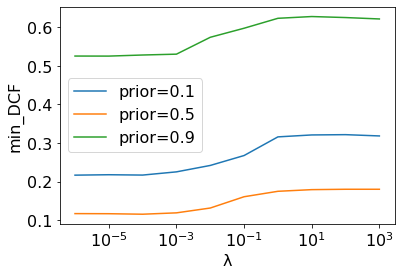
Since classes on training set are not balanced, the expression has been modified as:

Initially is considered the main application, thus πT is set to 0.5. Then results with πT = 0.1 and

πT =0.9 will be explored.

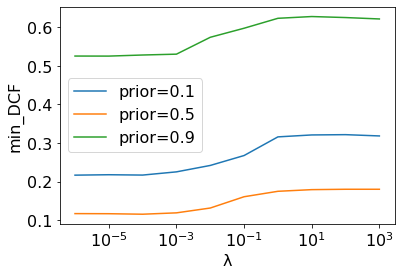
*scipy.optimize.fmin\_l\_bfgs\_b*[7] is employed to find minimum of . Then **score** of a sample can be computed as:

To understand which is a good value for , min DCF for logistic regression with different values of are computed. The results are reported for raw (left) and Gaussianized (right) features:



Linear Logistic Regression (πT=0.5) : Min DCF with respect to ––

Raw features – no PCA



Linear Logistic Regression (πT=0.5) : Min DCF with respect to ––

Raw features – no PCA

Values of ranging from 10-6 to 103 are considered. The x-axis is reported in logarithm scale.

The Gaussianization does not affect the results, the plots remain quite similar both for raw and Gaussianized features. However in the following also the results for Gaussianized features are reported to confirm it.

From these plot we can observe that terms does not help in reducing the minDCF. Considering the plot of the application with prior π=0.5 the lowest value of minDCF corresponds to the lowest value of (10-6). So, best results are expected for with very low values. is considered.

Min DCF results for Linear Logistic Regression (with different values of πT) with K-fold cross validation (K=5).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.117 | 0.216 | 0.526 | 0.124 | 0.232 | 0.513 |
| LogReg(λ=10-6, πT=0.1 ) | 0.113 | 0.211 | 0.555 | 0.127 | 0.229 | 0.523 |
| LogReg(λ=10-6, πT=0.9 ) | 0.119 | 0.220 | 0.511 | 0.127 | 0.230 | 0.515 |
| Raw features – PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.114 | 0.215 | 0.542 | 0.124 | 0.237 | 0.512 |
| LogReg(λ=10-6, πT=0.1 ) | 0.112 | 0.209 | 0.561 | 0.129 | 0.231 | 0.521 |
| LogReg(λ=10-6, πT=0.9 ) | 0.116 | 0.217 | 0.528 | 0.127 | 0.238 | 0.507 |
| Tied Full-Cov | 0.112 | 0.222 | 0.573 | 0.134 | 0.243 | 0.533 |
| Raw features – PCA m= 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.126 | 0.242 | 0.578 | 0.122 | 0.232 | 0.517 |
| LogReg(λ=10-6, πT=0.1 ) | 0.126 | 0.236 | 0.565 | 0.128 | 0.228 | 0.533 |
| LogReg(λ=10-6, πT=0.9 ) | 0.134 | 0.240 | 0.523 | 0.124 | 0.234 | 0.530 |
| Raw features – PCA m = 5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.136 | 0.230 | 0.602 | 0.121 | 0.234 | 0.541 |
| LogReg(λ=10-6, πT=0.1 ) | 0.134 | 0.236 | 0.598 | 0.127 | 0.228 | 0.531 |
| LogReg(λ=10-6, πT=0.9 ) | 0.143 | 0.242 | 0.555 | 0.128 | 0.236 | 0.534 |

On the main application () and on application with prior all the three models obtain similar results, Logistic Regression performs slightly better in this two cases.

On the unbalanced application with prior the outcomes are quite worse as expected, in this case Logistic Regression would be the best choose.

As expected Gaussianization doesn’t improve the performance, except for application with prior .

Linear Logistic Regression on PCA with m = 7 obtains the best results, even though they are quite similar to results without applying PCA. While outcomes with PCA with m<7 are even worse than outcomes on data without PCA.

So, according to validation results, best Linear Logistic Model has and is trained on raw features with PCA = 7. This model has similar values to finest MVG model (full tied-covariance trained on raw features with PCA m=7).

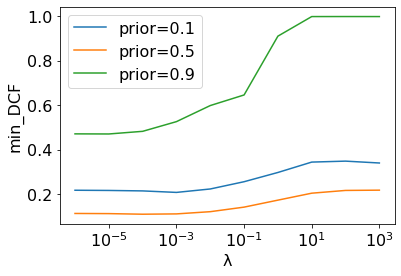
## Quadratic Logistic Regression

To compute quadratic separation surfaces we define an expanded feature space

Where is the operator that stacks the columns of matrix .

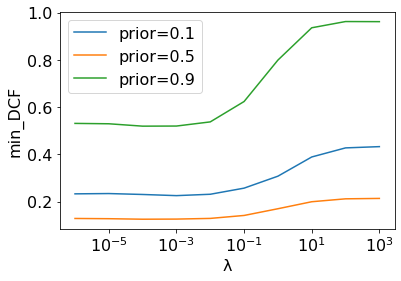
We can train a Logistic Regression model using rather than . We will obtain a model that has linear separation surface in the expanded space, but actually estimates quadratic separation surfaces in the original space.

As for Linear Logistic Regression, min DCF for logistic regression with different values of are computed to identify which is the best value for . The results are reported for raw (left) and Gaussianized (right) features:



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect to ––

Raw features – no PCA



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect to ––

Gaussianized features – no PCA

As for linear regression terms does not help in reducing the minDCF. is considered

Min DCF results for Linear Logistic Regression (with different values of πT) with K-fold cross validation (K=5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quad Log Reg (λ=10-6, πT=0.5 ) | 0.115 | 0.219 | 0.472 | 0.129 | 0.233 | 0.531 |
| Quad Log Reg (λ=10-6, πT=0.1 ) | 0.109 | 0.207 | 0.501 | 0.131 | 0.214 | 0.524 |
| Quad Log Reg (λ=10-6, πT=0.9 ) | 0.125 | 0.236 | 0.522 | 0.131 | 0.259 | 0.537 |
| Raw features –PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quad Log Reg (λ=10-6, πT=0.5 ) | 0.114 | 0.213 | 0.475 | 0.129 | 0.234 | 0.531 |
| Quad Log Reg (λ=10-6, πT=0.1 ) | 0.111 | 0.205 | 0.479 | 0.131 | 0.207 | 0.526 |
| Quad Log Reg (λ=10-6, πT=0.9 ) | 0.123 | 0.234 | 0.509 | 0.131 | 0.259 | 0.537 |
| Linear Log Reg(λ=10-6, πT=0.1 ) | 0.112 | 0.209 | 0.561 | 0.129 | 0.231 | 0.521 |
| Raw features –PCA m= 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quad Log Reg (λ=10-6, πT=0.5 ) | 0.115 | 0.213 | 0.481 | 0.126 | 0.227 | 0.250 |
| Quad Log Reg (λ=10-6, πT=0.1 ) | 0.116 | 0.205 | 0.508 | 0.130 | 0.216 | 0.553 |
| Quad Log Reg (λ=10-6, πT=0.9 ) | 0.124 | 0.224 | 0.499 | 0.129 | 0.250 | 0.511 |
| Raw features –PCA m = 5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quad Log Reg (λ=10-6, πT=0.5 ) | 0.112 | 0.211 | 0.529 | 0.127 | 0.232 | 0.555 |
| Quad Log Reg (λ=10-6, πT=0.1 ) | 0.116 | 0.211 | 0.529 | 0.128 | 0.211 | 0.553 |
| Quad Log Reg (λ=10-6, πT=0.9 ) | 0.117 | 0.234 | 0.540 | 0.130 | 0.244 | 0.553 |

Quadratic Logistic regression with λ=10-6 and πT=0.1 provides better result than the linear models we considered so far. As for Linear Logistic Regression PCA doesn’t improve the performance. And finally as we expected Gaussianization it’s not useful.

# Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised discriminative model for classification. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other. Training samples are mapped to points in space so as to maximize the **margin**: width of the gap between the two categories.

## Linear SVM

Linear SVM looks for separation hyperplane which maximizes the margin. The linear SVM objective consists in minimizing the expression (**primal formulation**):

Where C is an hyperparameter; actually include also the bias term b and is extended with 1:

The **dual formulation** of the linear SVM is used. This can be obtained using the Lagrange multiplier and become the maximization of the expression:

Where

For solving the maximization of the dual formulation of SVM *scipy.optimize.fmin\_l\_bfgs\_b* [6] is used. This implementation compute minimizer of a function, so actually minimization of is considered rather than maximization of .

Once solution with respect to is computed, the primal solution can be retrieved as:

At this point the **score** for a sample is computed as:

To make the classes balanced different values of hyperparameter C for the 2 different classes are used:

* For samples of class 1 the hyperparameter considered is
* For samples of class 1 the hyperparameter considered is

Where and are the empirical priors, i.e. number of samples of class 1 and 0 over the total number of samples in training set. In the following are considered linear SVM with

; , as well as the linear SVM without re-balance. .

For linear SVM we need to tune the hyperparameter C. As we did for selecting λ in Logistic Regression section, min\_DCF with respect to different values of C is plotted and the optimal value of C is chosen from this plot. Linear SVM with is considered for this plot.



Linear SVM (): Min\_DCF with respect to C - Gaussianized features – no PCA

Linear SVM (): Min\_DCF with respect to C –

raw features – no PCA



C

In the range between 10-2 ­and 102 the min DCF is quite steady both for Gaussianized and raw data. Outside this range the min DCF has higher values. C=1 as optimal value for the hyperparameter is selected. As for Logistic Regression models Gaussianization is not expected to improve the results, nevertheless min DCF for both raw and Gaussianized features are reported to ensure this.

Min DCF results for Linear SVM (with different values of πT) with K-fold cross validation (K=5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.5 ) | 0.112 | 0.221 | 0.553 | 0.120 | 0.252 | 0.516 |
| Linear SVM(C=1, πT =0.1 ) | 0.113 | 0.213 | 0.543 | 0.128 | 0.228 | 0.515 |
| Linear SVM(C=1, πT =0.9 ) | 0.123 | 0.227 | 0.539 | 0.130 | 0.250 | 0.502 |
| Linear SVM(C=1, no-rebalancing) | 0.112 | 0.213 | 0.551 | 0.130 | 0.250 | 0.502 |
| Raw features –PCA m=7 | | | | Gaussianized features – PCA m=7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.5 ) | 0.112 | 0.220 | 0.559 | 0.120 | 0.254 | 0.507 |
| Linear SVM(C=1, πT =0.1 ) | 0.112 | 0.213 | 0.554 | 0.131 | 0.226 | 0.516 |
| Linear SVM(C=1, πT =0.9 ) | 0.120 | 0.225 | 0.537 | 0.136 | 0.250 | 0.505 |
| Linear SVM(C=1, no-rebalancing) | 0.113 | 0.212 | 0.554 | 0.132 | 0.225 | 0.514 |
| LogReg(λ=10-6, πT=0.1 ) | 0.112 | 0.209 | 0.561 | 0.129 | 0.231 | 0.521 |
| Tied Full-Cov | 0.112 | 0.222 | 0.573 | 0.134 | 0.243 | 0.533 |
| Raw features – PCA m=6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.1 ) | 0.121 | 0.236 | 0.608 | 0.130 | 0.228 | 0.530 |
| Raw features –PCA m=5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.1 ) | 0.123 | 0.231 | 0.628 | 0.130 | 0.228 | 0.532 |

Gaussianization improve performance just for application with prior , in general best results are obtained on raw features.

It can be observed that Linear SVM with πT =0.1, Linear SVM with πT =0.5 and SVM without rebalancing obtain similar results, while SVM with πT = 0.9 performs slightly worse. So, re-balance doesn’t improve results, but it doesn't even downgrade them.

As for Logistic Regression and MVG models PCA doesn’t improve the performance a lot: results on data with PCA m=7 are quite similar to results on data without PCA. While outcomes on PCA with m<7 are poorer than outcomes on data without PCA.

On the unbalanced application with prior the outcomes are quite worse than outcomes on application with prior and with prior as expected.

Linear SVM with πT =0.1, Linear SVM with πT =0.5 and Linear SVM without rebalancing on raw data without PCA (or with PCA m = 7) obtain similar results to best Linear Logistic Regression and best MVG classifiers analyzed before.

## Quadratic SVM

Considering the dual formulation of SVM problem, embedding a non–linear transformation only requires to perform dot–products in the expanded space. It is possible to compute a linear separation surface in the expanded space, which corresponds to a non–linear separation surface in the original feature space.

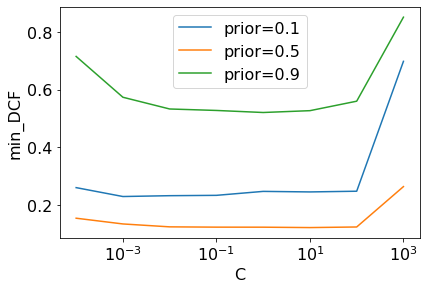
The **kernel function *k***computes the dot product in the expanded space:

The matrix H becomes:

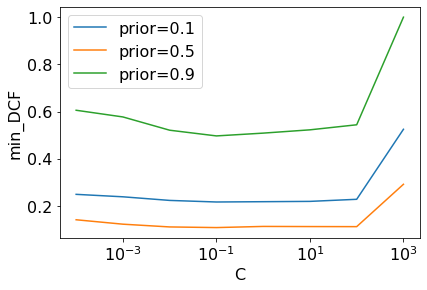
The quadratic (non-linear) SVM provides quadratic separation surfaces, it has a kernel computed as:

As for linear SVM the min DCF for different values of C is plotted to choose the best value of hyperparameter C.

observing the plots C=0.1 is selected.



Quadratic SVM (): Min\_DCF with respect to C - Gaussianized features – no PCA



Quadratic SVM (): Min\_DCF with respect to C - raw features – no PCA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.109 | 0.218 | 0.497 | 0.124 | 0.234 | 0.528 |
| Quadratic SVM(C=0.1, πT =0.1 ) | 0.112 | 0.206 | 0.550 | 0.135 | 0.216 | 0.567 |
| Quadratic SVM(C=0.1, πT =0.9 ) | 0.134 | 0.272 | 0.492 | 0.151 | 0.329 | 0.504 |
| Quadratic SVM(C=0.1, no-rebalancing) | 0.113 | 0.205 | 0.552 | 0.136 | 0.212 | 0.577 |
| Linear SVM(C=1, πT =0.5 ) | 0.112 | 0.221 | 0.553 | 0.120 | 0.252 | 0.516 |
| Raw features –PCA m=7 | | | | Gaussianized features – PCA m=7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.111 | 0.216 | 0.503 | 0.122 | 0.235 | 0.534 |
| Quadratic SVM(C=0.1, πT =0.1 ) | 0.112 | 0.203 | 0.573 | 0.135 | 0.214 | 0.564 |
| Quadratic SVM(C=0.1, πT =0.9 ) | 0.146 | 0.300 | 0.510 | 0.152 | 0.335 | 0.511 |
| Quadratic SVM(C=0.1, no-rebalancing) | 0.112 | 0.205 | 0.578 | 0.134 | 0.214 | 0.566 |
| Raw features – PCA m=6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.112 | 0.216 | 0.497 | 0.123 | 0.230 | 0.545 |
| Raw features –PCA m=5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.108 | 0.219 | 0.521 | 0.123 | 0.242 | 0.588 |

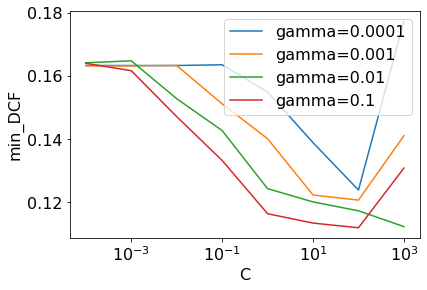
Min DCF results for Quadratic SVM (with different values of πT) with K-fold cross validation (K=5)

Quadratic SVM provides good results. Quadratic SVM(C=0.1, πT =0.5 ) preforms slightly better than linear SVM. As we expected Gaussianization it’s not useful. Extraordinarily lowest value of min DCF for the main application is obtained on (raw) data with PCA m=5, though results without PCA are very near to the ones obtained on raw data with PCA m=5.

## RBF SVM

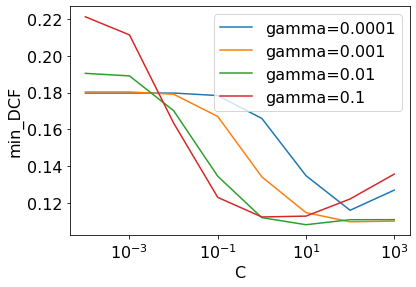
Another type of non-linear SVM can be obtained applying **Radial Basis Function (RBF)** kernel:

RBF introduces an additional hyperparameter which set width of the kernel. So, for this model 2 hyperparameters need to be selected (e.g. and ). To perform this choice min DCF with different with respect to for different values of is plotted:



RBF SVM (): Min\_DCF with respect to C and γ - Gaussianized features – no PCA

=0.01 (green) and C=10 are considered



RBF SVM (): Min\_DCF with respect to C and γ - raw features – no PCA

Min DCF results for RBF SVM (with different values of πT) with K-fold cross validation (K=5)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10,γ=0.01, πT =0.5 ) | 0.108 | 0.214 | 0.516 | 0.120 | 0.230 | 0.529 |
| RBF SVM(C=10, γ=0.01, πT =0.1 ) | 0.110 | 0.204 | 0.506 | 0.132 | 0.223 | 0.604 |
| RBF SVM(C=10, γ=0.01, πT =0.9 ) | 0.133 | 0.290 | 0.514 | 0.142 | 0.333 | 0.512 |
| RBF SVM(C=10, γ=0.01, no-rebalancing) | 0.117 | 0.204 | 0.528 | 0.133 | 0.225 | 0.599 |
| Linear SVM(C=1, πT =0.5 ) | 0.112 | 0.221 | 0.553 | 0.120 | 0.252 | 0.516 |
| Raw features –PCA m=7 | | | | Gaussianized features – PCA m=7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10, γ=0.01, πT =0.5 ) | 0.109 | 0.217 | 0.508 | 0.120 | 0.233 | 0.528 |
| RBF SVM(C=10, γ=0.01, πT =0.1 ) | 0.110 | 0.207 | 0.530 | 0.133 | 0.223 | 0.600 |
| RBF SVM(C=10, γ=0.01, πT =0.9 ) | 0.132 | 0.288 | 0.533 | 0.143 | 0.338 | 0.514 |
| RBF SVM(C=10, γ=0.01, no-rebalancing) | 0.112 | 0.205 | 0.524 | 0.133 | 0.225 | 0.599 |
| Raw features – PCA m=6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10, γ=0.01, πT =0.5 ) | 0.109 | 0.218 | 0.515 | 0.121 | 0.227 | 0.561 |
| Raw features –PCA m=5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10, γ=0.01, πT =0.5 ) | 0.110 | 0.219 | 0.544 | 0.122 | 0.228 | 0.577 |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.108 | 0.219 | 0.521 | 0.123 | 0.242 | 0.588 |

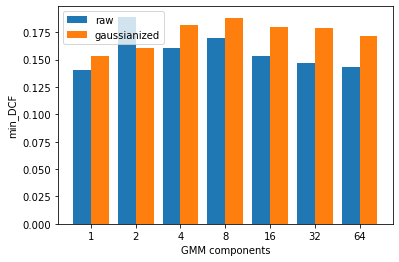
RBF SVM with C=10,γ=0.01, πT=0.5 is the best option for main application (π= 0.5). It obtains results comparable to other quadratic models analyzed. PCA is not effective but it does not downgrades the results too much (provides even better results for some options with the imbalanced application). Gaussianization doesn’t help at all.

# Gaussian Mixture Models

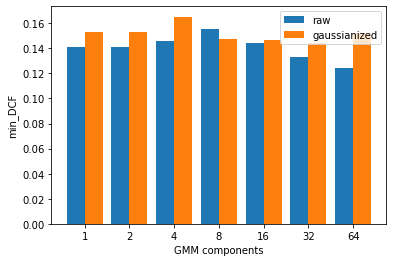
Last model considered is Gaussian Mixture Model (GMM). This a generative approach which can approximate generic distribution, there is no supposition on distribution of data like in the Multivariate Gaussian Models (MVG). So, GMM is expected to obtain better results than Gaussian models.  
Full covariance, diagonal covariance, full tied covariance and diagonal covariance models are considered. Fix width of covariance matrix is considered to avoid singular models and numerical issues.

First we plot the min DCF (without PCA, considering π= 0.5 and) for each GMM models with different number of components.

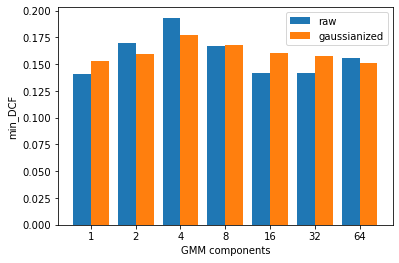
Diag-Tied-cov GMM: min DCF with respect to number of components – no PCA



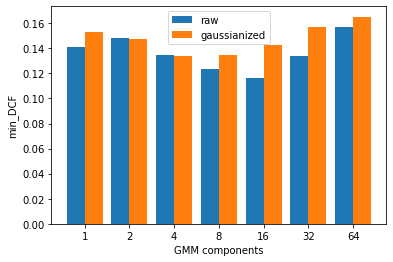
Full-Tied-cov GMM: min DCF with respect to number of components – no PCA



Diag-cov GMM: min DCF with respect to number of components – no PCA



Full-cov GMM: min DCF with respect to number of components – no PCA



For each GMM option we consider the number of components which provide the lowest min DCF. Here are shown the results of the selected GMM’s.

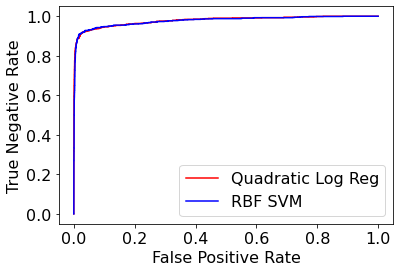
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** |  | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov, 16 Gaus | 0.117 | 0.233 | 0.545 | 0.142 | 0.229 | 0.675 |
| Diag-Cov, 32 Gaus | 0.141 | 0.260 | 0.634 | 0.158 | 0.268 | 0.609 |
| Tied Full-Cov, 64 Gaus | 0.124 | 0.233 |  | 0.152 | 0.229 | 0.682 |
| Tied Diag-Cov, 64 Gaus |  |  |  |  |  |  |
| Raw features – PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov, 16 Gaus | 0.117 | 0.219 | 0.536 | 0.132 | 0.214 | 0.648 |
| Diag-Cov, 32 Gaus | 0.132 | 0.237 | 0.647 | 0.137 | 0.231 | 0.648 |
| Tied Full-Cov, 64 Gaus | 0.133 | 0.260 | 0.630 | 0.153 | 0.228 | 0.656 |
| Tied Diag-Cov, 64 Gaus |  |  |  |  |  |  |
| Raw features – PCA m = 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov, 16 Gaus |  |  |  |  |  |  |
| Tied Full-Cov, 64 Gaus |  |  |  |  |  |  |

As we expected Gaussianization doesn’t improve the results at all. The best option is Full Covariance GMM with 16 components trained on Raw features without PCA or with PCA m=7. Though we can immediately observe that GMM doesn’t not provides better results than the other models we observed so far.

# Model selection

All the models considered up to here obtains similar results in terms of min DCF. Quadratic models performs slightly better. Quadratic Logistic Regression (λ=10-6, πT=0.1 ) and RBF SVM(C=10,γ=0.01, πT=0.5 ) trained or raw features without PCA are considered as candidate models. These 2 models will be considered for further analysis.

ROC curve is plotted to show behavior of the 2 selected models for different applications



ROC plot for selected models

As we expected, since the 2 models provided essentially same results, the 2 ROC curves are almost overlapped.

Solve this warning in code: C:\Users\lucaf\Desktop\PoliTo\PRIMO ANNO\MachineLearning\Project\ML-PulsarClassifier\Code\validate.py:94: RuntimeWarning: invalid value encountered in double\_scalars

TPR[idx]= Conf[1,1] / (Conf[1,1]+Conf[0,1])

C:\Users\lucaf\Desktop\PoliTo\PRIMO ANNO\MachineLearning\Project\ML-PulsarClassifier\Code\validate.py:95: RuntimeWarning: invalid value encountered in double\_scalars

FPR[idx]= Conf[1,0] / (Conf[1,0]+Conf[0,0])

# Score calibration

Up to now the metric min DCF has been used to compare the models. This is the cost if the optimal threshold is known. In other words, min DCF measures the cost to pay if optimal decisions for the evaluation set are made using the recognizer scores. However what it is paid in practice is not the minimum cost, but it is the **actual cost**. In practice it is not possible to know the optimal threshold for evaluation data, this would require to know the evaluation labels, but this is impossible.  
If scores are well calibrated the optimal threshold is a threshold that optimize the Bayes risk. If scores are well calibrated optimal threshold corresponds to the **theoretical threshold:**

where π is the effective prior.  
To compute actual cost starting from scores there could be 2 options:

1. **Score are well calibrated**: because recognizer has given well calibrated scores or because

re-calibration on recognizer output has been applied. In this case theoretical threshold

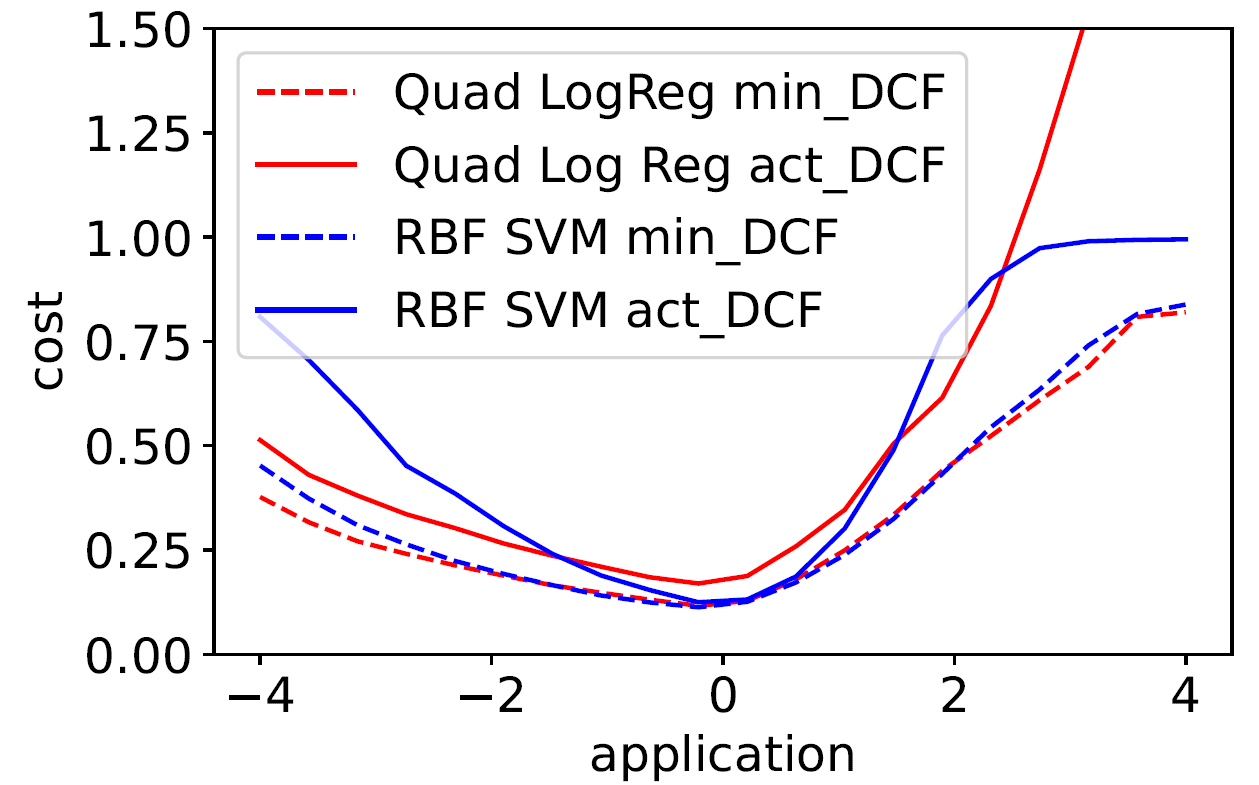
to compute actual cost.

1. **Scores are not well calibrated**: we need to estimate a good threshold for the target application (using validation set).

We can check if the scores are well calibrated comparing the min DCF with the actual cost obtained using the theoretical threshold for each application. This comparison is applied to the selected models to judge if the score calibration need to be applied or not.  
For the selected best models the minimum and actual cost for application with different prior are compared in the Bayes error plot:

Minimum VS actual DCF (with theoretical threshold) with respect to different application

priors



The value of actual DCF and minimum DCF for the applications considered (

) are reported in the table below:

Min and Actual DCF for selected models

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **π** | **π= 0.5** | | **π= 0.1** | | **π=0.9** | |
| Cost | Min DCF | Act DCF | Min DCF | Act DCF | Min DCF | Act DCF |
| Quadratic Log Reg (λ=10-6, πT=0.1 ) | 0.109 | 0.163 | 0.207 | 0.289 | 0.501 | 0.753 |
| RBF SVM(C=10,γ=0.01, πT=0.5 ) | 0.108 | 0.110 | 0.214 | 0.365 | 0.516 | 0.875 |

Observing the plot and the table results we can conclude that the scores of both models need to be calibrated since the actual DCF is quite far from minimum DCF except that for RBF SVM around main application (π= 0.5).

How the problem of not well calibrated scores can be solved? We need to estimate an application dependent threshold; given a target application we estimate an optimal threshold. To do this the threshold corresponding to minimum of the DCF on the validation set is taken.   
This approach doesn’t provide scores that are well-calibrated for different applications. For each application we have to re-estimate an optimal threshold on the validation set.  
Before actually using the threshold, it is needed to access whether the threshold is good before using it. Cross validation can be used to select the optimal threshold. Cross validation has an additional problem, we are working on the validation scores themselves, validation scores are treated as if they were a training set for which we want to solve threshold estimating problem. We take all the scores and we need to solve an estimation problem over the score and we want to evaluate how good is the solution.  
Basically the set of scores returned by the kfold function are treated as a training set over which finding optimal threshold. This set of scores will be splitted into training and validation set as well: the training set will be used to estimate the threshold, while the validation set will be used to estimate how good the threshold chosen is.

The results of this approach are:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **π** | **π= 0.5** | | | **π= 0.1** | | | **π=0.9** | | |
| Cost | Min DCF | Act DCF  **Theoretical threshold** | Act DCF  **t\*** | Min DCF | Act DCF  **Theoretical threshold** | Act DCF **t\*** | Min DCF | Act DCF  **Theoretical threshold** | Act DCF **t\*** |
| Quad Log Reg | 0.113 | 0.209 | 0. 128 | 0.220 | 0.315 | 0.261 | 0.365 | 0.816 | 0.? |
| RBF SVM | 0.105 | 0.108 | 0.110 | 0.220 | 0.408 | 0.231 | 0.418 | 0.874 | 0.454 |

As expected, actual DCF on theoretical threshold is higher than actual DCF on best threshold optimal threshold t\*. So this techniques provides benefits. Alternatively a second approach can be used to estimate optimal threshold. Scores can be transformed in a way that the theoretical threshold can be used. A calibration function Scal = f(s) take as input the scores output of a classifier, gives us new score that then can be compared with the theoretical threshold to obtain optimal decisions. The calibration function f(s) should be **monotone** because the higher non-calibrated scores should favor class 1 while lower non-calibrated scores should favor class 0.   
There are several approaches to do this, in the following **discriminative score model** is applied. This consists on training a new machine learning model directly on the score, a model that has outputs with probabilistic interpretation in terms of log-likelihood ratio. Why a new model is needed and it is not used the original classifier? The original classifier (for example logistic regression model) is trained in a feature space that is very large, small differences in the distribution between validation and evaluation populations may lead to significant errors in term of calibration. The new model for score calibration is trained in a very small space: the space of scores. Each sample will be described by a single score, so the new model will be trained on a one dimensional data. This gives a model with probabilistic interpretation is much stronger because less affected by issues related to overfitting, underfitting or mismatch in distributions of features. Furthermore since the data have one dimension no Z-normalization is needed and so the results score interpretation are no affected by regularization.  
So the idea is to use a transformation function that apply a linear scaling and then sum a byas term:

function f(s) should be interpreted as a **calibrated log-likelihood ratio** for the 2 classes:

If f(s) is a log-likelihood ratio we can use the theoretical threshold to assign the labels. The class posterior probability is obtained as:

So, for this new model we can interpret score as a feature and the log posterior ratio has a very similar form of the log posterior of logistic regression. If we set then log posterior ratio of this new model is actually the same as logistic regression.  
So, we can train a logistic regression model using as prior the application prior that we are interesting in, to estimate a linear transformation of the score that gives us well calibrated score.   
The linear transformation is estimated using the prior-weighted logistic regression model to learn the model parameters . The transformation of the score provided by this mode corresponds to class posterior ratio. The log-likelihood ratio can be obtained starting from the transformation compensating with the prior, the calibrated score f(s) can be retrieved as:

A prior must be specified, the calibration is still optimized for a specific application as for the previous approach. However this kind of approach the model provide good calibration results for different applications. Usually the score calibration solution with this approach works well for a wide range of applications.

|  |  |  |  |
| --- | --- | --- | --- |
| **π** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quad Log Reg(π=0.5) | Act DCF =0.117 | Act DCF =0.255 | Act DCF =0.367 |
| Quad Log Reg (π=0.1) | Act DCF = 0.119 | Act DCF = 0.250 | Act DCF = 0.366 |
| Quad Log Reg (π=0.9) | Act DCF =0.119 | Act DCF =0.267 | Act DCF =0.382 |
| Quad Log Reg  No calibration | Act DCF =0.309 | Act DCF =0.315 | Act DCF =0.816 |
| Quad Log Reg | Min DCF =0.113 | Min DCF =0.220 | Min DCF =0.365 |

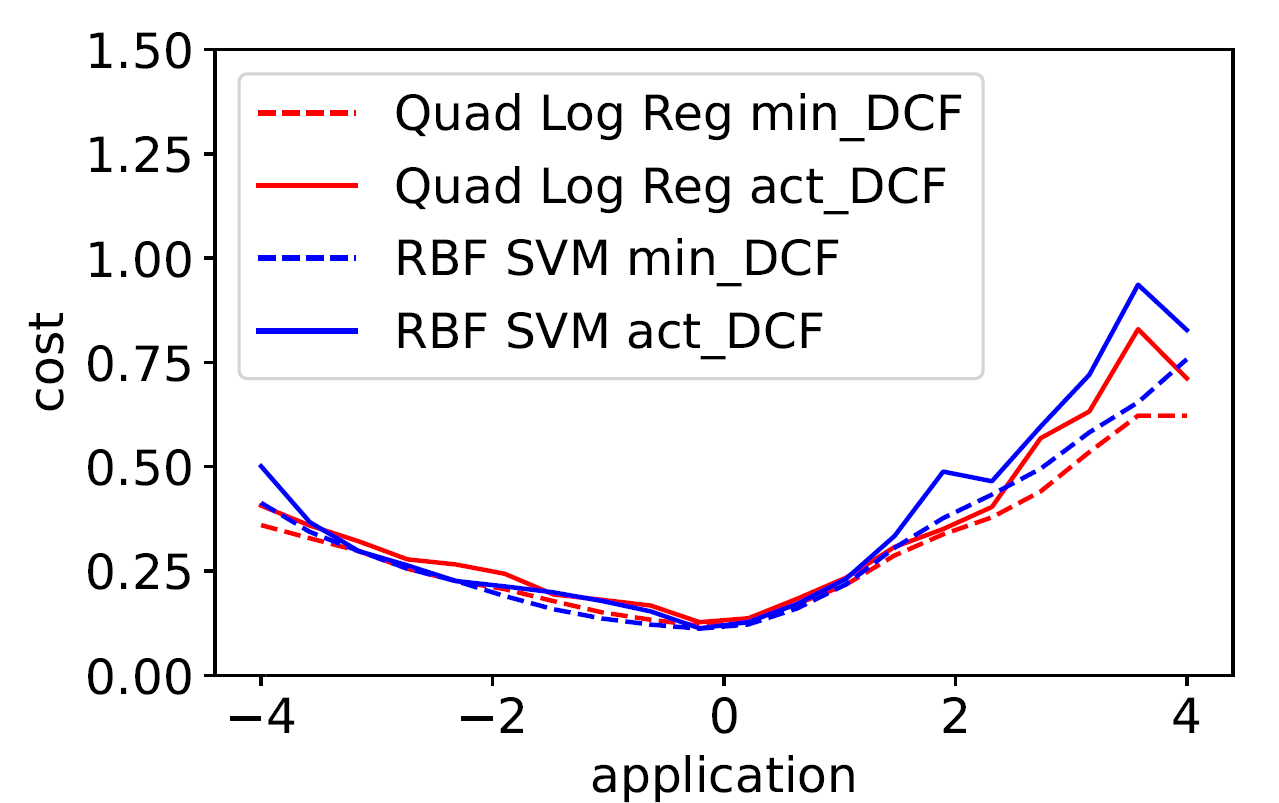
|  |  |  |  |
| --- | --- | --- | --- |
| **π** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM (π=0.5) | Act DCF =0.114 | Act DCF =0.225 | Act DCF =0.463 |
| RBF SVM (π=0.1) | Act DCF = 0.114 | Act DCF = 0.220 | Act DCF = 0.463 |
| RBF SVM (π=0.9) | Act DCF =0.114 | Act DCF =0.225 | Act DCF =0.457 |
| RBF SVM  No calibration | Act DCF =0.108 | Act DCF =0.408 | Act DCF =0.874 |
| RBF SVM | Min DCF =0.105 | Min DCF =0.220 | Min DCF =0.418 |

The calibration reduce the actual cost in every cases except for RBF SVM with prior π= 0.5, because in this latter case the actual DCF scores are already calibrated.

The results confirm that the value of the training prior π affects the result (there are only slight differences in the actual cost for the same application prior with different training prior).

Moreover this score calibration technique provides better outcomes than the approach based on optimal threshold analyzed before.

We can compute the Bayes error plot to check if scores of selected models are well-calibrated over a wide-range of applications.



Min DCF vs Act DCF for selected models with calibrated scores (π=0.5)

# Combining classifiers(fusion approach)

Different classifier based on different assumptions will provide different results; they may agree on some decisions while disagree on others. Combining the decisions of both may result in better predictions labels.   
Two classifiers can be combined with simple **voting scheme approach**: each classifiers assign a label and at the end the label assigned more often is selected. The simple voting approach has some issues, if one classifier is almost certain about class 1 and two other classifiers are only slightly in favor of class 0 it not granted that assigning label 0 is a good choice. So, rather than fusing classifiers at decision level, is better to perform a **score-level fusion** voting.   
The idea is to introduce a **fused score** which is a function of the scores of different classifiers. Considering a sample **x**t, if st,A is the score provided by classifier A, while st,B is the score provided by classifier B, the fused score for sample **x**t will be:

The scores of single classifiers should be already calibrated.  
A simple linear form for function f to fuse scores is:

are the parameters to be estimated. The scores of different classifiers are treated as a feature vector. A prior-weighted logistic regression is used to train the model parameters similar to what has been done for score calibration.

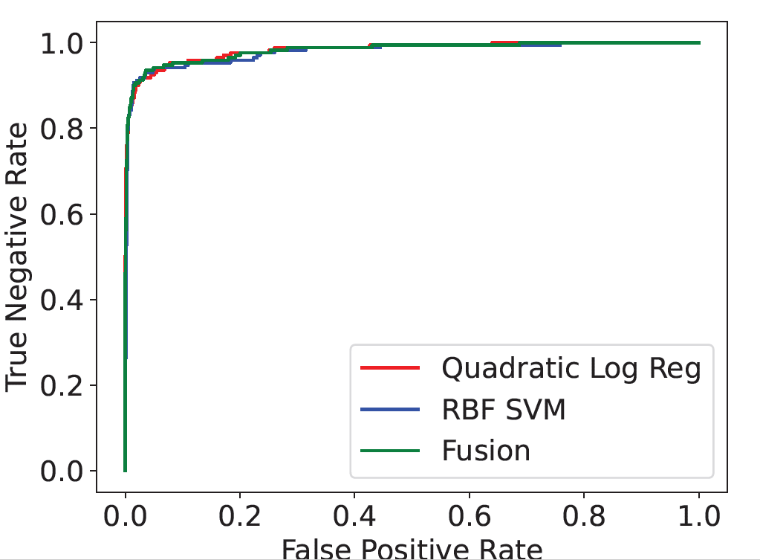
In the table below are reported the results of the fusion models compared with the results of selected models.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **π** | **π= 0.5** | | **π= 0.1** | | **π=0.9** | |
| Cost | Min DCF | Act DCF | Min DCF | Act DCF | Min DCF | Act DCF |
| Quadratic Log Reg (λ=10-6, πT=0.1 ) | 0.113 | 0.117 | 0.220 | 0.255 | 0.365 | 0.367 |
| RBF SVM(C=10,γ=0.01, πT=0.5 ) | 0.105 | 0.114 | 0.220 | 0.225 | 0.418 | 0.463 |
| Fusion | 0.100 | 0.113 | 0.213 | 0.238 | 0.388 | 0.414 |

Act DCF of Quadratic Logistic Regression and RBF SVM are reported considering calibrated scores (π=0.5).

We can also plot ROC curve to compare fusion with previously selected models.

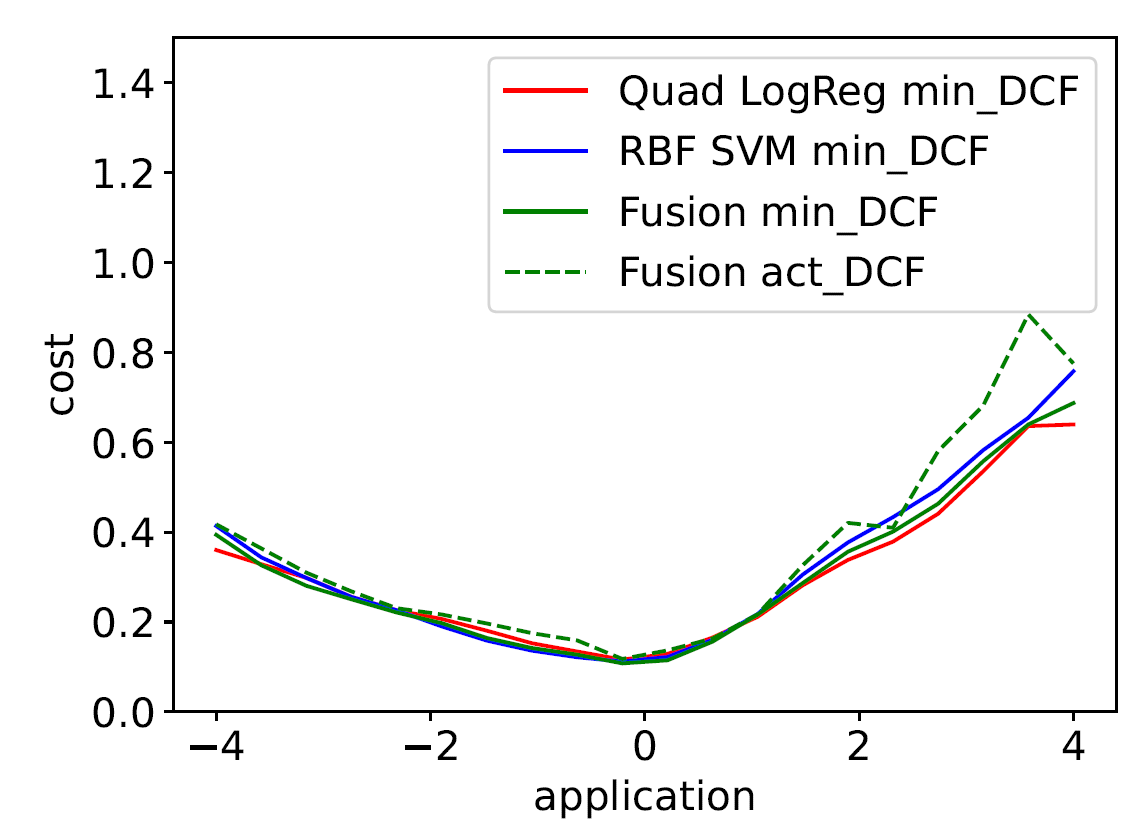
ROC curve for selected models and fusion



Fusion model provides slightly better results than selected Quadratic Logistic Regression and RBS SVM classifier. So, fusion can be selected as the final model.

The new scores of the fusion model are expected to be well calibrated, this can be checked with Bayes error plot:

Act and min DCF of fusion. Act DCF of selected models



# Experimental results on evaluation set

In this section all the classifiers will be evaluated using the evaluation set to check if the choices made before on validation set are consistent.

Each evaluation sample has been transformed through Z-normalization but using mean and standard deviation of training set:

Where is the evaluation sample after the Z-score normalization, while is the original evaluation sample. and are mean ad standard deviation of training set.

Both results on Gaussianized and raw data are compared. To Gaussianize evaluation samples we compute rank of each evaluation samples comparing them with training samples:

To apply PCA on evaluation set we compute the projection matrix P on training data and we compute the reduced space for evaluation data by multiplying the matrix P with the matrix that contains all the evaluation samples.

Minimum DCF is initially used as metric. Afterwards we will assess how good are actual decisions for the models evaluating the actual DCF. We will both consider the case where we estimated an optimal threshold per application on the validation set, and the case where scores were calibrated and the theoretical threshold was then used for different applications.

## MVG

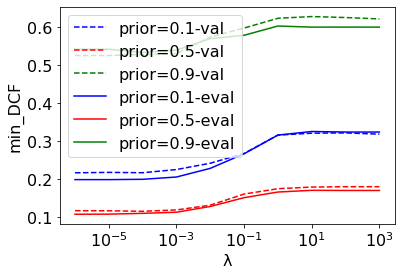
Min DCF results for MVG trained on the whole training set

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** |  | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.140 | 0.283 | 0.646 | 0.150 | 0.241 | 0.622 |
| Diag-Cov | 0.185 | 0.330 | 0.621 | 0.154 | 0.285 | 0.574 |
| Tied Full-Cov | 0.110 | 0.207 | 0.590 | 0.125 | 0.221 | 0.540 |
| Tied Diag-Cov | 0.151 | 0.262 | 0.544 | 0.158 | 0.301 | 0.568 |
| Raw features – PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.139 | 0.293 | 0.574 | 0.151 | 0.246 | 0.640 |
| Diag-Cov | 0.201 | 0.514 | 0.755 | 0.153 | 0.250 | 0.622 |
| Tied Full-Cov | 0.110 | 0.208 | 0.587 | 0.125 | 0.229 | 0.523 |
| Tied Diag-Cov | 0.140 | 0.257 | 0.556 | 0.127 | 0.242 | 0.518 |
| Raw features – PCA m = 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.146 | 0.277 | 0.600 | 0.147 | 0.249 | 0.628 |
| Diag-Cov | 0.218 | 0.544 | 0.725 | 0.150 | 0.251 | 0.593 |
| Tied Full-Cov | 0.133 | 0.244 | 0.593 | 0.129 | 0.241 | 0.519 |
| Tied Diag-Cov | 0.159 | 0.293 | 0.593 | 0.135 | 0.251 | 0.534 |
| Raw features – PCA m = 5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Full-Cov | 0.151 | 0.249 | 0.632 | 0.149 | 0.255 | 0.626 |
| Diag-Cov | 0.211 | 0.508 | 0.707 | 0.148 | 0.254 | 0.587 |
| Tied Full-Cov | 0.142 | 0.253 | 0.596 | 0.129 | 0.241 | 0.523 |
| Tied Diag-Cov | 0.163 | 0.310 | 0.591 | 0.136 | 0.251 | 0.539 |

All the considerations made for results on validation set remains valid, even results are slightly better on evaluation set. Tied Full covariance is the best choice for main application, PCA and Gaussianization don’t improve the results.

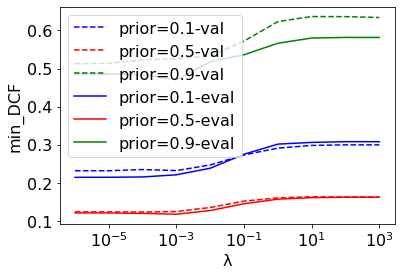
## Linear Logistic Regression

First we can verify if the choice of the optimal value for hyper parameter is still valid for evaluation set.2



Linear Logistic Regression (πT=0.5) : Min DCF with respect to ––

Raw features – no PCA



Linear Logistic Regression (πT=0.5) : Min DCF with respect to –– Gaussianized features – no PCA

The curves computed on validation and evaluation set are very close, so the choice actually provides close to optimal results also on evaluation set.

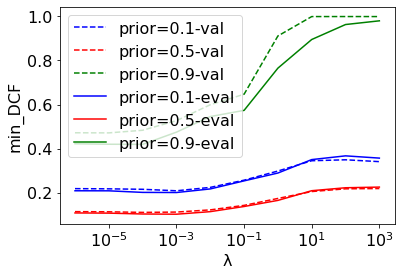
Min DCF results for Linear Logistic Regression trained on the whole training set

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.107 | 0.198 | 0.542 | 0.122 | 0.215 | 0.485 |
| LogReg(λ=10-6, πT=0.1 ) | 0.110 | 0.199 | 0.530 | 0.125 | 0.210 | 0.503 |
| LogReg(λ=10-6, πT=0.9 ) | 0.115 | 0.203 | 0.509 | 0.123 | 0.212 | 0.484 |
| Raw features – PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.106 | 0.201 | 0.542 | 0.119 | 0.216 | 0.481 |
| LogReg(λ=10-6, πT=0.1 ) | 0.110 | 0.199 | 0.534 | 0.121 | 0.210 | 0.485 |
| LogReg(λ=10-6, πT=0.9 ) | 0.113 | 0.202 | 0.532 | 0.121 | 0.216 | 0.469 |
| Raw features – PCA m= 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.118 | 0.222 | 0.567 | 0.117 | 0.230 | 0.474 |
| LogReg(λ=10-6, πT=0.1 ) | 0.118 | 0.222 | 0.560 | 0.126 | 0.217 | 0.500 |
| LogReg(λ=10-6, πT=0.9 ) | 0.129 | 0.227 | 0.524 | 0.119 | 0.223 | 0.471 |
| Raw features – PCA m = 5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| LogReg(λ=10-6, πT=0.5 ) | 0.128 | 0.235 | 0.545 | 0.120 | 0.228 | 0.488 |
| LogReg(λ=10-6, πT=0.1 ) | 0.126 | 0.231 | 0.566 | 0.126 | 0.217 | 0.498 |
| LogReg(λ=10-6, πT=0.9 ) | 0.137 | 0.243 | 0.539 | 0.123 | 0.231 | 0.487 |

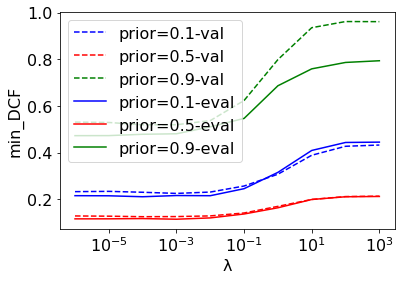
Again results on evaluation set are very close to the ones obtained before on validation set.

## Quadratic Logistic regression

As for Linear Logistic Regression we first check if the chosen is still close to optimal for evaluation set.



Quadratic Logistic Regression (πT=0.5) : Min DCF with respect to ––Raw features – no PCA



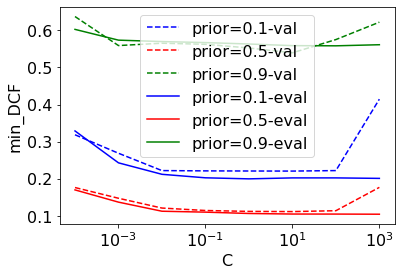
Quadratic Logistic Regression (πT=0.5) : Min DCF with respect to ––Gaussianized features – no PCA

It can be observed that trend of evaluation and validation curves are the same. is confirmed to be the best value.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic Log Reg  (λ=10-6, πT=0.5 ) | 0.109 | 0.209 | 0.422 | 0.117 | 0.216 | 0.473 |
| Quadratic Log Reg  (λ=10-6, πT=0.1 ) | 0.100 | 0.205 | 0.455 | 0.117 | 0.204 | 0.466 |
| Quadratic Log Reg  (λ=10-6, πT=0.9 ) | 0.116 | 0.222 | 0.437 | 0.120 | 0.239 | 0.477 |
| Raw features –PCA m = 7 | | | | Gaussianized features – PCA m = 7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic Log Reg  (λ=10-6, πT=0.5 ) | 0.107 | 0.208 | 0.440 | 0.116 | 0.213 | 0.504 |
| Quadratic Log Reg  (λ=10-6, πT=0.1 ) | 0.102 | 0.206 | 0.472 | 0.117 | 0.202 | 0.503 |
| Quadratic Log Reg  (λ=10-6, πT=0.9 ) | 0.112 | 0.215 | 0.441 | 0.119 | 0.230 | 0.478 |
| Raw features –PCA m= 6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic Log Reg  (λ=10-6, πT=0.5 ) | 0.106 | 0.207 | 0.472 | 0.119 | 0.222 | 0.490 |
| Quadratic Log Reg  (λ=10-6, πT=0.1 ) | 0.099 | 0.212 | 0.456 | 0.118 | 0.215 | 0.487 |
| Quadratic Log Reg  (λ=10-6, πT=0.9 ) | 0.111 | 0.219 | 0.424 | 0.122 | 0.238 | 0.486 |
| Raw features –PCA m = 5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic Log Reg  (λ=10-6, πT=0.5 ) | 0.113 | 0.225 | 0.468 | 0.116 | 0.226 | 0.495 |
| Quadratic Log Reg  (λ=10-6, πT=0.1 ) | 0.112 | 0.220 | 0.490 | 0.118 | 0.213 | 0.524 |
| Quadratic Log Reg  (λ=10-6, πT=0.9 ) | 0.118 | 0.239 | 0.461 | 0.120 | 0.240 | 0.492 |

Also Quadratic Logistic Regression results on evaluation set follow the same trend of results on validation set. The best option is confirmed to be Quadratic Log Reg with λ=10-6, πT=0.1 . The only difference is that on validation set the lowest value of min DCF for main application was obtained without PCA; while here for evaluation set is obtained with m = 6. Though there are only very small differences between results on data with and without PCA.

## Linear SVM



Linear SVM (πT=0.5) : Min DCF with respect to ––Raw features – no PCA

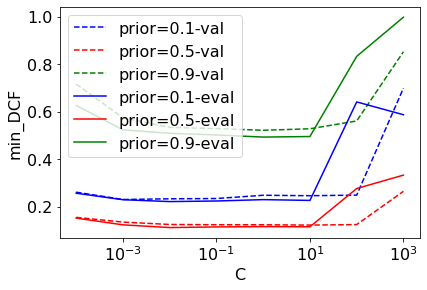
s

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.5 ) | 0.107 | 0.200 | 0.563 | 0.115 | 0.227 | 0.482 |
| Linear SVM(C=1, πT =0.1 ) | 0.113 | 0.201 | 0.554 | 0.125 | 0.208 | 0.500 |
| Linear SVM(C=1, πT =0.9 ) | 0.116 | 0.203 | 0.504 | 0.128 | 0.216 | 0.467 |
| Linear SVM(C=1, no-rebalancing) | 0.114 | 0.202 | 0.551 | 0.125 | 0.208 | 0.500 |
| Raw features –PCA m=7 | | | | Gaussianized features – PCA m=7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.5 ) | 0.106 | 0.203 | 0.572 | 0.112 | 0.230 | 0.463 |
| Linear SVM(C=1, πT =0.1 ) | 0.112 | 0.201 | 0.551 | 0.125 | 0.208 | 0.491 |
| Linear SVM(C=1, πT =0.9 ) | 0.120 | 0.203 | 0.508 | 0.126 | 0.207 | 0.468 |
| Linear SVM(C=1, no-rebalancing) | 0.114 | 0.201 | 0.550 | 0.125 | 0.207 | 0.489 |
| Raw features – PCA m=6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.1 ) | 0.120 | 0.224 | 0.588 | 0.127 | 0.219 | 0.492 |
| Raw features –PCA m=5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Linear SVM(C=1, πT =0.1 ) | 0.124 | 0.231 | 0.603 | 0.125 | 0.219 | 0.496 |

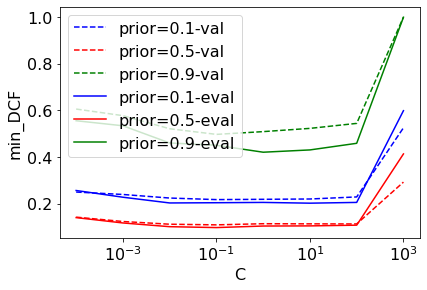
Results on evaluation set for Linear SVM confirms what we observed on validation set. Values of min DCF are even slightly lower for the evaluation set.

## Quadratic SVM

First we plot min DCF with respect to C to check if the selected value of C on validation set is actually close to optimal.



Quadratic SVM (): Min\_DCF with respect to C - Gaussianized features – no PCA



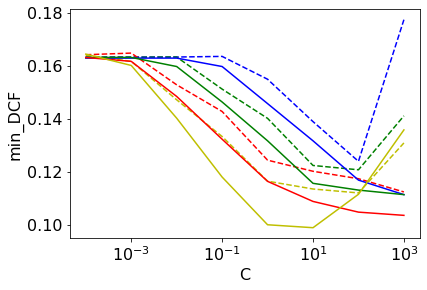
Quadratic SVM (): Min\_DCF with respect to C - raw features – no PCA

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.098 | 0.205 | 0.451 | 0.114 | 0.223 | 0.486 |
| Quadratic SVM(C=0.1, πT =0.1 ) | 0.103 | 0.201 | 0.516 | 0.125 | 0.208 | 0.549 |
| Quadratic SVM(C=0.1, πT =0.9 ) | 0.123 | 0.259 | 0.474 | 0.144 | 0.296 | 0.491 |
| Quadratic SVM(C=0.1, no-rebalancing) | 0.102 | 0.204 | 0.521 | 0.125 | 0.209 | 0.540 |
| Raw features –PCA m=7 | | | | Gaussianized features – PCA m=7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.097 | 0.204 | 0.534 | 0.112 | 0.225 | 0.494 |
| Quadratic SVM(C=0.1, πT =0.1 ) | 0.104 | 0.205 | 0.535 | 0.125 | 0.206 | 0.548 |
| Quadratic SVM(C=0.1, πT =0.9 ) | 0.127 | 0.242 | 0.472 | 0146 | 0.270 | 0.489 |
| Quadratic SVM(C=0.1, no-rebalancing) | 0.104 | 0.204 | 0.532 | 0.125 | 0.209 | 0.549 |
| Raw features – PCA m=6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.097 | 0.207 | 0.455 | 0.112 | 0.229 | 0.494 |
| Raw features –PCA m=5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quadratic SVM(C=0.1, πT =0.5 ) | 0.108 | 0.211 | 0.470 | 0.114 | 0.239 | 0.519 |

We can observe that Quadratic SVM with C=0.1, πT =0.5 is confirmed to be the best option. For main application, on evaluation set the lowest min DCF is obtained training on raw features with m=7, while on validation set the lowest value was obtained with PCA m=5. However the differences are very small among results with different value of m.

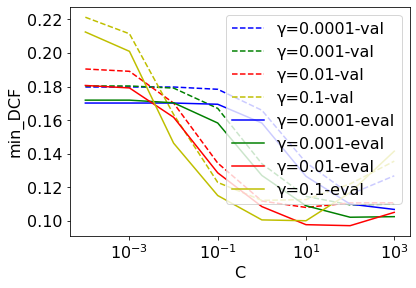
## RBF SVM

d



RBF SVM (πT=0.5) : Min DCF with respect to and γ ––Gaussianized features – no PCA

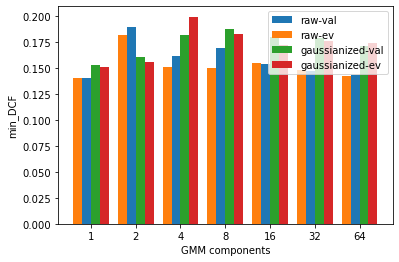
RBF SVM (πT=0.5) : Min DCF with respect to and γ ––Raw features – no PCA



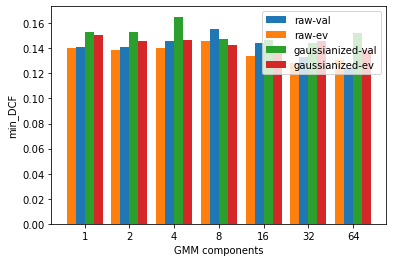
d

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Raw features – no PCA | | | |  | Gaussianized features – no PCA | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10,γ=0.01, πT=0.5 ) | 0.098 | 0.200 | 0.478 | 0.109 | 0.218 | 0.469 |
| RBF SVM(C=10, γ=0.01, πT=0.1 ) | 0.105 | 0.201 | 0.452 | 0.123 | 0.219 | 0.567 |
| RBF SVM(C=10, γ=0.01, πT=0.9 ) | 0.131 | 0.298 | 0.458 | 0.129 | 0.307 | 0.468 |
| RBF SVM(C=10, γ=0.01, no-rebalancing) | 0.105 | 0.201 | 0.458 | 0.123 | 0.220 | 0.579 |
| Raw features –PCA m=7 | | | | Gaussianized features – PCA m=7 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10, γ=0.01, πT=0.5 ) | 0.099 | 0.202 | 0.486 | 0.108 | 0.218 | 0.465 |
| RBF SVM(C=10, γ=0.01, πT=0.1 ) | 0.106 | 0.201 | 0.480 | 0.123 | 0.219 | 0.566 |
| RBF SVM(C=10, γ=0.01, πT=0.9 ) | 0.125 | 0.275 | 0.480 | 0.128 | 0.302 | 0.463 |
| RBF SVM(C=10, γ=0.01, no-rebalancing) | 0.107 | 0.202 | 0.478 | 0.123 | 0.219 | 0.579 |
| Raw features – PCA m=6 | | | | Gaussianized features – PCA m = 6 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10, γ=0.01, πT=0.5 ) | 0.102 | 0.200 | 0.443 | 0.108 | 0.219 | 0.477 |
| Raw features –PCA m=5 | | | | Gaussianized features – PCA m = 5 | | |
|  | **π= 0.5** | **π= 0.1** | **π=0.9** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM(C=10, γ=0.01, πT=0.5 ) | 0.105 | 0.209 | 0.469 | 0.109 | 0.219 | 0.515 |

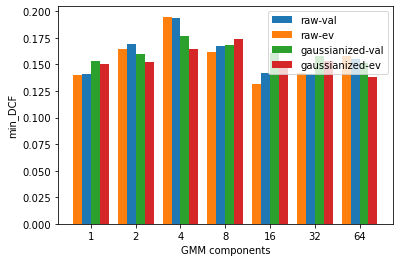
## GMM



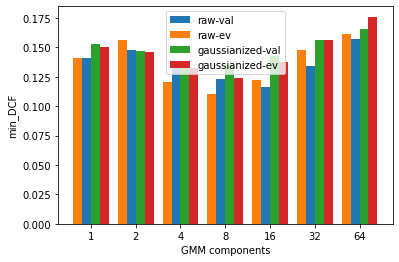
Diag-tied



Full-tied



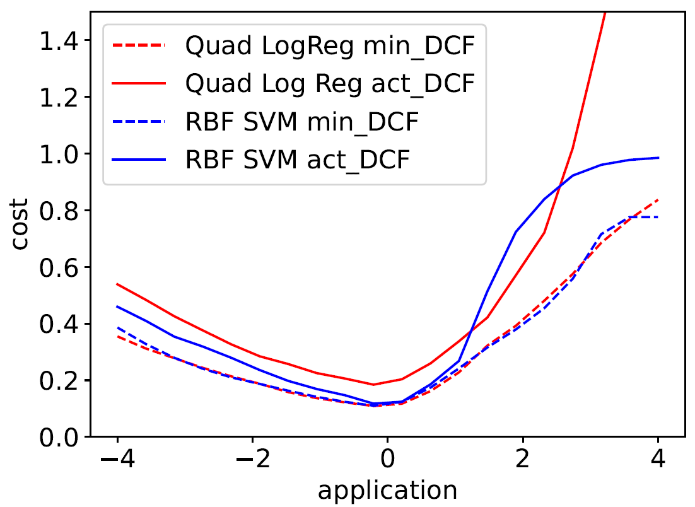
Diag cov



Full cov

## Score Calibration

Up to here we have considered results on evaluation set only in term of min DCF. We need to assess whether the scores of the selected models on evaluation set are calibrated or not. We can check Bayer error plot for actual and minimum DCF:



Min DCF vs Act DCF on evaluation set

The trends of Bayes error plot is the same obtained with validation set. Scores need to be recalibrated. We previously considered 2 approaches to score calibration: score transformation and threshold optimization. On validation set score transformation provided better results. Now we consider again both 2 approach on evaluation set to verify that what we obtained on validation set is still valid.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **π** | **π= 0.5** | | | **π= 0.1** | | | **π=0.9** | | |
| Cost | Min DCF | Act DCF  **Theoretical threshold** | Act DCF  **t\*** | Min DCF | Act DCF  **Theoretical threshold** | Act DCF **t\*** | Min DCF | Act DCF  **Theoretical threshold** | Act DCF **t\*** |
| Quad Log Reg | 0.097 | 0.181 | 0.103 | 0.212 | 0.401 | 0.230 | 0.478 | 0.663 | 0.504 |
| RBF SVM | 0.099 | 0.108 | 0.100 | 0.182 | 0.429 | 0.183 | 0.447 | 0.898 | 0.499 |

|  |  |  |  |
| --- | --- | --- | --- |
| **π** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| Quad Log Reg(π=0.5) | 0.103 | 0.235 | 0.492 |
| Quad Log Reg (π=0.1) | 0.106 | 0.228 | 0.492 |
| Quad Log Reg (π=0.9) | 0.100 | 0.225 | 0.505 |
| Quad Log Reg  No calibration | 0.181 | 0.401 | 0.663 |
| Quad Log Reg | 0.097 | 0.212 | 0.478 |

|  |  |  |  |
| --- | --- | --- | --- |
| **π** | **π= 0.5** | **π= 0.1** | **π=0.9** |
| RBF SVM (π=0.5) | 0.103 | 0.197 | 0.490 |
| RBF SVM (π=0.1) | 0.103 | 0.204 | 0.465 |
| RBF SVM (π=0.9) | 0.103 | 0.197 | 0.494 |
| RBF SVM  No calibration | 0.108 | 0.429 | 0.898 |
| RBF SVM | 0.099 | 0.182 | 0.447 |

## Fusion

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **π** | **π= 0.5** | | **π= 0.1** | | **π=0.9** | |
| Cost | Min DCF | Act DCF | Min DCF | Act DCF | Min DCF | Act DCF |
| Quadratic Log Reg (λ=10-6, πT=0.1 ) | 0.097 | 0.103 | 0.212 | 0.235 | 0.478 | 0.492 |
| RBF SVM(C=10,γ=0.01, πT=0.5 ) | 0.099 | 0.103 | 0.182 | 0.197 | 0.447 | 0.490 |
| Fusion | 0.097 | ? | 0.190 | ? | 0.422 | 0.? |

# References

[1] https://archive.ics.uci.edu/ml/datasets/HTRU2

[2] M. J. Keith et al., 'The High Time Resolution Universe Pulsar Survey - I. System Configuration and Initial Discoveries',2010, Monthly Notices of the Royal Astronomical Society, vol. 409, pp. 619-627. DOI: 10.1111/j.1365-2966.2010.17325.x

[3] R. J. Lyon, 'Why Are Pulsars Hard To Find?', PhD Thesis, University of Manchester, 2016.

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.norm.html?highlight=norm%20ppf

[5] <https://numpy.org/doc/stable/reference/generated/numpy.corrcoef.html>

[6]<https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fmin_l_bfgs_b.html?highlight=scipy%20optimize%20fmin#scipy.optimize.fmin_l_bfgs_b>

[7] scipy.optimize.fmin\_l\_bfgs\_b