s317661 - Tcaciuc Claudiu Constantin

Machine Learning & Pattern Recognition

Project - fingerprint spoofing detection

GITHUB LINKS:

- All course laboratories
- Project repositories

Project details

The project task consists of a binary classification. The goal is to perform fingerprint spoofing detection to identify genuine vs counterfeit fingerprint images. The dataset consists of labeled samples corresponding to the genuine (True, label 1) class and the fake (False, label 0) class. The samples are computed by a feature extractor that summarizes high-level characteristics of a fingerprint image. The data is 6-dimensional.

Dataset Analysis

We start by analyzing the dataset with some plots, the first thing we do is plot the histogram graph for each feature.

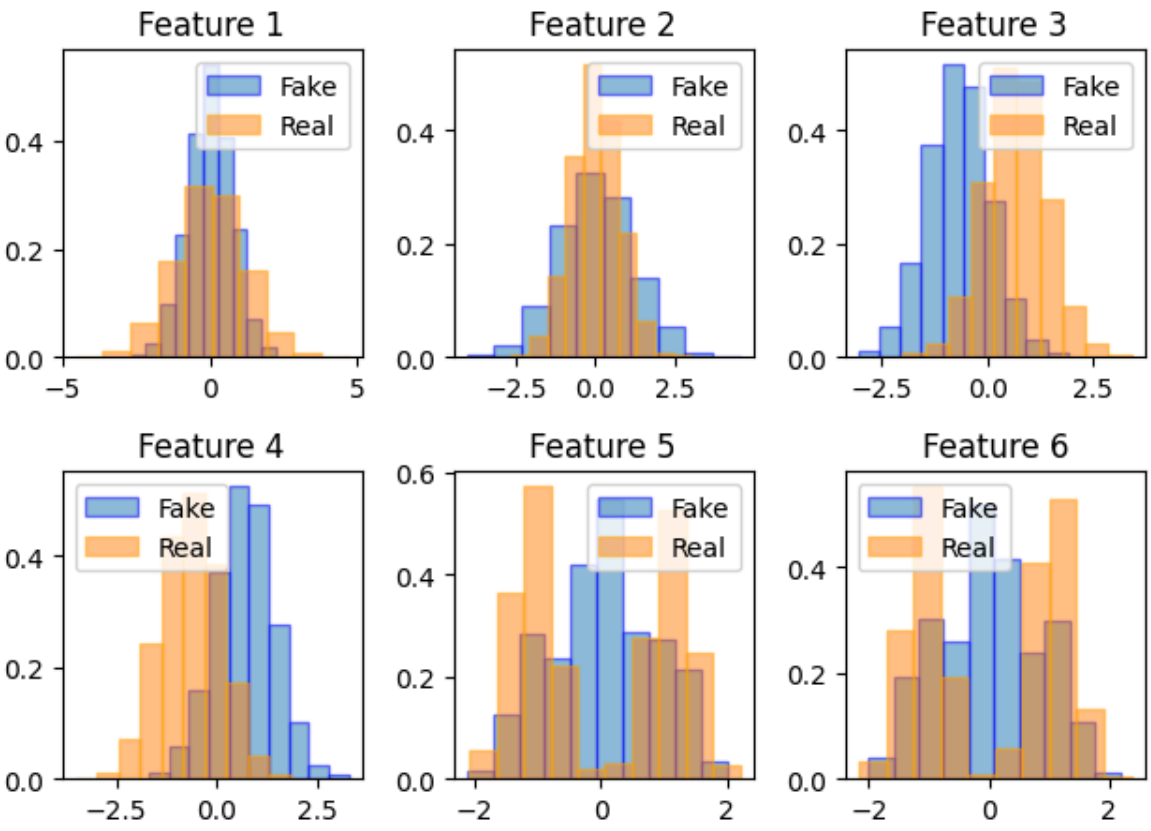

fig 1: Histogram plot of the various features present inside the dataset.

As we can see from the various plots we can identify some graphs that resemble a gaussian plot, such as feature 1, 2, 3, 4.

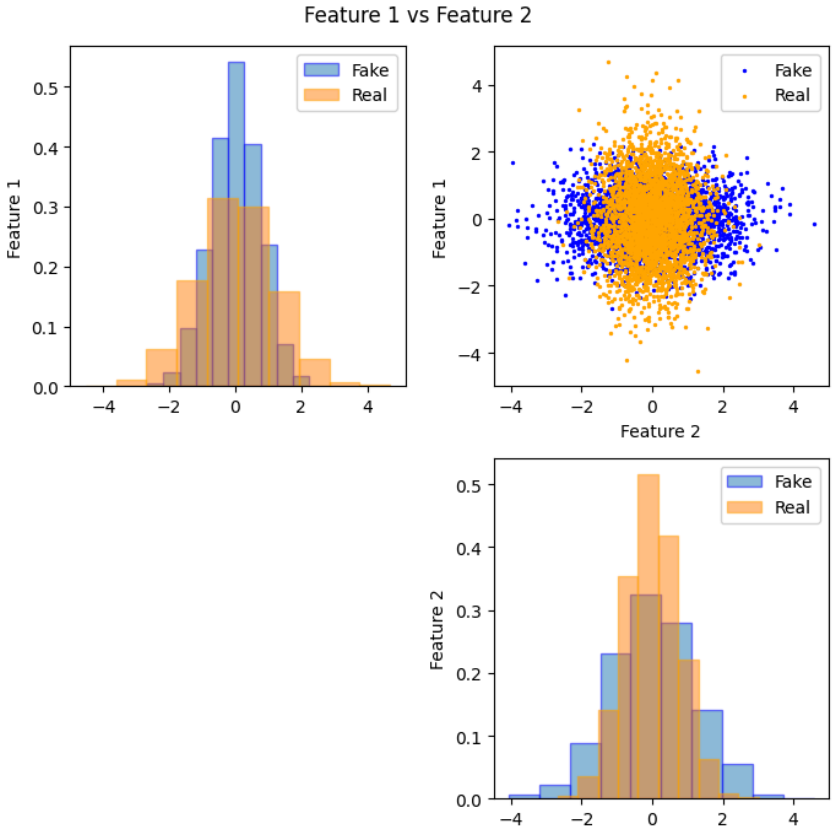

fig 2: Histogram plot and scatter plot of features 1 and 2

Feature 1 analysis:

- Histogram analysis:
    - Overlap: There is significant overlap between the distributions of real and fake fingerprints, indicating that this feature alone might not strongly distinguish between the two classes.
    - Peaks: Each class displays a single prominent peak. The peaks are nearly aligned, centered slightly right of zero. This central alignment suggests that the feature values for both classes are similarly distributed in terms of central tendency.
    - Shape: Both histograms are somewhat bell-shaped, suggesting a roughly normal distribution.

- Statistical Analysis:
    - Mean: Both classes seem to have a similar mean, centered around zero.
    - Variance: The variance of the real class appears slightly lower than that of the fake class, as indicated by the slightly tighter clustering around the mean for real fingerprints.

Feature 2 analysis:

- Histogram analysis
    - Overlap: Similar to Feature 1, there is considerable overlap in the histograms of Feature 2 for real and fake fingerprints.
    - Peaks: Both classes have a single peak which is closely aligned, centered slightly right of zero.
    - Shape: The histograms are less bell-shaped compared to Feature 1, especially for fake fingerprints which show a flatter spread indicating a higher variance.
- Statistical Analysis
    - Mean: Like Feature 1, the means of both classes in Feature 2 are similar.
    - Variance: The variance in Feature 2 for the fake class is noticeably higher than for the real class, as the fake class exhibits a broader spread of values.

Pairwise Scatter plot:

- Distribution: The scatter plot of Feature 1 vs Feature 2 shows a dense cloud with no clear separation between real and fake classes, indicating complex similarities in the way these features are represented across both classes.
- Correlation: The plot suggests a low or negligible correlation between Feature 1 and Feature 2 as there isn’t a clear linear or monotonic pattern visible. Both features vary widely without a distinct trend relative to each other.

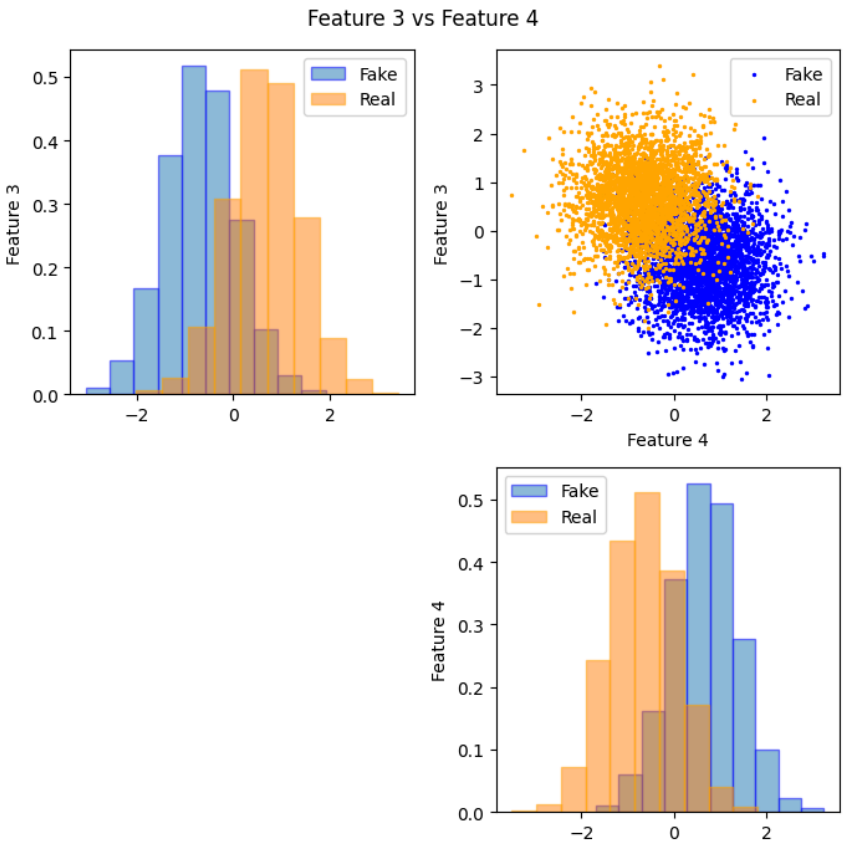

fig 3: Histogram plot and scatter plot of features 3 and 4

Feature 3 analysis:

- Histogram analysis:
    - Overlap: The distributions of real and fake classes for Feature 3 overlap significantly, but not as uniformly as seen in Features 1 and 2. The peaks are staggered, with the real class peaking slightly left of zero and the fake class peaking right of zero.
    - Peaks: Both classes exhibit one primary peak each. The fake class shows a more defined peak, whereas the real class has a broader and flatter distribution.
    - Shape: The histogram for the real class approximates a skewed normal distribution, leaning towards the left. The fake class has a broader, less symmetric distribution, suggesting higher variability.
- Statistical Analysis:

- Mean and Variance: Given the shifting peaks, the mean of the fake class may be slightly lower than that of the real class. The variance of the real class seems larger due to the wider spread of values.

Feature 4 analysis:

- Histogram analysis
  - Overlap: There is a significant overlap in Feature 4 as well, although the fake class shows a higher peak near the zero while the real class is more evenly distributed across a wider range of values.
  - Peaks: The real class has a broader peak distribution, while the fake class has a more pronounced peak near one, suggesting a key difference in their behavior.
  - Shape: The distribution shapes differ, with the real class showing a more normal-like distribution and the fake class exhibiting a peakier, narrower distribution centered around zero.
- Statistical Analysis
  - Mean and Variance: The real class appears to have a slightly lower mean and a higher variance, reflecting the broader spread in its histogram.

Pairwise Scatter plot:

- Distribution: The scatter plot displays a considerable mixing of the real and fake classes without a clear boundary, showing that these two features also have complex interdependencies.
- Correlation: The plot shows a slight trend where both features increase together, indicating a possible mild positive correlation between them.

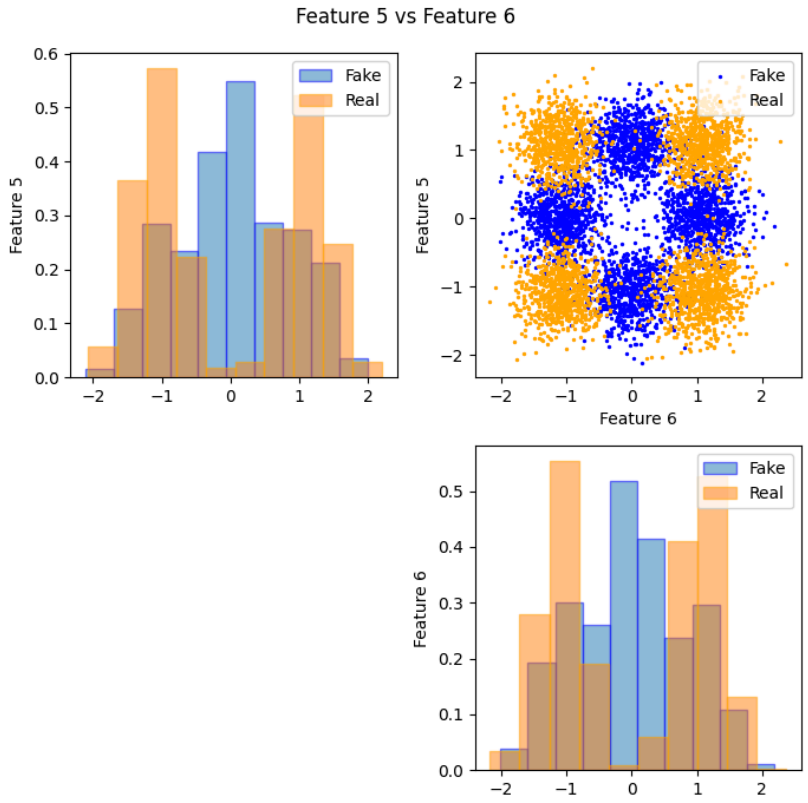

fig 4: Histogram plot and scatter plot of features 5 and 6

Feature 5 analysis:

- Histogram analysis:
    - Overlap: The overlap between real and fake classes in Feature 5 is noticeable but shows some distinct differences in distribution, particularly around the center of the range.
    - Peaks: The real class has a double-peaked distribution, with one peak near 1 and another around -1. The fake class has a single peak near zero.
    - Shape: The real class distribution is bimodal, while the fake class distribution is unimodal and more concentrated around zero.
- Statistical Analysis:
    - Mean and Variance: The real class, even though bimodal, could have a mean closer to zero due to the double peaks. The variance of the real

class may be higher due to the broader spread of values. The fake class could also have similar variance but centered around zero.

Feature 6 analysis:

- Histogram analysis
    - Overlap: Significant overlap is evident, with both classes showing substantial presence across the entire range, though the fake class tends to cluster more towards the center.
    - Peaks: Similar to Feature 5, the real class has a double-peaked distribution, while the fake class has a single peak.
    - Shape: The real class distribution is bimodal, while the fake class distribution is unimodal and more concentrated around zero.
- Statistical Analysis
    - Mean and Variance: The mean of the real class may be slightly higher than that of the fake class, as the real class distribution is more spread out. The variance of the real class could be higher due to the broader distribution of values.

Pairwise Scatter plot:

- Distribution: The scatter plot shows a good mix but not a complete overlap, indicating that these two features combined might offer better class separation compared to previous pairs. The spread suggests some degree of correlation but not strongly defined.
- Correlation: There appears to be a slight positive correlation between Features 5 and 6, as indicated by the general trend of points stretching from the bottom-left towards the top-right.

PCA & LDA

Principal Component Analysis (PCA) allows reducing the dimensionality of a dataset by projecting the data over the principal components. These PCA are orthogonal to each other, meaning they are independent and do not correlate with

each other.

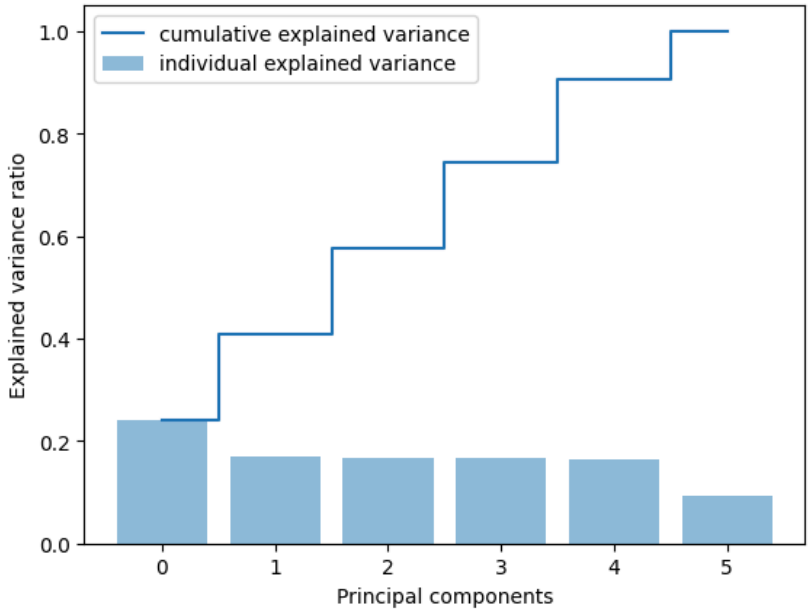

fig 5: explained variance ratio for principal components derived from PCA on the dataset.

Individual Explained Variance: The height of the bars represents the variance explained by each principal component. The first few components capture more variance than the later ones, which is typical in PCA. Each bar's height indicates the proportion of the dataset's total variability captured by that specific component.

Cumulative Explained Variance: The line shows the cumulative sum of the explained variances. This is crucial for determining how many principal components are needed to capture a substantial amount of information from the data.

As we can see the first principal component captures the most variance, with each subsequent component capturing progressively less, by the time we reach the fifth component we capture more than 80% of the total variance. This suggests that the first 5 components may be enough to perform classification on our dataset.

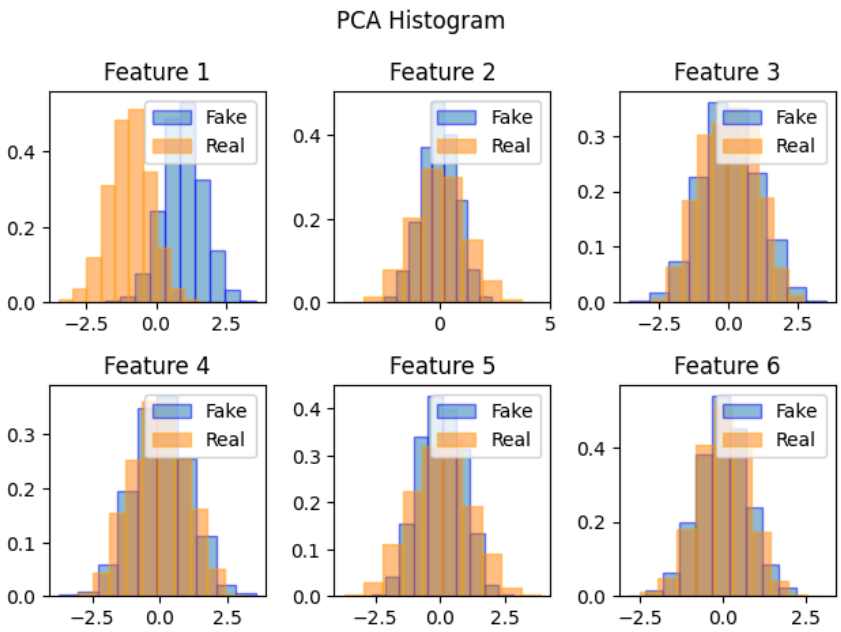

fig 6: Histogram plot of the various features present inside the dataset after applying PCA

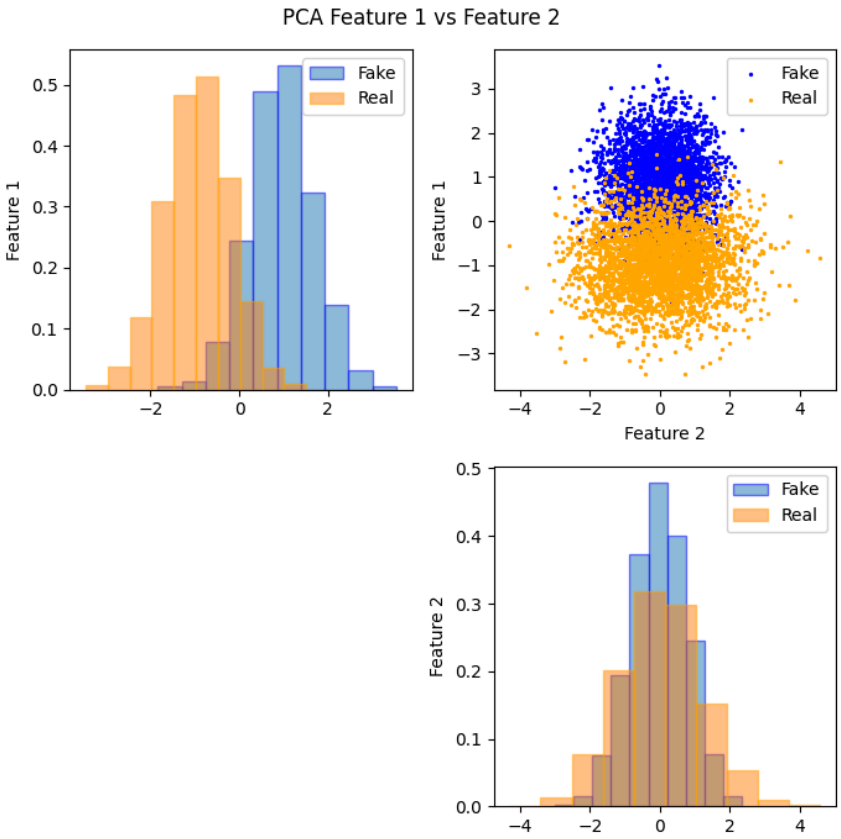

fig 7:Histogram plot and scatter plot of paired features after applying PCA.

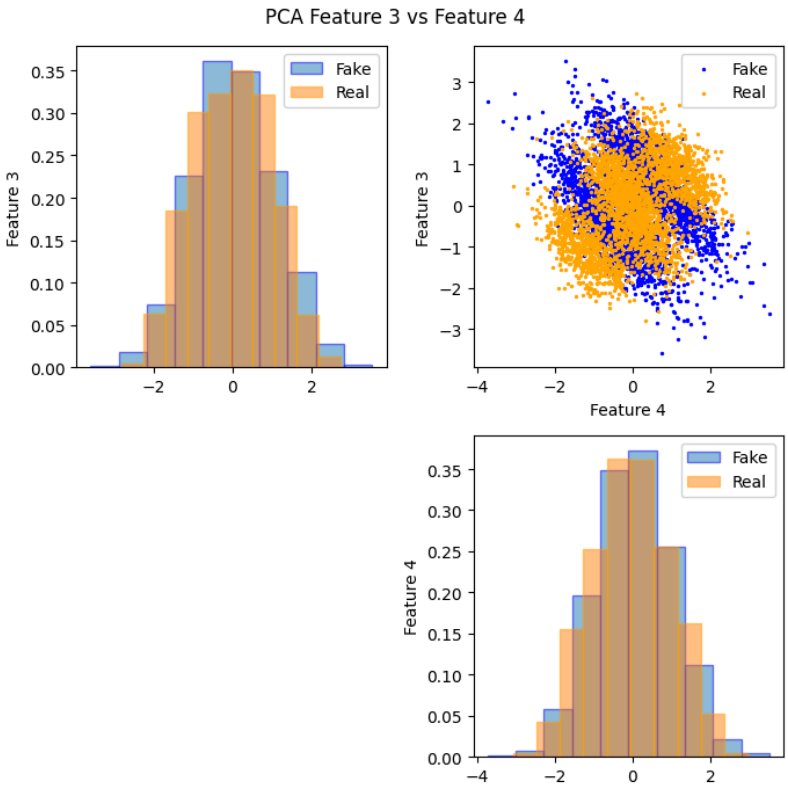

fig 8: Histogram plot and scatter plot of paired features after applying PCA.

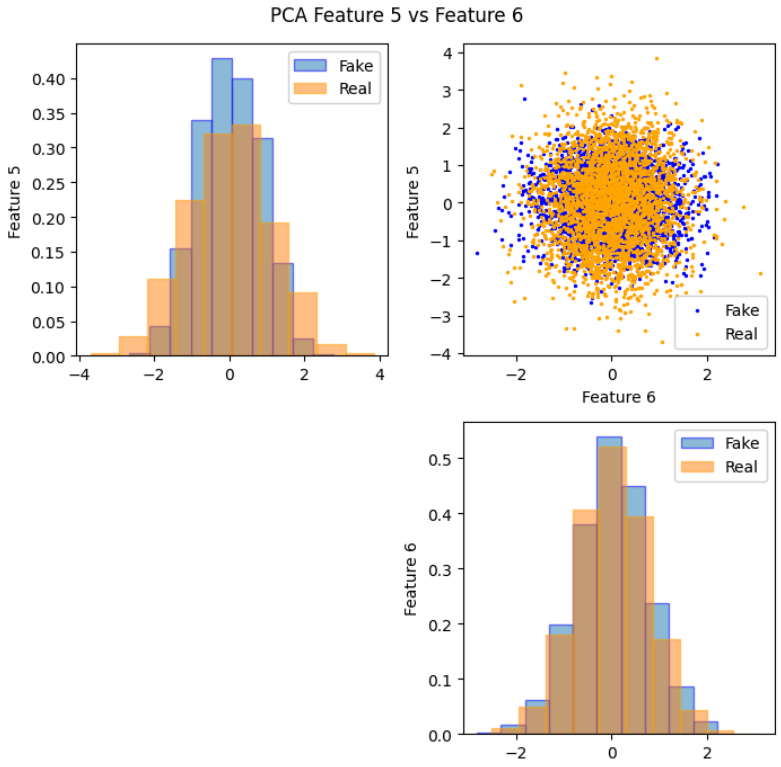

fig 9: Histogram plot and scatter plot of paired features after applying PCA.

Feature 1:

- Overlap: There is significant overlap between the real and fake classes, although there are distinct peaks for each class. The fake class peaks around 1, while the real class peaks just above -1. This suggests that PCA Feature 1 does offer some discriminative information but is not fully separable.
- Peaks: The existence of distinct peaks for each class is promising as it indicates that PCA Feature 1 captures aspects that are somewhat characteristic of each class.
- Shape: Both classes exhibit fairly normal distributions.

Feature 2:

- Overlap: Feature 2 also shows considerable overlap. This suggests that Feature 2 might be less powerful in discriminating between the classes on its own.
- Peaks: The histograms for both classes are centered around zero but the real class spreads in a more widely manner across the value range.
- Shape: The histograms for both classes are quite similar in shape, indicating that PCA Feature 2 may not capture as distinct differences between the classes as PCA Feature 1.

Feature 3:

- Overlap: The histograms for PCA Feature 3 exhibit a significant overlap between the real and fake classes. Both classes follow a similar distribution with a central peak around zero.
- Peaks: Both classes display single, central peaks which are nearly aligned. This similarity in peaks suggests that Feature 3 alone may not provide sufficient discrimination power between the classes.
- Shape: The distribution is roughly normal for both classes, indicating Feature 3 captures general, rather than distinctive, variations within the data.

Feature 4:

- Overlap: Feature 4 also shows substantial overlap between real and fake classes. The peaks are less distinct compared to Feature 3.
- Peaks: The distributions are less sharply peaked and more evened out, which could indicate that Feature 4 is capturing a broader range of variations within each class.
- Shape: The distribution is roughly normal for both classes, indicating Feature 4 captures general, rather than distinctive, variations within the data.

Feature 5:

- Overlap: There is considerable overlap in the histograms for PCA Feature 5, although the real class has a slight shift to the right compared to the fake class. This suggests a small difference in the central tendency of the data for each class.
- Peaks: Both classes show central peaks, with the real class peaking around zero and the fake class showing a peak slightly left of zero.
- Shape: The distributions for both classes are nearly normal.

Feature 6:

- Overlap: Significant overlap is observed in PCA Feature 6 as well, with both classes exhibiting distributions that are almost identical in terms of location and spread.
- Peaks: Both classes peak around zero, making this feature less useful for discrimination on its own.
- Shape: The shape of the distribution for both classes is symmetric and normally distributed, which indicates that while Feature 6 captures variance, it may not be discriminative between the classes.

Linear Discriminant Analysis (LDA) is a dimensionality reduction technique primarily used in the field of pattern recognition and machine learning. Unlike PCA, which focuses on maximizing the variance in the dataset, LDA aims to find a lower dimensional space that maximizes the separation between different classes or

categories in the data.

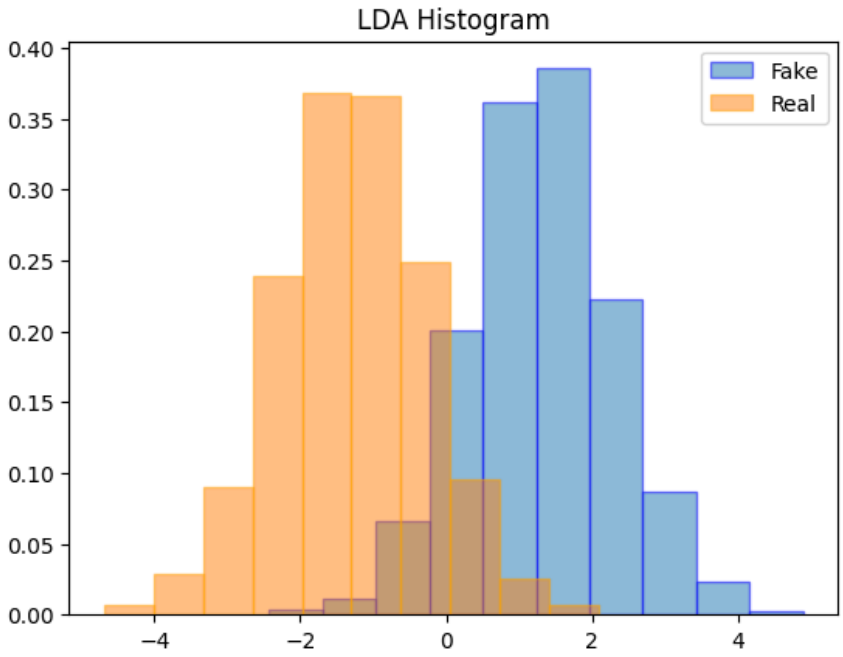

fig 10: Histogram plot of the dataset after applying LDA (since the classes are only two, we have only one feature).

As we can see from the plot the classes slightly overlap around zero, their distribution is fairly normal, with the real class having a more evened out peak and shifted to -2, and on the other hand the fake class has a single peak shifted to 2.

We can now try to apply LDA as a classifier, to do so we need to define a threshold to classify the prediction the LDA will make, to do so we can simply employ the mean of the projected class means computed on the training data.

```
LDA Accuracy: 90.70% (threshold: -0.02)
```

Now we change the threshold to see if we can find a better threshold.

THRESHOLD	ACCURACY
-10.00	50.40%
-8.00	50.40%
-6.00	50.40%
-3.99	50.50%
-1.99	61.00%
0.01	91.10%
2.01	61.60%
4.01	49.85%

fig 11: Table with threshold and corresponding accuracy for the LDA classifier

As we can see from the table we find a slightly better accuracy with a different threshold: `Best accuracy: 91.10% with threshold: 0.01`

We can now try to process the data with PCA then apply it to the LDA classifier to see what results we can achieve.

PCA FEATURES	ACCURACY
6	90.70%
5	90.70%
4	90.75%
3	90.75%
2	9.75%
1	90.65%

fig 12: Table with PCA feature and corresponding accuracy

As we can see from the table the best accuracy we can find is with pca 4 and 3 features, but if we used 2 for classification we would have had a very bad result. Also pca 6 results, as we expect, to be useless.

Multivariate Gaussian model

We are going now to try and fit uni-variate Gaussian models to the different features of the project

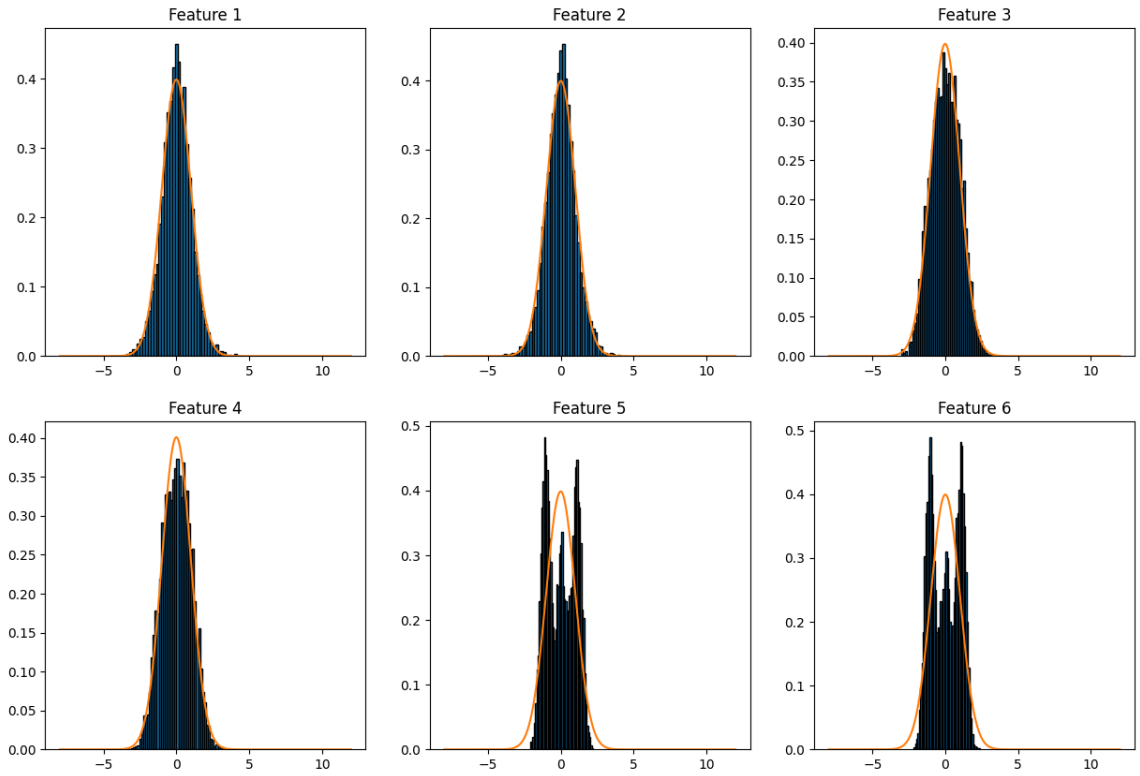

fig 13: Distribution density on top of the normalized histogram of the data

Looking at the various plots we can see that the visual comparison between the empirical distribution (histograms) and the theoretical Gaussian distribution (orange line) offers useful insights into how well a Gaussian model fits each feature.

Feature 1, 2, 3 and 4 demonstrate a very good fit with the Gaussian model. The peaks of the histogram align closely with the peak of the Gaussian density, and the spreads also match. This suggests that these features are well-modeled by a Gaussian distribution.

Feature 5 and 6 are the least well-fitted by a Gaussian model, as evidenced by the presence of multiple sharp peaks that don't align with the theoretical Gaussian, also the spread doesn't match. This could indicate that the data distribution for these features is not well structured for a Gaussian model.

We can now try applying the Gaussian model to our dataset to evaluate the performance of the model (we assume class prior probabilities equals 0.5, and we split the dataset in ⅔ training and ⅓ testing).

- LDA model accuracy: LDA Accuracy: `90.70%` (threshold: `-0.02`)
- MVG model accuracy: MVG Accuracy: `93.00%` (threshold: `0.00`)
- NB model accuracy: NB Accuracy: `92.80%` (threshold: `0.00`)
- TC model accuracy: TC Accuracy: `90.70%` (threshold: `0.00`)

What we can see is that the Tied Covariance model and the LDA achieve the same accuracy (although with different thresholds) and the best one is the normal Multivariate Gaussian Model.

We can now analyze the correlation between each feature, both overall and for each class separately.

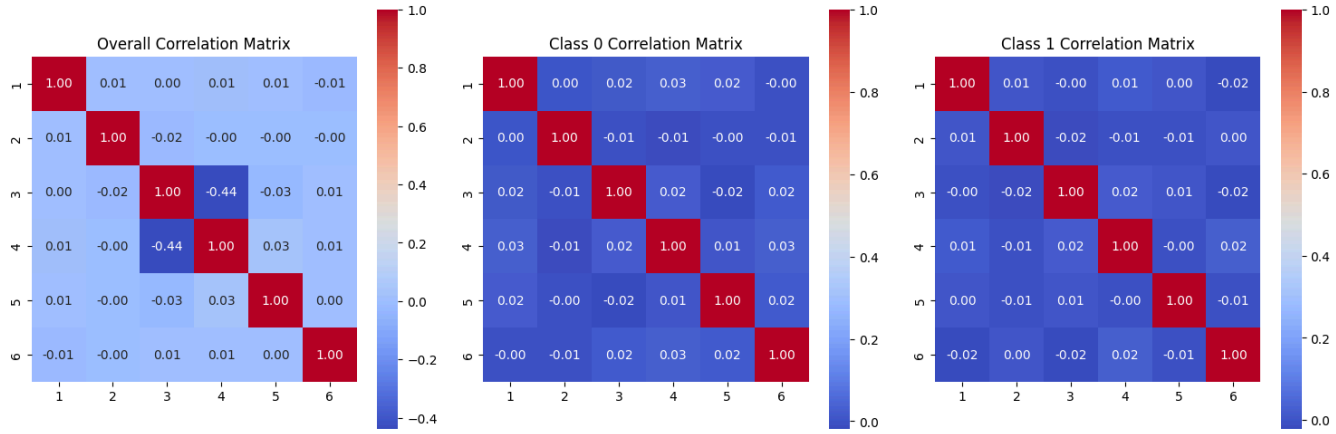

fig 14: Correlation matrix between each feature in the dataset.

The values on the diagonal represent the correlation of each feature with themself, which is always 1.

Most of the off diagonal values are very close to zero, indicating very little correlation between most pairs of features. This lack of correlation is generally good for machine learning models because it suggests that features provide unique information. There is only a slight inverse correlation between feature 3 and 4 (-0.44) indicating an inverse relationship between these two features.

We saw in fig 13 that the last two sets of features may negatively affect the accuracy of our Gaussian model. To assert this sentence we can try to fit the model only on the features 1 to 4.

- `MVG Accuracy: 92.05% (threshold: -0.00)`
- `NB Accuracy: 92.35% (threshold: -0.00)`
- `TC Accuracy: 90.50% (threshold: -0.00)`

As we can see from these results removing the last two features negatively affects all of the models accuracy. The biggest dropout if from the MVG model with drops from 93.00% to 92.05%

We can now analyze how each pair of features (1 and 2, 3 and 4) affects the performance of the Gaussian models.

Feature 1 and 2:

- MVG Accuracy: 63.50% (threshold: -0.00)
- NB Accuracy: 63.70% (threshold: -0.00)
- TC Accuracy: 50.55% (threshold: -0.00)

Feature 3 and 4:

- MVG Accuracy: 90.55% (threshold: -0.00)
- NB Accuracy: 90.55% (threshold: -0.00)
- TC Accuracy: 90.60% (threshold: -0.00)

As we can see using only the first two feature to classify the dataset is not a good choice, as we have seen from fig 2 from the scatter plot there is not a clear line of separation between the two classes (although here we are using only ⅔ of the dataset) but for feature 3 and 4 we can achieve a fairly good results. One thing we can see is that for feature 1 and 2 the Tied Covariance performance is the worst between the classifiers, but for feature 3 and 4 is the one performing better (slightly).

We can now analyze the effects of PCA as pre-processing.

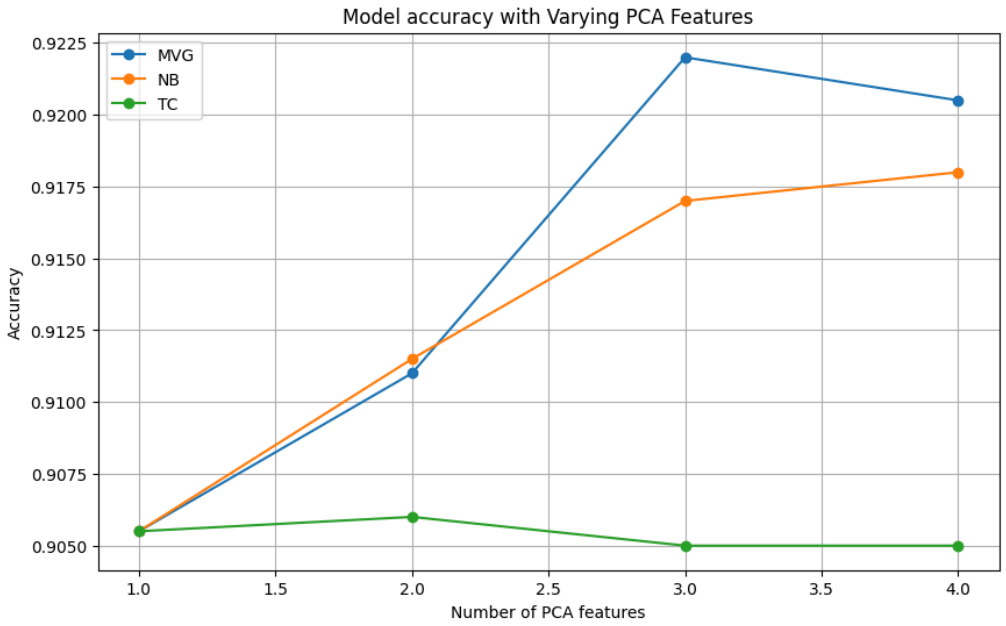

fig 15: Accuracy graph for the dataset with feature 1 to 4 with PCA applied as pre processing

- MVG - Best Accuracy: (3, 0.922)
- NB - Best Accuracy: (4, 0.918)
- TC - Best Accuracy: (2, 0.906)

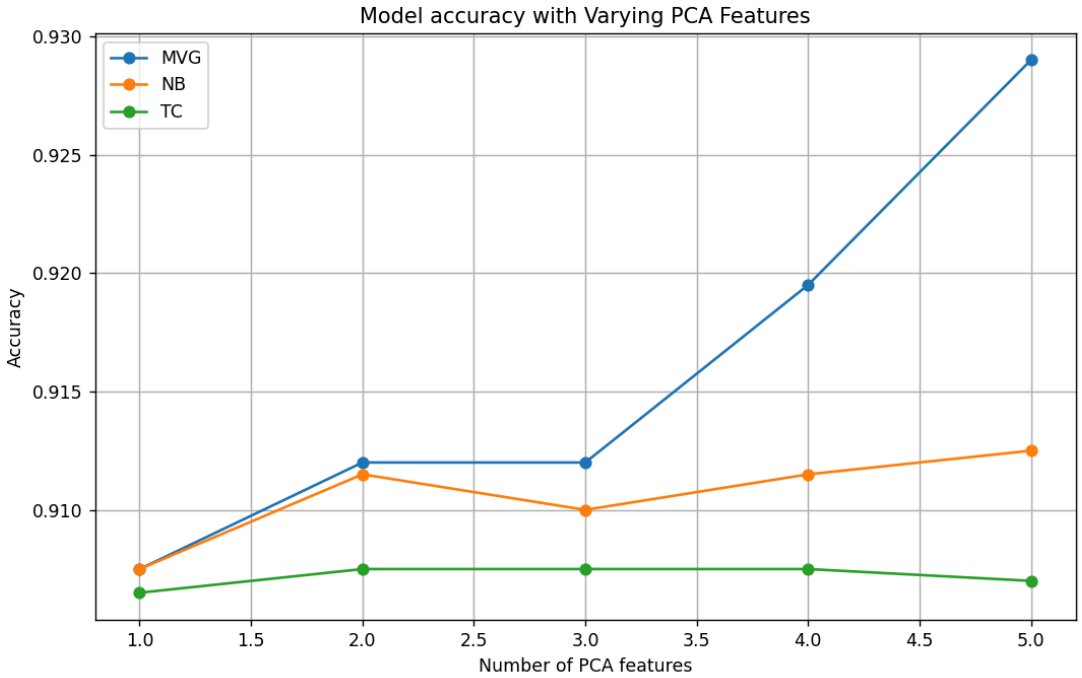

fig 16: Accuracy graph for the dataset with feature 1 to 5 with PCA applied as pre processing

- MVG - Best Accuracy: (5, 0.929)
- NB - Best Accuracy: (5, 0.9125)
- TC - Best Accuracy: (4, 0.9075)

As we can see the results with and without applying PCA are very similar, if we consider the best accuracy for different pca feature extraction. For the classic MVG we basically have the same accuracy, for the Naive Bayes we got a slightly worse accuracy, and for Tied covariance we get a slightly better accuracy.

Bayes decision evaluation

From now on we will change how we perform evaluation on our model. We will have the model output scores and then use this score to make a decision based on an application (pi, Cfn, Cfp). One key factor to this new evaluation method is the detection cost function (DCF) and the minimum detection cost function (min DCF).

The DCF is basically a metric used to measure the cost associated with the errors made by a binary classifier; this function helps to quantify the overall cost of errors made by the systems. The min DCF is the value of the DCF over a range of decision thresholds; this is useful because min DCF explores various thresholds to find the one that minimizes the overall cost according to the DCF formula.

To see the effect of the DCF on the Gaussian models we can test various applications (pi, Cfn, Cfp).

APPLICATION	MIN DCF	DCF NORM	ACCURACY
0.5, 1.0, 1.0	0.1302	0.1399	93.50%
0.9, 1.0, 1.0	0.3423	0.4001	89.70%
0.1, 1.0, 1.0	0.2629	0.3051	89.60%
0.5, 1.0, 9.0	0.1302	0.4001	93.50%
0.5, 9.0, 1.0	0.1302	0.3051	93.50%

table 1: min DCF, DCF norm for MVG model for various applications

APPLICATION	MIN DCF	DCF NORM	ACCURACY
0.5, 1.0, 1.0	0.1311	0.1439	93.45%
0.9, 1.0, 1.0	0.3510	0.3893	90.45%
0.1, 1.0, 1.0	0.2570	0.3022	89.90%
0.5, 1.0, 9.0	0.1311	0.3893	93.45%
0.5, 9.0, 1.0	0.1311	0.3022	93.45%

table 2: min DCF, DCF norm for NB model for various applications

APPLICATION	MIN DCF	DCF NORM	ACCURACY
0.5, 1.0, 1.0	0.1812	0.1860	90.95%
0.9, 1.0, 1.0	0.4421	0.4626	84.75%
0.1, 1.0, 1.0	0.3628	0.4061	86.60%
0.5, 1.0, 9.0	0.1812	0.4626	90.95%
0.5, 9.0, 1.0	0.1812	0.4061	90.95%

table 3: min DCF, DCF norm for TC model for various applications

One thing to note is that the prior probability passed to the application reflects the cost of miss-calibration. In our dataset we basically have a 50/50 split (2990 samples of class 0 and 3010 samples of class 1, without the splitting in training and test datasets) so when choosing prior = 0.5 we get a lower cost of errors compared to prior 0.1 or 0.9.

Another thing to note is that we get the biggest gap between min DCF and DCF when we set the cost of false positive or false negative different from 1.0.

From this table what we can take is that the best performing application is (0.5, 1.0, 1.0) for all the models in terms not only of accuracy but also minimizing the DCF.

What we can test now is the effect of PCA as pre-processing on our training dataset to see if applying it has some positive effects on the DCF, and we test it on application Cfp = 1.0 and Cfn = 1.0

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.2738	0.3041	86.30%	5
NB	0.3545	0.3930	83.45%	5
TC	0.3648	0.4051	83.25%	5

table 5: min DCF, DCF norm for Gaussian model with PCA = 5 and application (0.1, 1.0, 1.0)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.3012	0.3529	84.25%	4
NB	0.3614	0.3970	83.25%	4
TC	0.3610	0.4031	83.35%	4

table 6: min DCF, DCF norm for Gaussian model with PCA = 4 and application (0.1, 1.0, 1.0)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.3563	0.3879	83.30%	3
NB	0.3645	0.3950	83.35%	3
TC	0.3681	0.4082	83.50%	3

table 7: min DCF, DCF norm for Gaussian model with PCA = 3 and application (0.1, 1.0, 1.0)

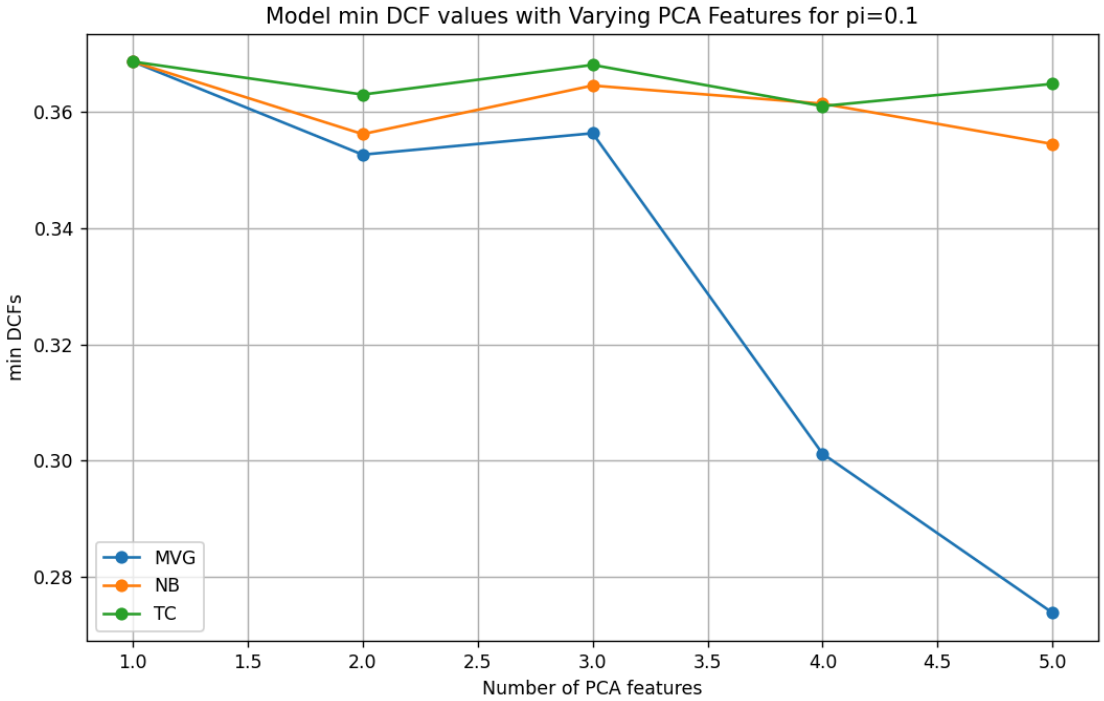

fig 17: min DCF values over different PCA feature and application (0.1, 1.0, 1.0)

- MVG - Pca, Accuracy, Best minDCF: (5, 0.863, 0.2738)
- NB - Pca, Accuracy, Best minDCF: (5, 0.8345, 0.3545)
- TC - Pca, Accuracy, Best minDCF: (4, 0.8335, 0.3609)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.1331	0.1419	92.90%	5
NB	0.1737	0.1750	91.25%	5
TC	0.1812	0.1860	90.70%	5

table 9: min DCF, DCF norm for Gaussian model with PCA = 5 and application (0.5, 1.0, 1.0)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.1537	0.1609	91.95%	4
NB	0.1717	0.1770	90.75%	4
TC	0.1821	0.1850	91.20%	4

table 10: min DCF, DCF norm for Gaussian model with PCA = 4 and application (0.5, 1.0, 1.0)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.1734	0.1759	91.00%	3
NB	0.1746	0.1799	90.75%	3
TC	0.1830	0.1850	91.20%	3

table 11: min DCF, DCF norm for Gaussian model with PCA = 3 and application (0.5, 1.0, 1.0)

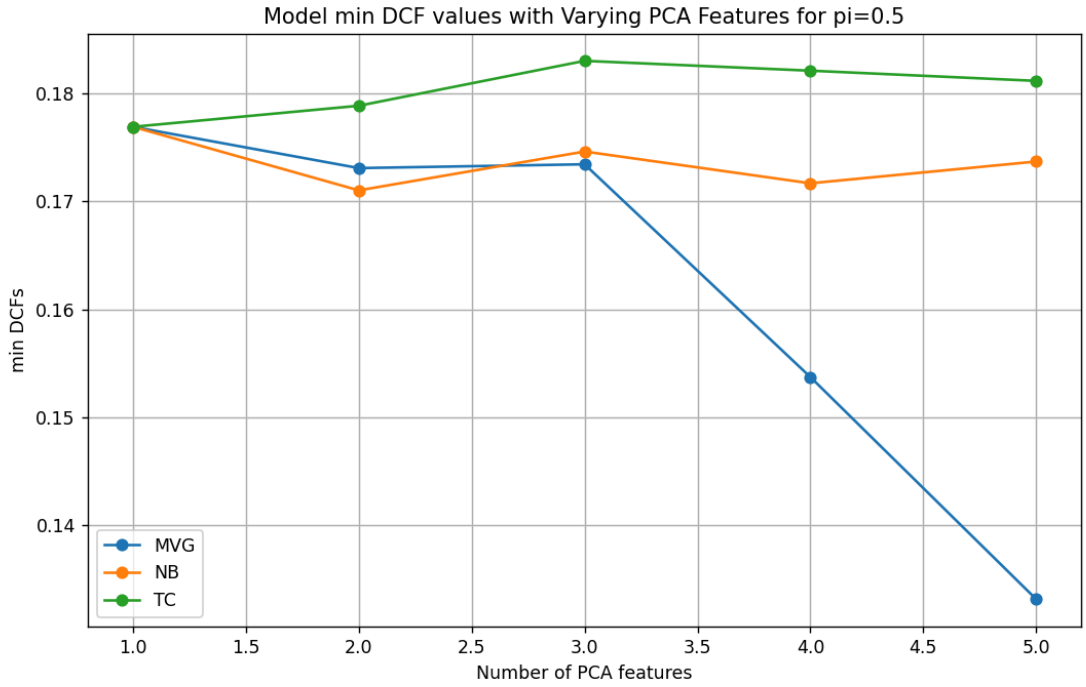

fig 18: min DCF values over different PCA feature and application (0.5, 1.0, 1.0)

- MVG - Pca, Accuracy, Best minDCF: (5, 0.929, 0.1331)
- NB - Pca, Accuracy, Best minDCF: (2, 0.9115, 0.1710)
- TC - Pca, Accuracy, Best minDCF: (1, 0.9065, 0.1769)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.3512	0.3980	86.15%	5
NB	0.4340	0.4660	84.35%	5
TC	0.4451	0.4626	82.95%	5

table 13: min DCF, DCF norm for Gaussian model PCA = 5 and application (0.9, 1.0, 1.0)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.4150	0.4598	85.05%	4
NB	0.4313	0.4630	84.50%	4
TC	0.4441	0.4615	83.00%	4

table 14: min DCF, DCF norm for Gaussian model with PCA = 4 and application (0.9, 1.0, 1.0)

MODEL	MIN DCF	DCF NORM	ACCURACY	PCA
MVG	0.4392	0.4680	84.25%	3
NB	0.4343	0.4591	84.30%	3
TC	0.4342	0.4565	83.25%	3

table 15: min DCF, DCF norm for Gaussian model PCA = 3 and application (0.9, 1.0, 1.0)

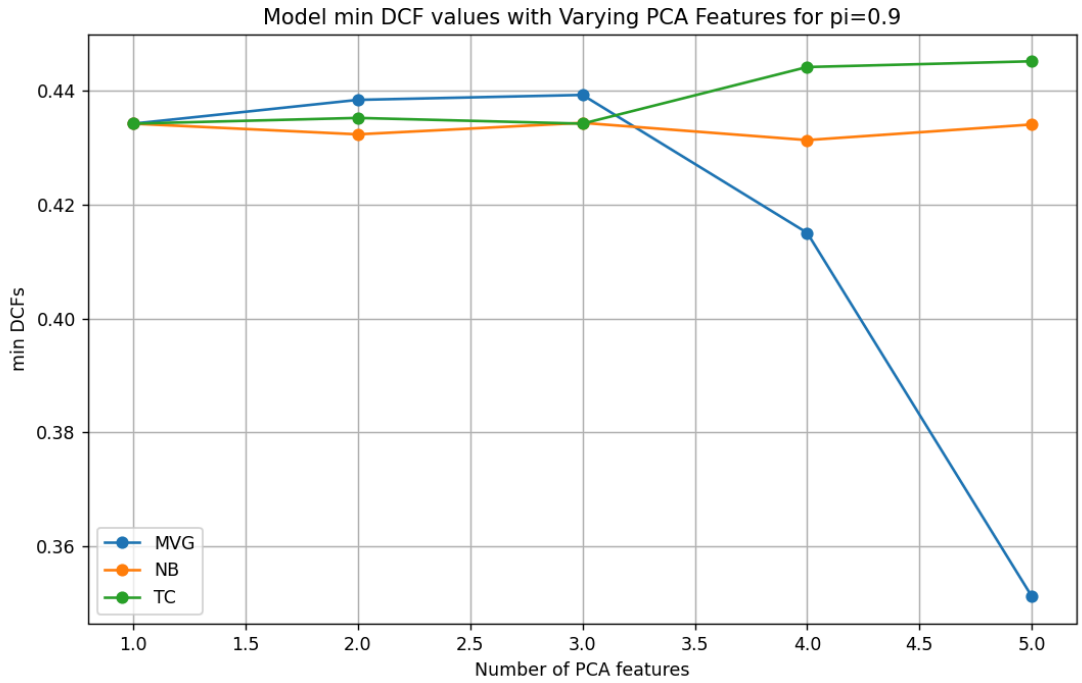

fig 18: min DCF values over different PCA feature and application (0.9, 1.0, 1.0)

- `MVG - Pca, Accuracy, Best minDCF: (5, 0.8615, 0.3512)`
- `NB - Pca, Accuracy, Best minDCF: (4, 0.8725, 0.4313)`
- `TC - Pca, Accuracy, Best minDCF: (3, 0.8475, 0.4341)`

Based on accuracy and DCF the best model after applying pre-processing results to be MVG with PCA 5 features and prior = 0.5 with a 92.90% in accuracy on the test set and a min DCF equals to 0.1331.

Based on the results we can see that applying PCA as pre-processing or not applying it doesn't really change the results in terms of minimum DCF, so it could be a good idea to keep the data as they are without pre-processing it with PCA.

The models that are more calibrated are the one with prior = 0.5, this as we said before is because the distribution of our datasets is about 50/50 for fake and real samples.

We now consider the setup with PCA that gave the best results for pi = 0.1 (that will be our main application). We can plot the Bayes error plot, which is designed to visualize the value of DCF and min DCF across a range of different prior probabilities. This is particularly useful in binary classification tasks where we're interested in understanding how changes in terms of prior probabilities impact the performance of the classification system in terms of cost.

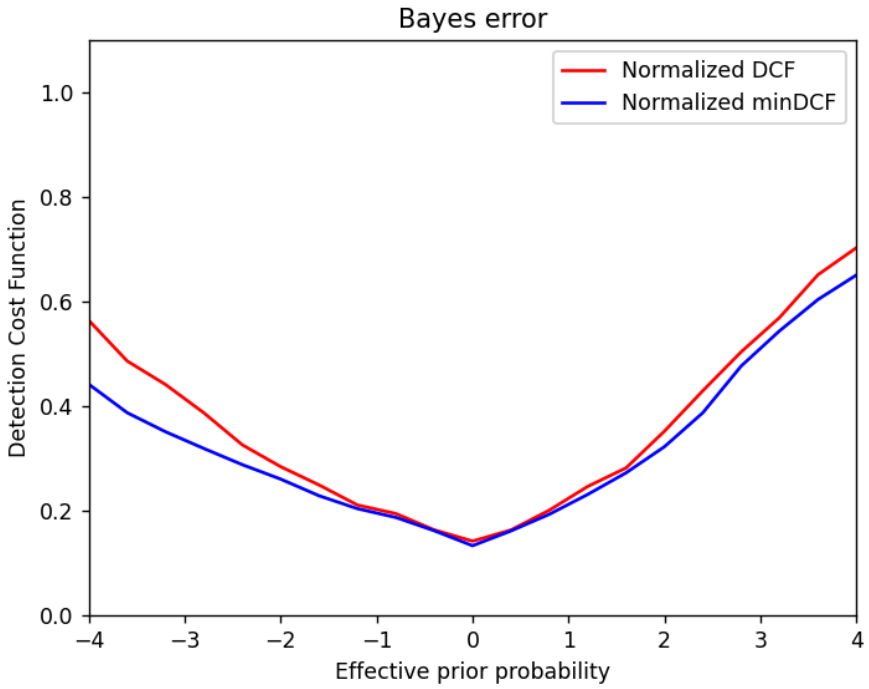

fig 19: Bayes error plot for MVG on data preprocessed with pca 5 on application (0.1, 1.0, 1.0)

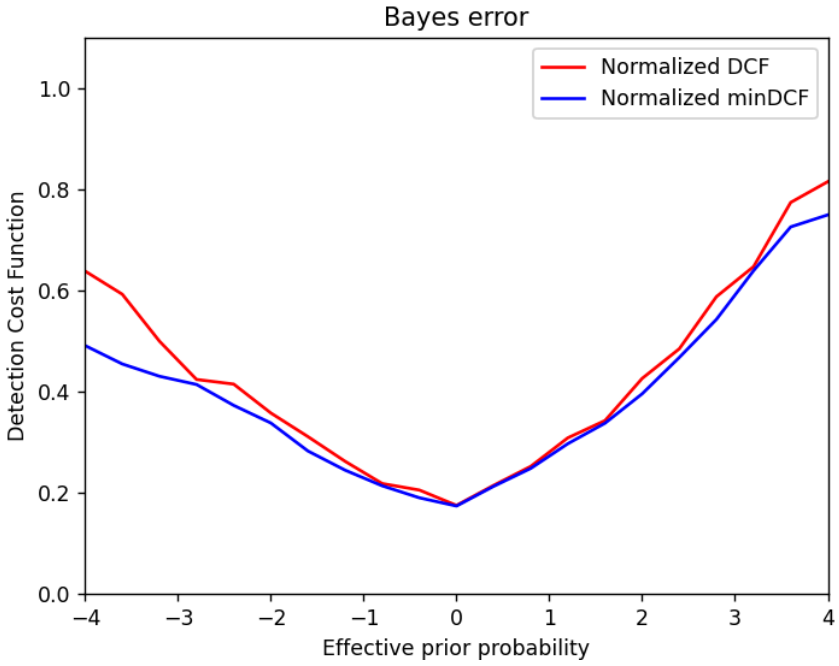

fig 20: Bayes error plot for NB on data preprocessed with pca 5 on application (0.1, 1.0, 1.0)

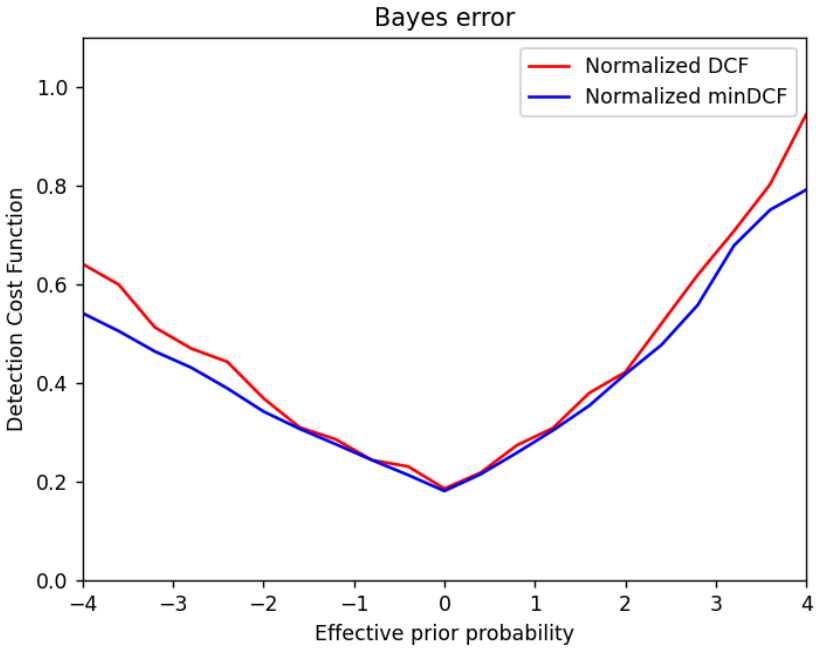

fig 21: Bayes error plot for TC on data preprocessed with pca 5 on application (0.1, 1.0, 1.0)

From this graphs we can observe that in terms of minimum DCF all the models have similar plot (with the MVG one being a little batter, with a lower DCF value), also from values -1 to 2 we have a fairly good calibration over this range, but on the edges the values are very disperse one from the other.

Logistic Regression

Logistic regression is a statistical model used primarily for binary classification tasks. It predicts the probability that a given input belongs to a particular category (typically 1 or 0) based on a sigmoid function.

As for the Gaussian model also here we will consider different variation of the Logistic regression classifier:

- Non weighted: In this version we don't have any set prior to fit the model, we just subtract the logarithmic empirical prior from the score to adapt the scores to different applications.
- Weighted: In this version we change our objective function in a way that allows us to simulate different priors, after fitting to adapt the scores to different applications we need to subtract the logarithmic prior.
- Quadratic expansions: In this version we expand the data to capture more complex relationships between the features and the response variable than what can be modeled by the original linear decision boundary.

We start by analyzing the performance of the non weighted version for different values of lambda, also all our tests would be for prior 0.1.

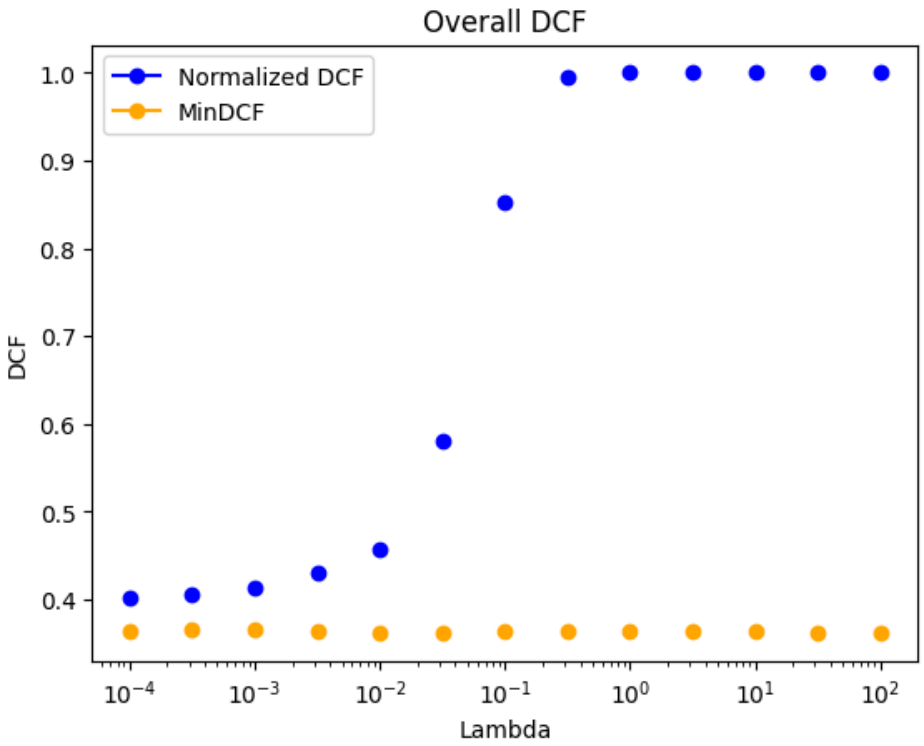

fig 22: DCF and min DCF plot over different values of lambda for LR model non weighted.

We can see from these graphs that the minimum DCF remains consistently lower across all lambda values, demonstrating the importance and effectiveness of threshold optimization in reducing classification costs. Also the Normalized DCF is generally higher, indicating that without threshold optimization, the cost performance is suboptimal.

Another key note is that the higher value of lambdas gives a higher difference between the two DCF.

Since we have a large number of samples, regularization seems ineffective, and actually degrades actual DCF since the regularized models tend to lose the probabilistic interpretation of the scores. To better understand the role of regularization, we analyze the results that we would obtain if we had fewer training samples.

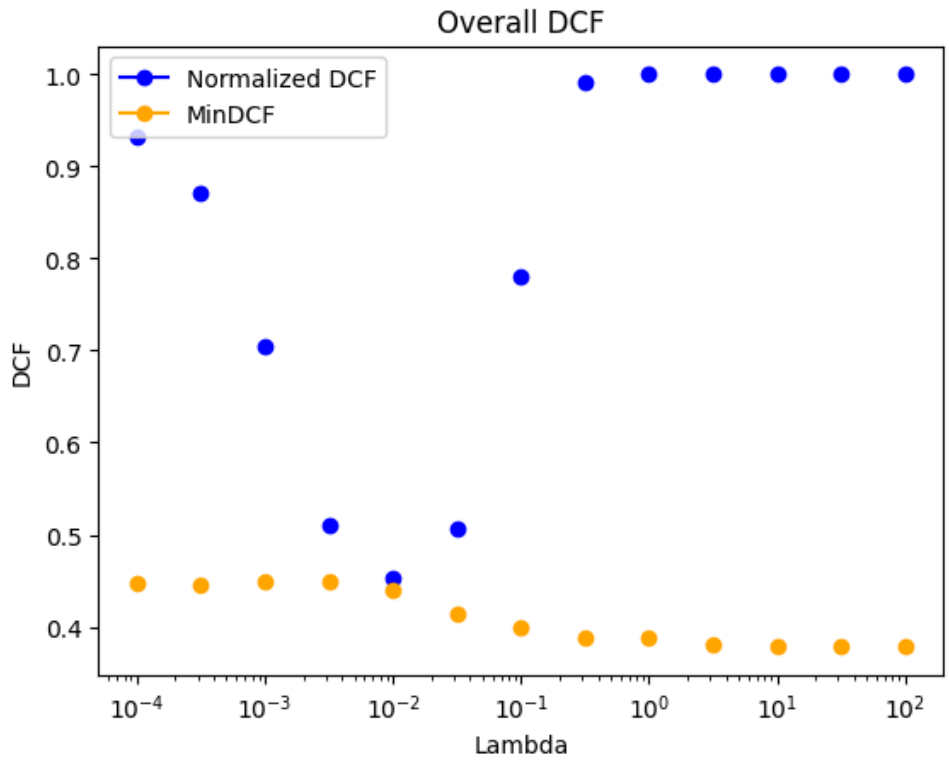

fig 23: DCF and min DCF plot over different values of lambda for LR model non weighted fitted on 1/50 of the datasets.

Knowing that lower values of the regularizer imply larger risk of overfitting, while higher values of the regularizer reduce overfitting, but may lead to underfitting. We could take from this graph that without proper thresholding regularization using a value of lambdas different from 10e-2 could lead us to poor performance on unseen data.

For the min DCF the graph is less stable than the one fitted on the whole dataset, but still can give us fairly good results.

We now move on testing the weighted version of the classifier.

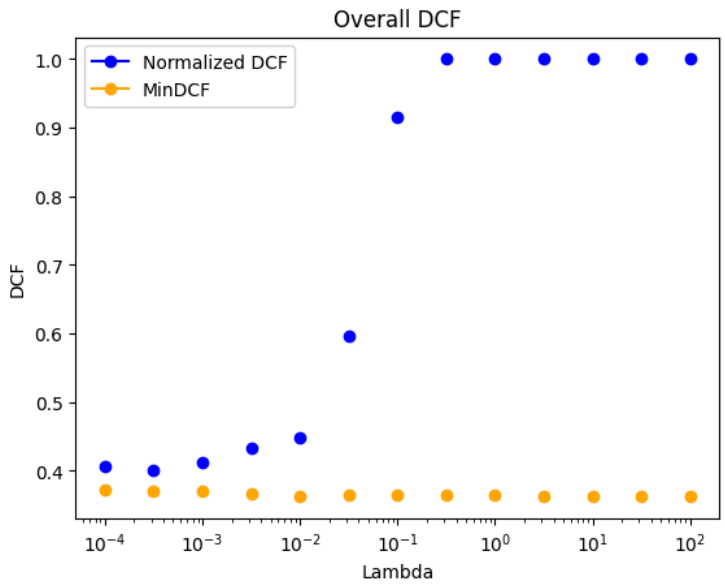

fig 24: DCF and min DCF plot over different values of lambda for LR model weighted.

Using the Weighted version of the classifier with prior = 0.1 doesn't really change the values of the minimum DCF, what we can see are minor differences primarily on the lower lambda values for the normalized DCF.

We can now test the Quadratic Expansion version of the classifier.

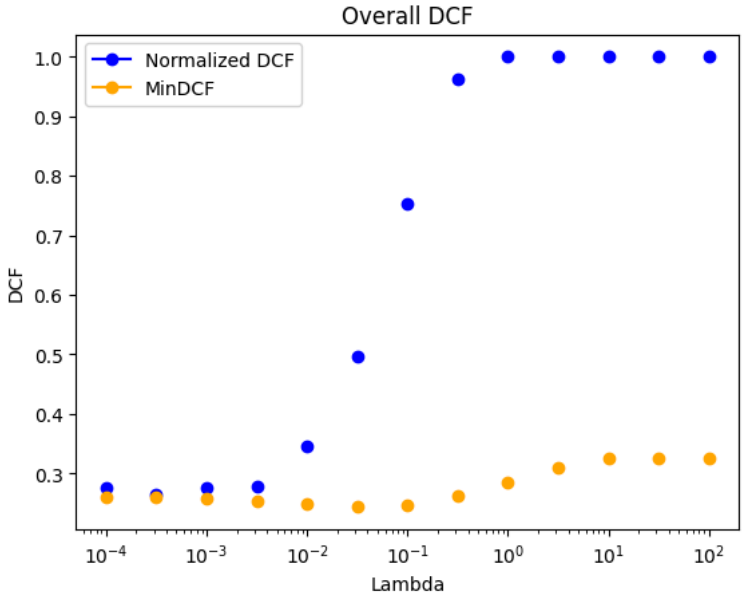

fig 25: DCF and min DCF plot over different values of lambda for LR model quadratic expansion non weighted.

In this case we achieve a less stable min DCF, compared to the previous models (excluding the one fitted on the reduced dataset), but with lower value; which is good since this model is designed to capture more details from the features. The lowest min DCF value we get is 0.2436 with lambda = 3.16e-2.

For the normalized DCF for higher values of lambda we pretty much get the same results as the previous models (excluding the one fitted on the reduced dataset), but for lower values we get values that are very similar to the min DCF, which means that those lambdas for prior = 0.1 are good values to set the hyperparameter, because it reduces the miscalibration due to poor threshold selection.

The non-regularized model is invariant to affine transformations of the data. However, once we introduce a regularization term, affine transformations of the data can lead to different results.

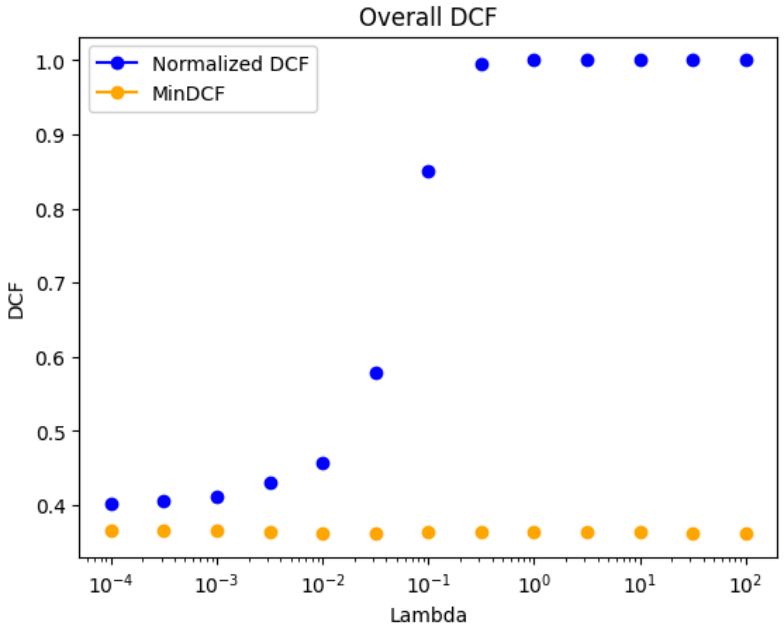

fig 26: DCF and min DCF plot over different values of lambda for LR non weighted model with standardized data

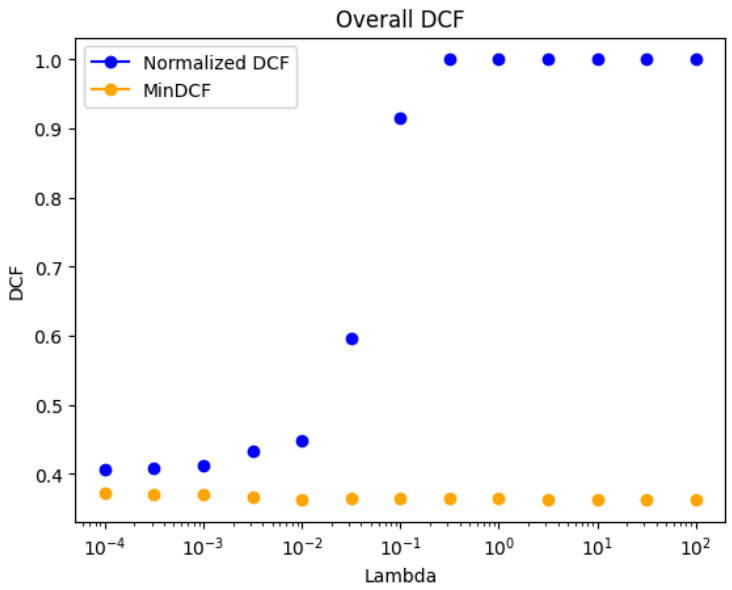

fig 27: DCF and min DCF plot over different values of lambda for LR weighted model with standardized data

Standardizing the data seems to not have any effect on the values of min DCF and normalized DCF for this application (0.1, 1.0, 1.0).

Since we have for now analyzed two models, Gaussian and its variation and Logistic regression and its variation, we can compare them in terms of minimum DCF to see which one gives us the best results for the target application pi = 0.1.

MODEL	SETTINGS	MIN DCF	DCF NORM	ACCURACY
MVG	RAW DATA	0.2629	0.3051	89.60%
NB	RAW DATA	0.2570	0.3022	89.90%
TC	RAW DATA	0.3628	0.4061	86.60%
MVG	PCA 5	0.2738	0.3041	86.30%
NB	PCA 5	0.3545	0.3930	83.45%
TC	PCA 4	0.3610	0.4031	87.10%

table 16: Top result for MVG, NB, TC with and without preprocessing (pca)

MODEL	SETTINGS	LAMBDA	MIN DCF	DCF NORM	ACCURACY
LR NON W	RAW DATA	3.16E-1	0.3620	1.0000	87.00%
LR W	RAW DATA	3.16E-1	0.3620	1.0000	87.00%
QLR	RAW DATA	3.16E-2	0.2436	0.4972	89.30%
LR NON W	STD	3.16E-1	0.3620	1.0000	87.00%
LR W	STD	3.16E-1	0.3620	1.0000	87.00%

table 17: Top result for Logistic Regression non-weighted, weighted and expanded with and without preprocessing (standardization).

If we would only consider the normalized DCF the LR model would not be a good idea, until we do some calibration, But the quadratic expansion variation gives us the best value in terms of minimum DCF.

As for the Gaussians model, they do give us promising results, but the problem is that the assumption on which the model is based is not fully respected (feature 5 and 6 can’t be jointly modeled), so it is possible that even though on this test datasets the model performance is not too bad, on unseen data this models may not give optimal results.

Support Vector Machines (SVM)

SVM works by finding the hyperplane that best separates two classes in the feature space with the maximum margin. The SVM uses points on the edge of the class distributions, known as support vectors, to define the separation boundary.

As for the other models analyzed, even here we have different variations of the model. For SVM we have:

- linear kernel
- polynomial kernel

- rbf kernel

In its simplest linear form, SVM looks for a linear hyperplane that separates the data with the largest possible margin. For non-linearly separable data, SVM can be extended using kernel functions, which allows the algorithm to operate in a higher-dimensional space without explicitly computing the dimensions.

We start with the linear version of the model and train the model on different values of C (and K = 1.0, and prior = 0.1).

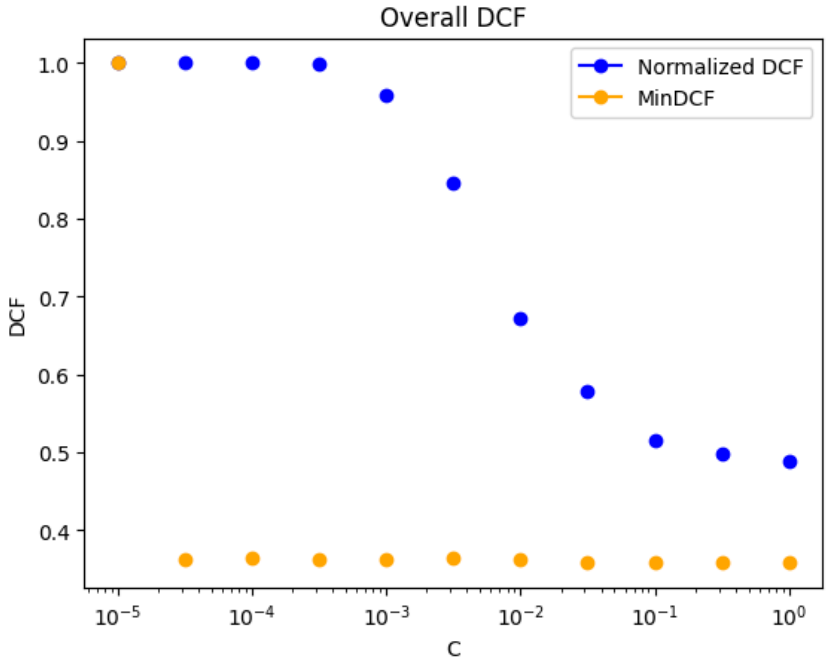

fig 28: DCF and min DCF plot over different values of C for SVM linear model.

For SVM, low values of C imply strong regularization, while large values of C imply weak regularization. The first thing we can note is that the scores are not well calibrated, this means that calibration is indeed needed for a good choice of thresholds.

From this graph we can see that for values under 10e-2 our model could be underfitted and for values over 10e-2 we could be overfitted, so a possible good choice in terms of C could be 10e-2.

As for the minimum DCF, its values are fairly stable except for values of C (probably) less than 10e-5, indicating that the model could perform better if calibration is performed.

Comparing the Logistic Regression non weighted version with the linear SVM we can see that we get slightly better results in terms of minimum DCF.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
LR NON W	RAW DATA	L = 3.16E-1	0.3620	1.0000	87.00%
SVM linear	RAW DATA	C = 1.0E-1	0.3582	0.5162	87.65%

table 18: Comparison between LR non weighted and SVM linear models based on the best min DCF.

We repeat the test on standardized data to verify if this gives us any improvements.

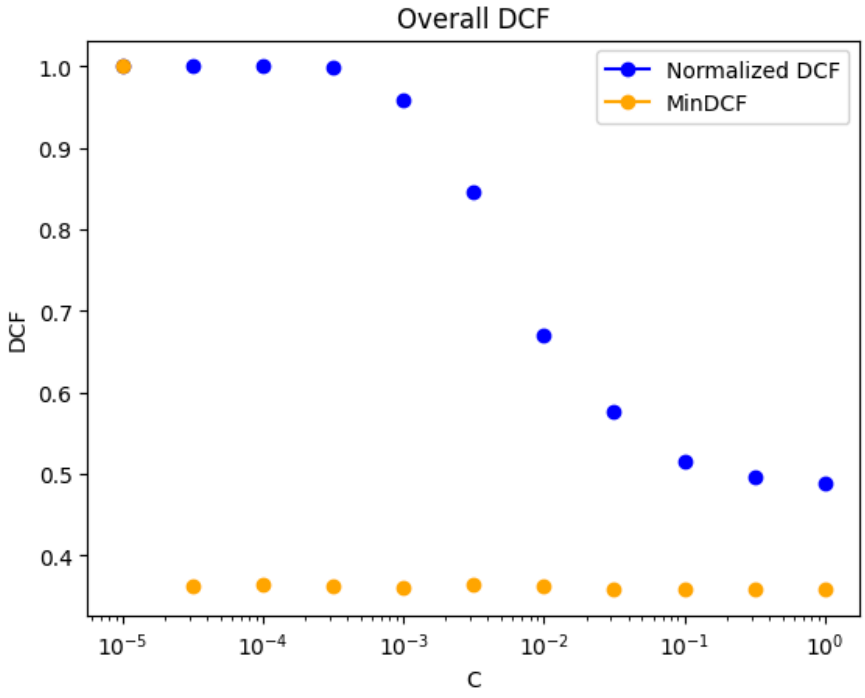

fig 29: DCF and min DCF plot over different values of C for SVM linear model on std data.

We get almost the same results, so we conclude that using raw data or standardizing data doesn't give us any advantages.

We now move to test if with more advanced kernels we can capture complex patterns in our data and achieve better results.

We start with a quadratic kernel (polynomial of degree 2, with c = 1), and we set eps = 0, since the kernel already accounts implicitly for the bias term (due to c = 1). For the next part we will only test on raw data.

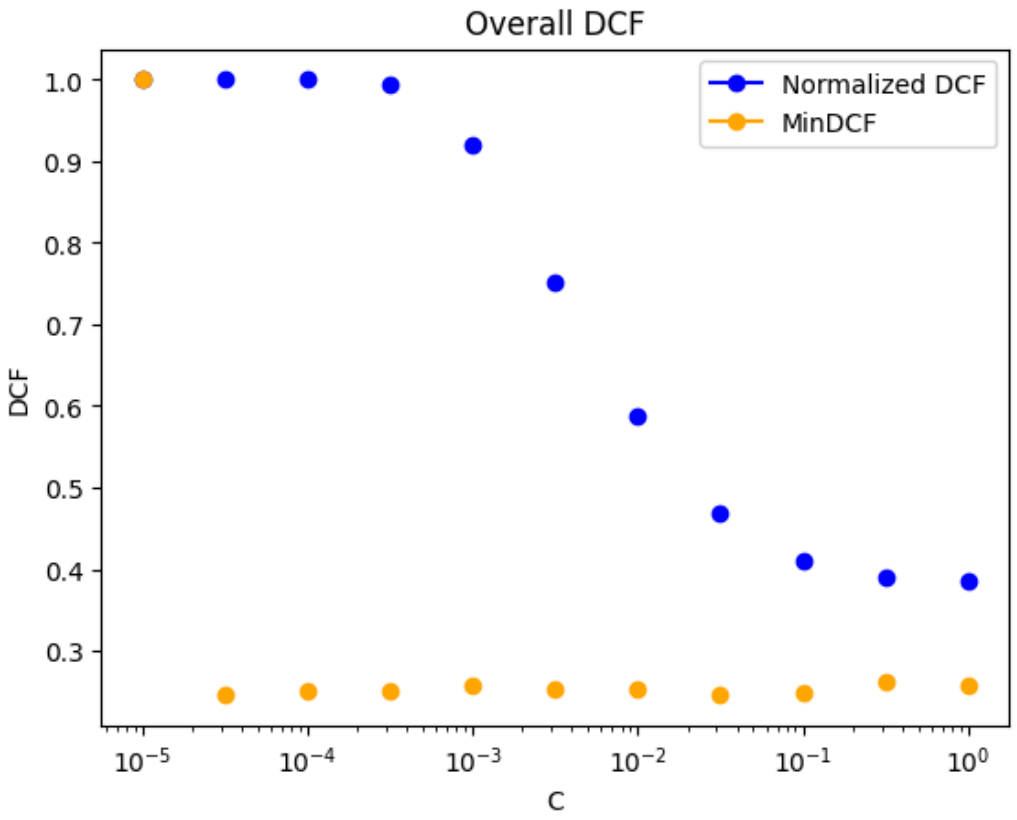

fig 30: DCF and min DCF plot over different values of C for SVM poly kernel model.

In general the pattern is the same for both normalized DCF and minimum DCF for linear and polynomial kernels. But we achieve better results in terms of values going as low as 0.2455 for the minimum DCF.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
SVM linear	RAW DATA	C = 1.0E-1	0.3582	0.5162	87.65%
SVM Poly	RAW DATA	C = 3.16E-2	0.2455	0.4674	88.85%

table 18: Comparison between SVM linear and SVM poly models based on the best min DCF.

So in general we obtain better results in terms of minimum DCF using a complex kernel (polynomial in this case) than with a linear kernel, suggesting that our data has more complex patterns that linear models don't comprehend.

To effectively see if using complex kernels can improve our classification performance we now test a different kernel called RBF on raw data with eps = 0 and different values of C, and with different values of gamma.

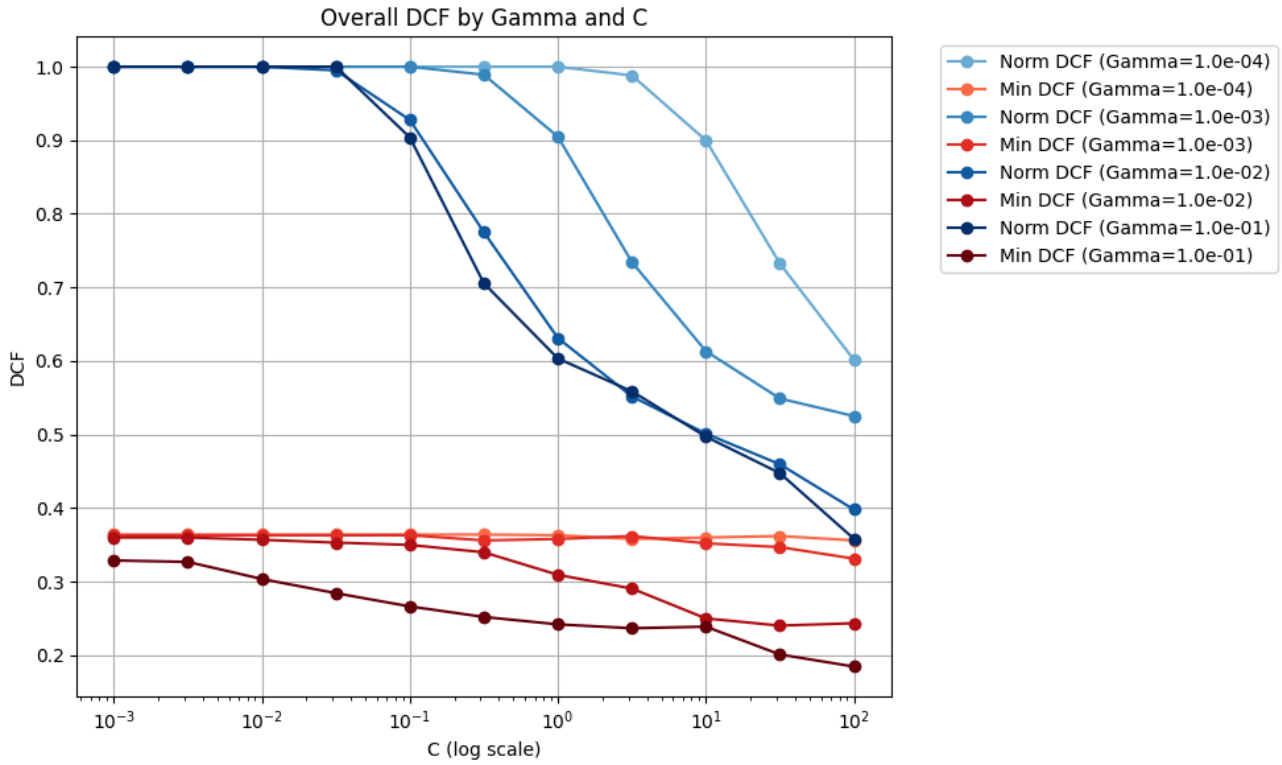

fig 31: DCF and min DCF plot over different values of C and different values of gamma for SVM RBF kernel model.

In this case we can clearly see that higher values of C gives us better results, also higher values of gamma gives us better results. So we can see a pattern, for higher values of C and higher values of gamma (up to 10E-2 in our case) we can achieve a min DCF equals to 0.1845, that is the best results for this application (pi = 0,1, Cfp = 1.0, Cfn = 1.0) we achieved up to now.

As for calibration, like the other models also this needs calibration to achieve results.

Gaussian Mixture Models (GMM)

We now move to our last model, before doing calibration on the whole set of trained models.

Gaussian Mixture Models (GMMs) are probabilistic models used for modeling complex data distributions. GMMs represent a dataset as a combination of multiple Gaussian distributions, each characterized by its mean, covariance, and weight.

For this model we take into consideration two of its variations, full and diagonal, that like for Gaussian models differ on how they handle the covariance matrix.

MODEL	TYPE	PARAMS	MIN DCF	DCF NORM
GMM	FULL	COMP = 1	0.2629	0.3051
GMM	FULL	COMP = 2	0.2159	0.2337
GMM	FULL	COMP = 4	0.2161	0.2395
GMM	FULL	COMP = 8	0.1786	0.1928
GMM	FULL	COMP = 16	0.1631	0.1766
GMM	FULL	COMP = 32	0.2337	0.2499

table 19: Performance for GMM Full with different components

MODEL	TYPE	PARAMS	MIN DCF	DCF NORM
GMM	DIAGONAL	COMP = 1	0.2570	0.3022
GMM	DIAGONAL	COMP = 2	0.2489	0.2674
GMM	DIAGONAL	COMP = 4	0.1481	0.1687
GMM	DIAGONAL	COMP = 8	0.1463	0.1809
GMM	DIAGONAL	COMP = 16	0.1622	0.1769
GMM	DIAGONAL	COMP = 32	0.1766	0.1989

table 20: Performance for GMM Diagonal with different components

What we can see from this table is that this model is performing very well on this dataset and it requires even lower calibration than the other models trained since now.

We achieve good results even with lower component numbers with this model, one thing we can note is that after component 16 for full and 8 for diagonal version we hit like a bottom and after that the minimum DCF keeps increasing again.

This model gave us the best results so far with a value for min DCF equals to 0.1481 on raw data with diagonal covariance and number of components equals to 8.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
QLR	RAW DATA	L = 3.16E-2	0.2436	0.4972
SVM RBF	RAW DATA	C = 1.0E-2	0.1845	0.3581
GMM DIAG	RAW DATA	COMP = 8	0.1463	0.1809

table 21: Comparison between LR non weighted, SVM linear and GMM diagonal models based on the best min DCF.

The model that gives us the best performance not only in terms of minimum DCF but also in terms of normalized DCF is GMM.

To further test our model, we test it also on different applications.

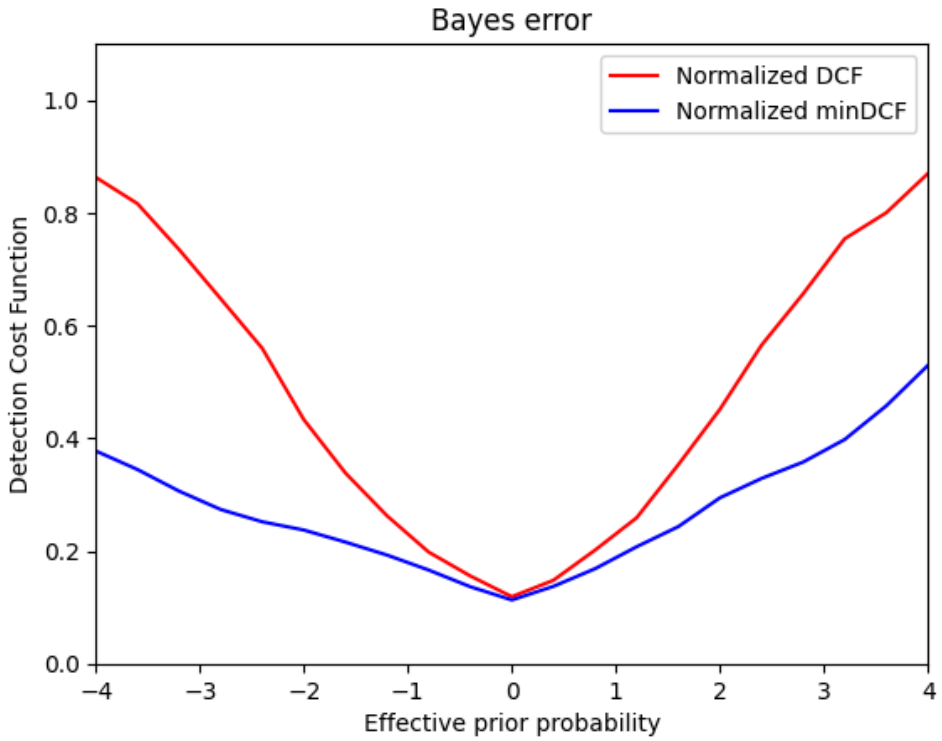

fig 32: Bayes error plot for QLR

- pi: 0.1, MinDCF: 0.2436, Normalized DCF: 0.4972
- pi: 0.5, MinDCF: 0.1130, Normalized DCF: 0.1188
- pi: 0.9, MinDCF: 0.3119, Normalized DCF: 0.4986

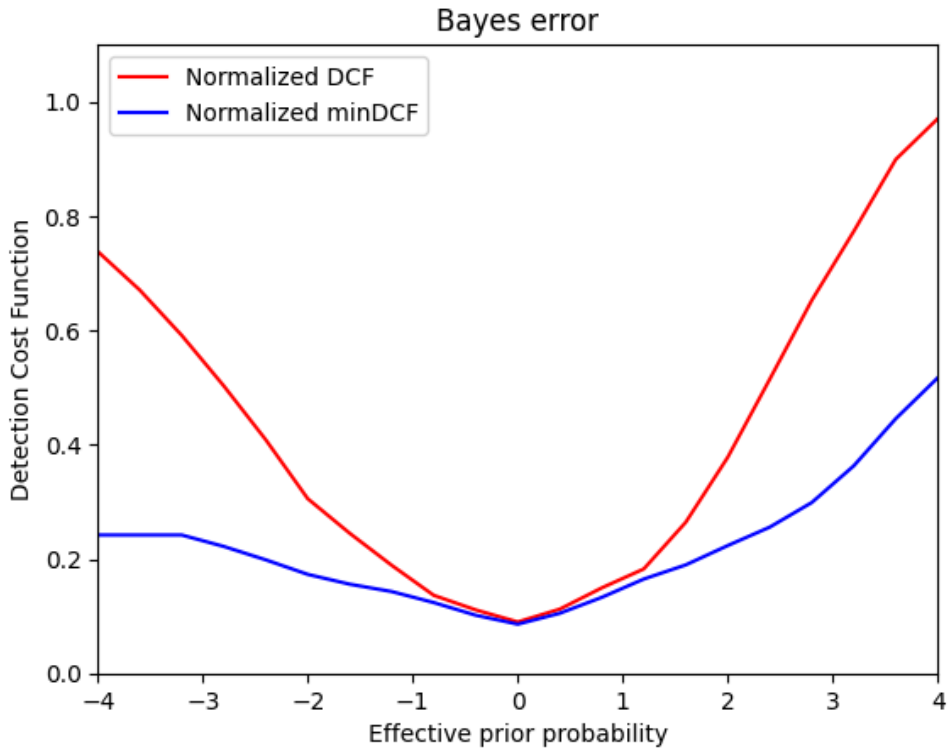

fig 33: Bayes error plot for SVM RBF Kernel

- pi: 0.1, MinDCF: 0.1845, Normalized DCF: 0.3581
- pi: 0.5, MinDCF: 0.0816, Normalized DCF: 0.0900
- pi: 0.9, MinDCF: 0.2398, Normalized DCF: 0.4369

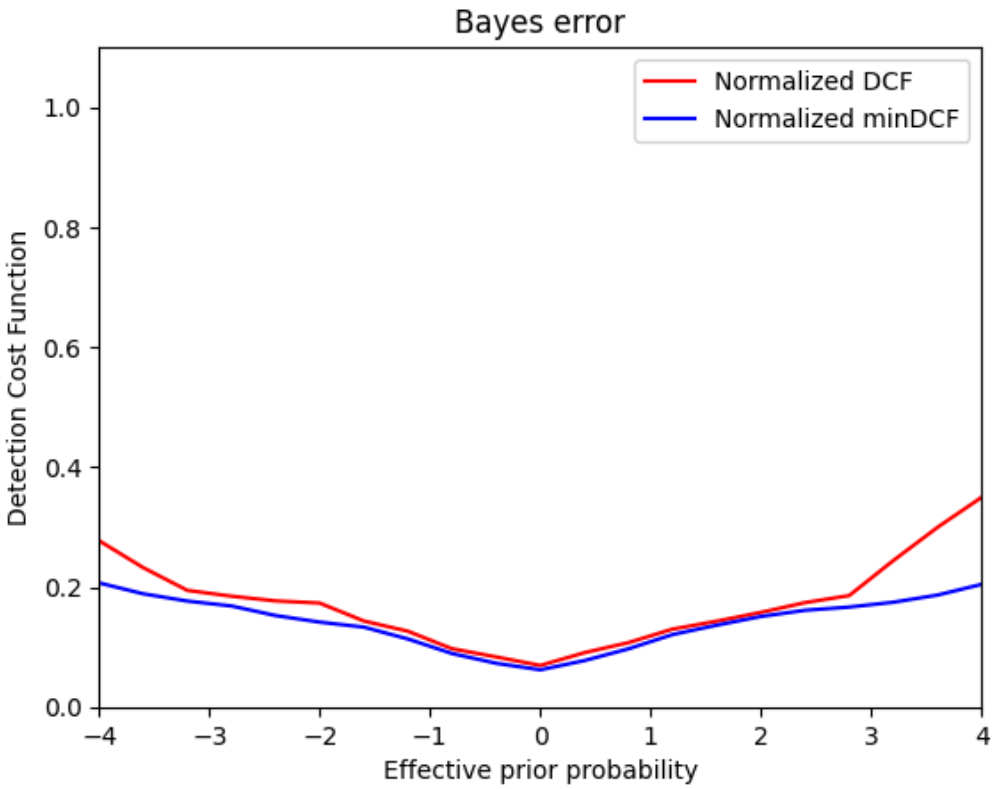

fig 34: Bayes error plot for GMM DIAG

- `pi: 0.1, MinDCF: 0.1463, Normalized DCF: 0.1809`
- `pi: 0.5, MinDCF: 0.0619, Normalized DCF: 0.0691`
- `pi: 0.9, MinDCF: 0.1577, Normalized DCF: 0.1668`

The first thing we can see is that GMM remains quite calibrated for values from -2 to 3 and then diverges a bit, for the other two models this range is smaller (-1, 1 for SVM and -0.5, 0.5 for QLR).

Also QLR (but even SVM a little bit) does not perform very well with effective prior in the range outside of -2, 2 making it a model good only with some precise applications.

Also GMM achieves the best minimum DCF even at effective prior equals to 0, making this model a good choice for our future test on the true evaluation set.

Calibration and Fusion

In this section we will perform calibration and fusion on the top model we found during our previous testing.

For calibration we will use a k-fold approach with a split of 1/10 of the training dataset we will use the training dataset (⅔ of the entire dataset) for calibration and test it on the testing dataset (⅓ of the entire dataset) and only in the end after choosing our final model we will employ the real evaluation dataset (that was not used in any other instance to not "corrupt" the dataset and make it bias towards that set of data). And as prior for calibration this time we will use prior = 0.2 to perform the required operation (but the prior for the target application remains 0.1).

The model we have selected to be calibrated are:

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
QLR	RAW DATA	L = 3.16E-2	0.2436	0.4972
SVM RBF	RAW DATA	C = 1.0E-2	0.1845	0.3581
GMM DIAG	RAW DATA	COMP = 8	0.1463	0.1809

table 21: Comparison between LR non weighted, SVM linear and GMM diagonal models based on the best min DCF.

We start as always with the Logistic Regression model.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
QLR	RAW DATA	L = 3.16E-2	0.2436	0.4972

table 22: QLR minDCF, DCF normalized before calibration

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
QLR	RAW DATA	L = 3.16E-2	0.2472	0.2670

table 23: QLR minDCF, DCF normalized after calibration

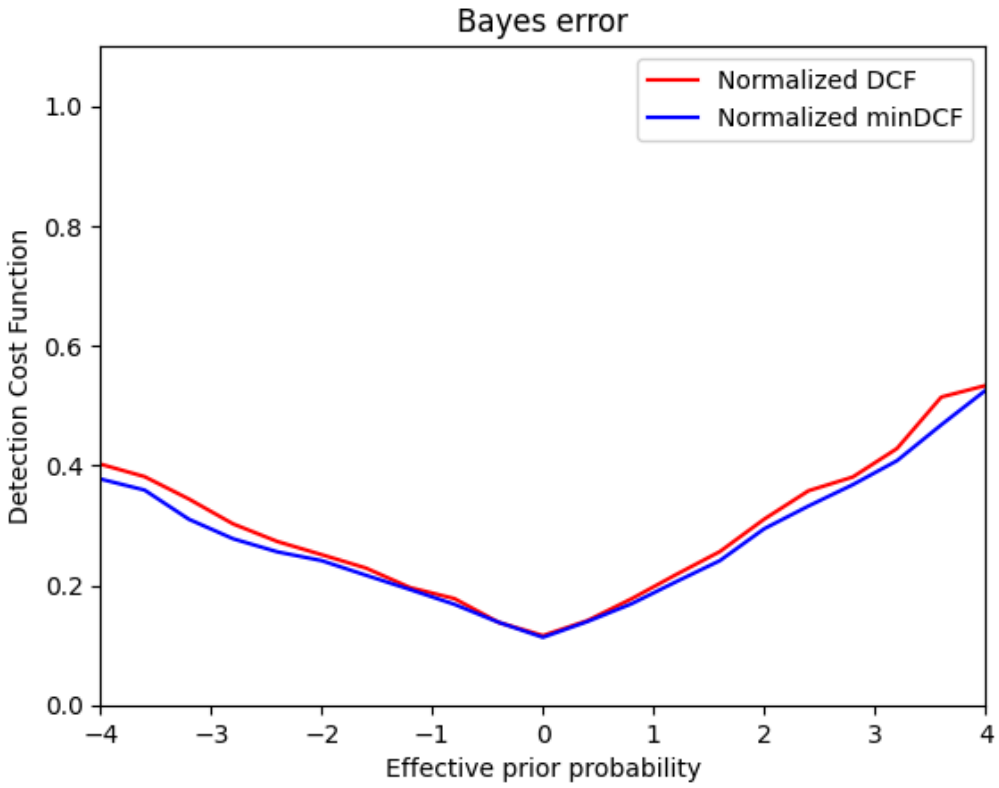

fig 35: Bayes error plot for QLR after calibration

Then we calibrate the Support Vector Machines model.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
SVM RBF	RAW DATA	C = 1.0E-2	0.1845	0.3581

table 24: SVM RBF minDCF, DCF normalized before calibration

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
SVM RBF	RAW DATA	C = 1.0E-2	0.1884	0.1953

table 25: SVM RBF minDCF, DCF normalized after calibration

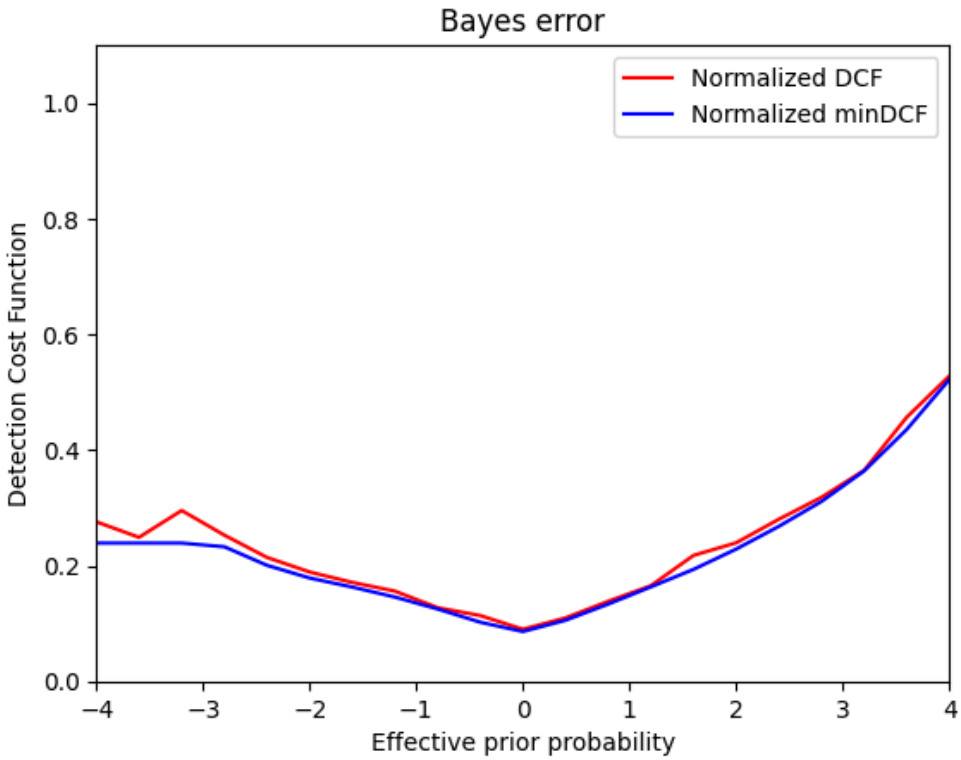

fig 36: Bayes error plot for SVM RBF after calibration

And then we move to the Gaussian Mixture model.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
GMM DIAG	RAW DATA	COMP = 8	0.1463	0.1809

table 26: GMM DIAG minDCF, DCF normalized before calibration

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM
GMM DIAG	RAW DATA	COMP = 8	0.1552	0.1787

table 27: GMM DIAG minDCF, DCF normalized after calibration

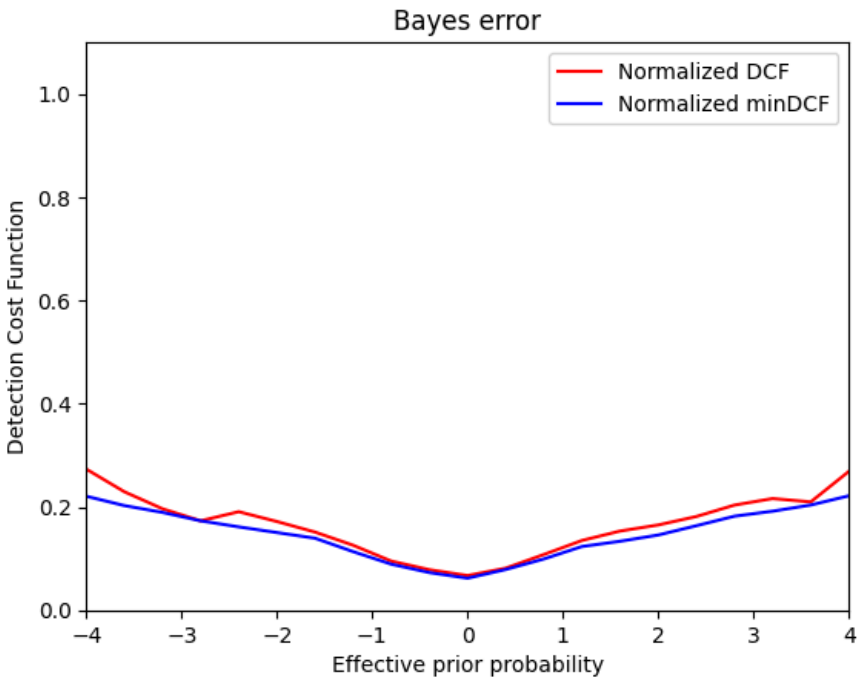

fig 37: Bayes error plot for GMM DIAL after calibration

After calibration we can finally rank the model based on DCF normalized instead of only using minimum DCF.

We can see that after calibration the model with the lowest DCF remains GMM DIAGONAL with 8 components, but in terms of being calibration (overlap between minDCF and DCF) SVM RBF seems to be the model to achieve it better for values of prior log odds from -3 to 4 with only a big discrepancy for value lower than -3.

Another thing to note is that after calibration for all models the minimum DCF became slightly higher, meaning a very small decrease in performance with optimal thresholding.

We now move to verify if fusion techniques on the three models could further improve our performance.

MODEL	SETTINGS	MIN DCF	DCF NORM
FUSION	RAW DATA	0.1582	0.1697

table 28: Fusion model minDCF, DCF normalized after calibration

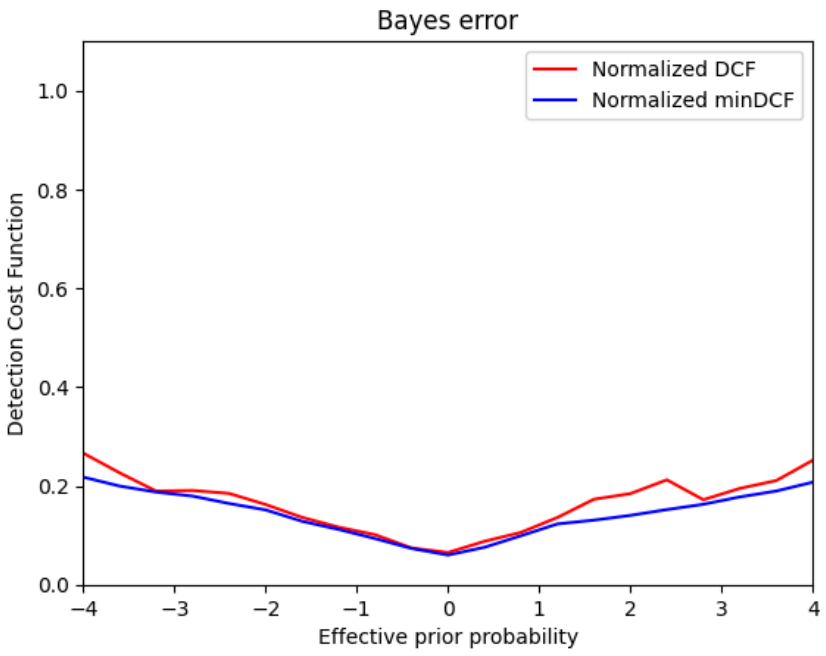

fig 38: Bayes error plot for Fusion model.

Overall the fusion model performance is substantially good compared to SVM RBF and QLR alone, but gives slightly worse results then GMM (for this dataset), making it not a good choice between the two because we will still need to train GMM.

As for calibration the fusion model achieves good results for values of prior log odds from -3 to 1 but with values after 1 we can see a big disparity between min DCF and Normalized DCF.

Before doing evaluation on the real test set we need to choose our best model, based on our results GMM for its capacity to model different gaussian for each feature, let's see his strength points:

- It doesn't take too long to train (obviously it depends on the number of components on which we fit the model); for comparison SVM RBF, which is our second best model, took 10 times the time needed to train GMM.
- Has the lowest minDCF of all three models for pi = 0.1 (0.1463).
- Even without calibration the performance of the model is good enough.
- It's the one on which calibration has the best effect.

Evaluation

So our model of choice is GMM DIAGONAL with 8 components. We will now evaluate this system that we choose and perform further analysis to understand if our choice was indeed good for our application.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
GMM DIAG	RAW DATA	COMP = 8	0.2026	0.2209	94.28%

table 29: GMM DIAG minDCF, DCF normalized before calibration for evaluation data before calibration

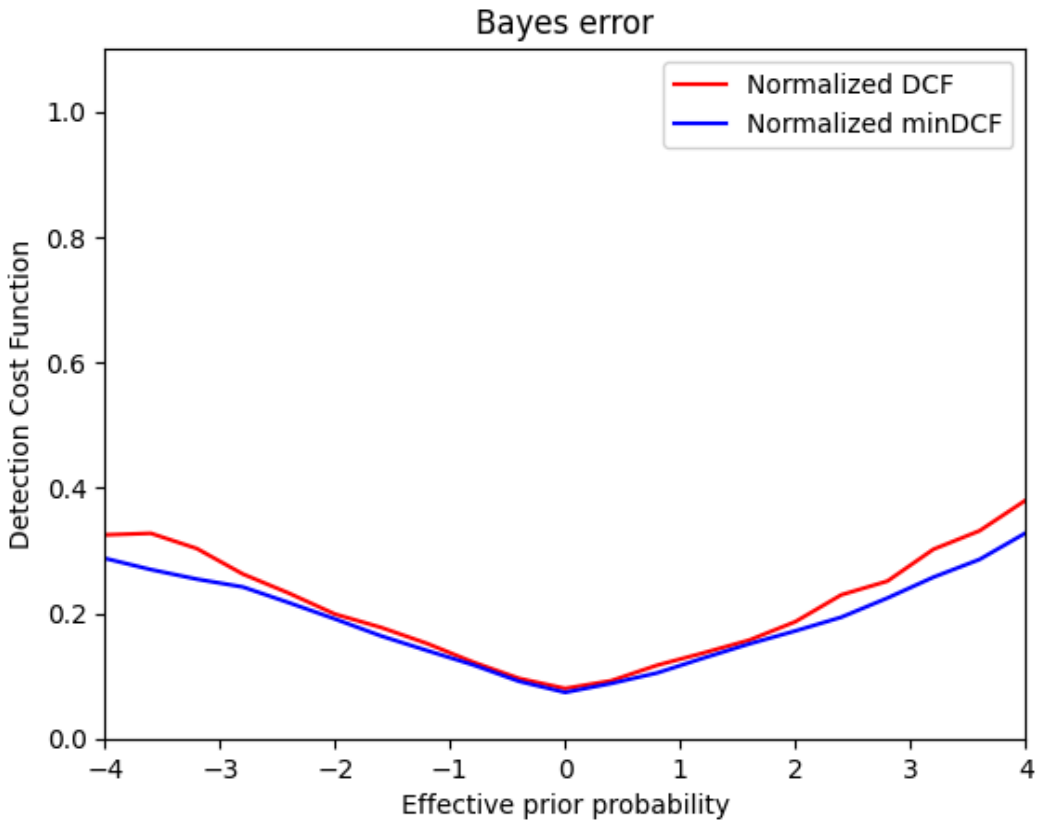

fig 39: Bayes error plot for GMM DIAG model on evaluation data before calibration.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
GMM DIAG	RAW DATA	COMP = 8	0.2026	0.2183	93.88%

table 29: GMM DIAG minDCF, DCF normalized before calibration for evaluation data after calibration

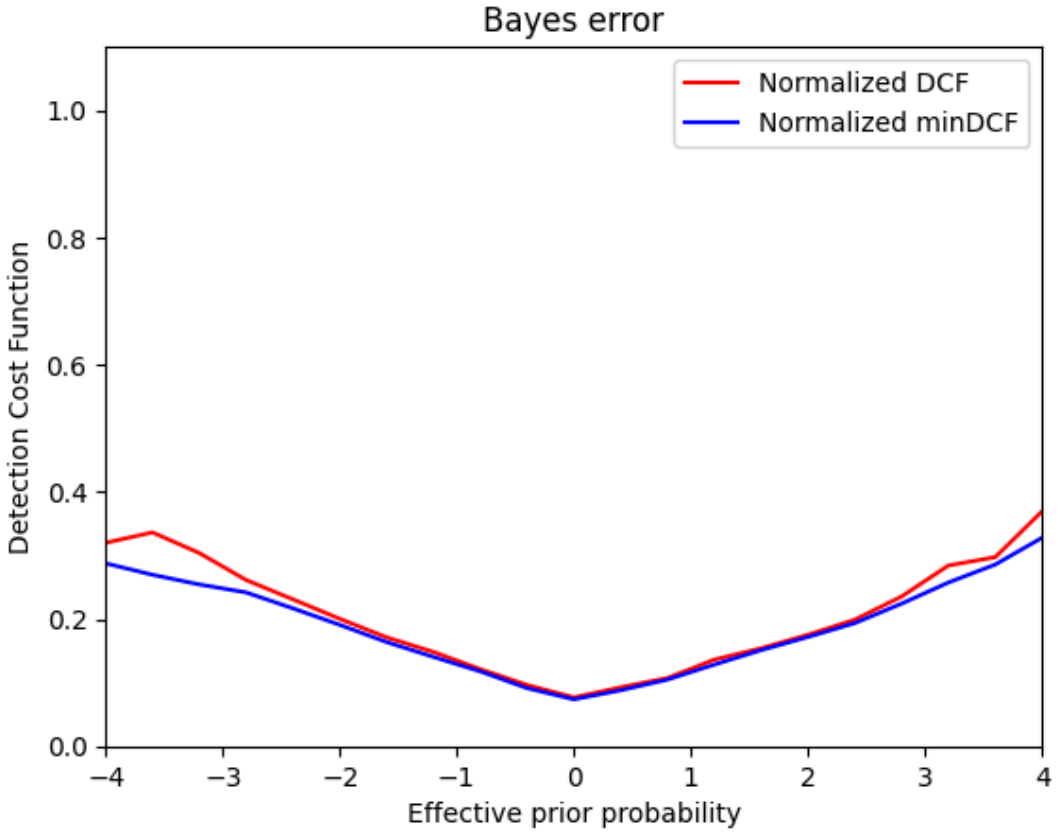

fig 40: Bayes error plot for GMM DIAG model on evaluation data after calibration.

Our model performance on the evaluation dataset results to be very good, with a minimum DCF of 0.2026 just 0.6 difference with our training dataset.

Analyzing the Bayes error plot we can also say we are confident that the model is well calibrated even without calibration, only on the edges we get values that do not quite conform with the minimum DCF for the normalized DCF.

To see if our choice was the best one we also test the other two models and the fused one.

We start as always by testing first the Logistic Regression model.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
QLR	RAW DATA	L = 3.16E-2	0.3515	0.4942	76.17%

table 30: QLR minDCF, DCF normalized before calibration on the evaluation set

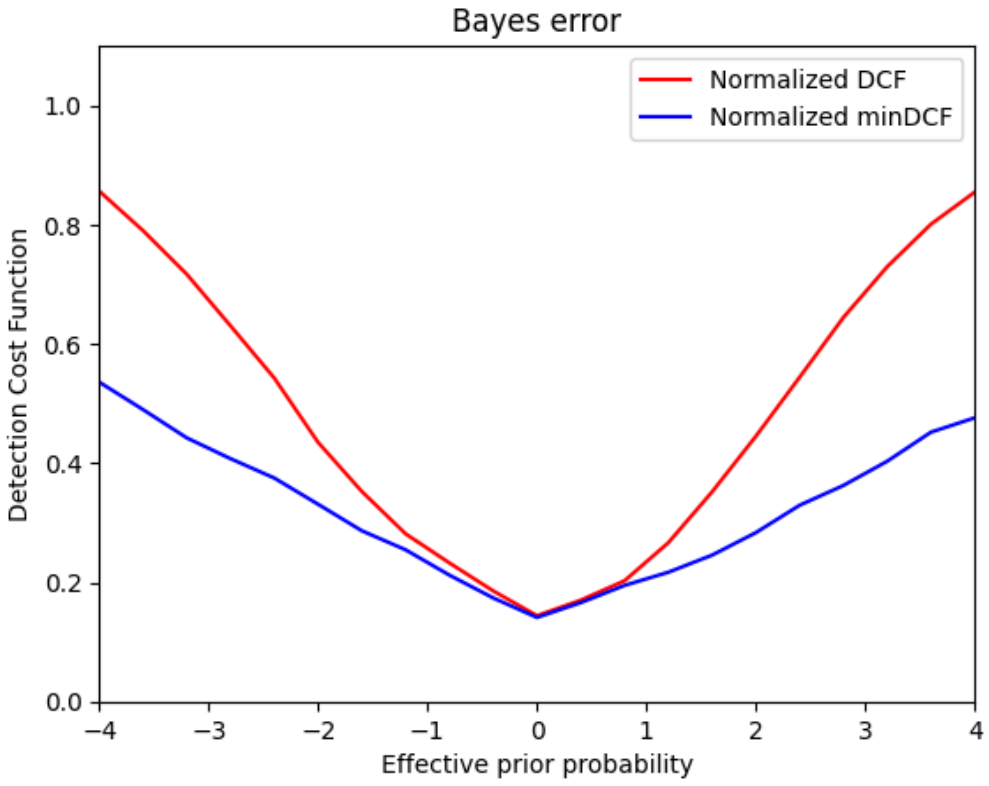

fig 39: Bayes error plot for QLR model on evaluation data before calibration.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
QLR	RAW DATA	L = 3.16E-2	0.3515	0.3780	89.77%

table 31: QLR minDCF, DCF normalized after calibration on the evaluation set

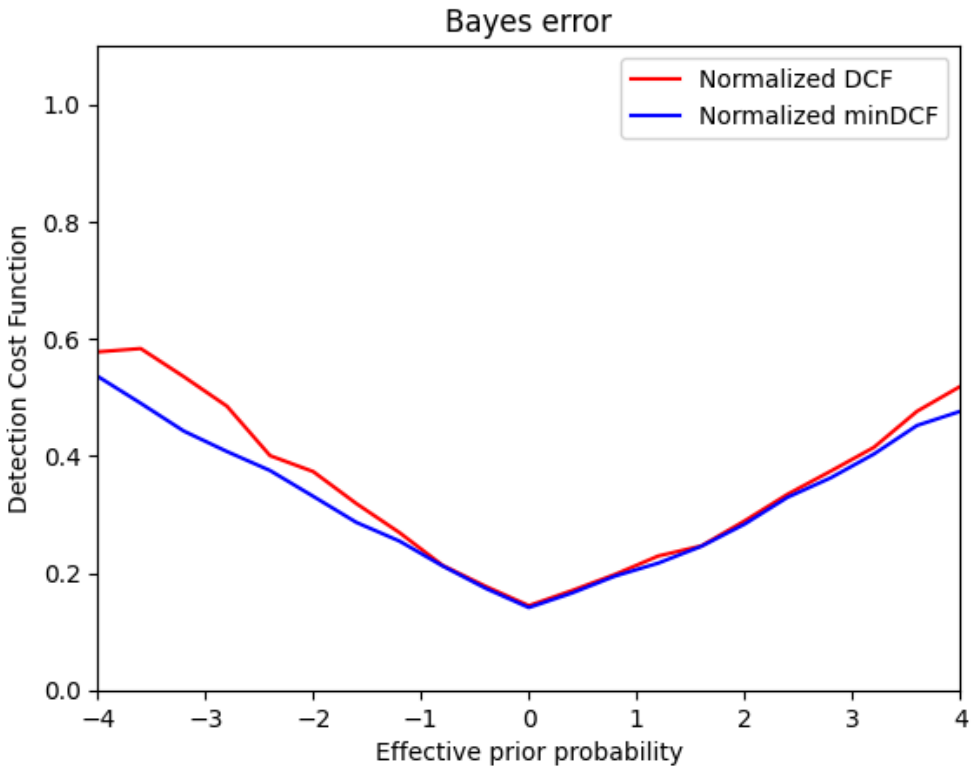

fig 39: Bayes error plot for QLR model on evaluation data after calibration.

Then we test the SVM RBF model.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
SVM RBF	RAW DATA	C = 1.0E-2	0.2636	0.3634	82.55%

table 32: SVM RBF minDCF, DCF normalized before calibration on the evaluation set

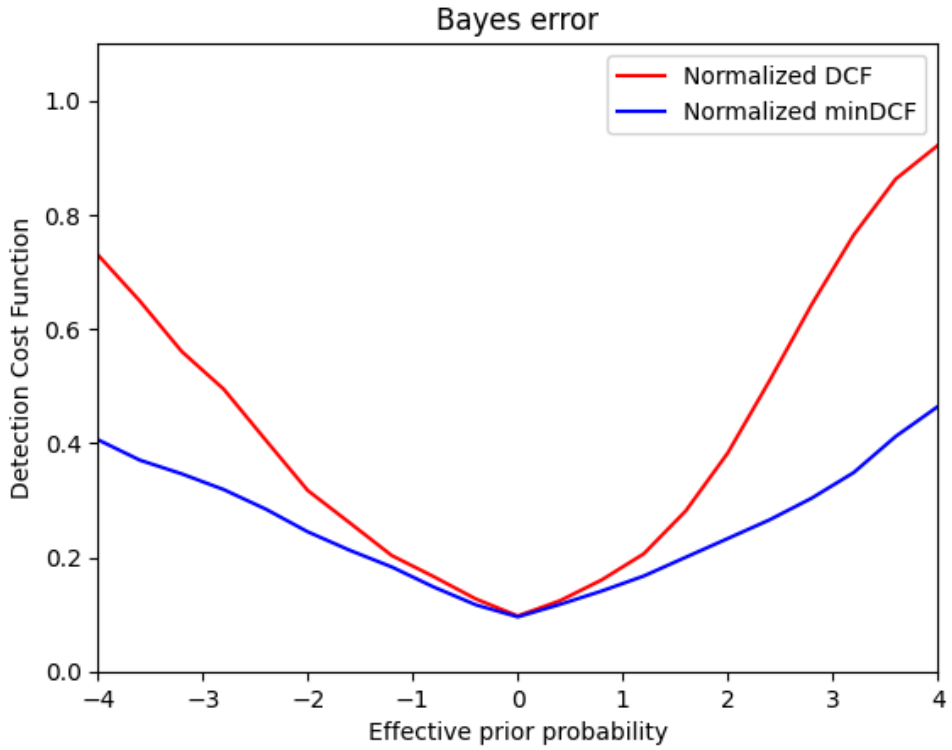

fig 39: Bayes error plot for SVM RBF model on evaluation data before calibration.

MODEL	SETTINGS	PARAMS	MIN DCF	DCF NORM	ACCURACY
SVM RBF	RAW DATA	C = 1.0E-2	0.2636	0.2843	92.32%

table 33: SVM RBF minDCF, DCF normalized after calibration on the evaluation set

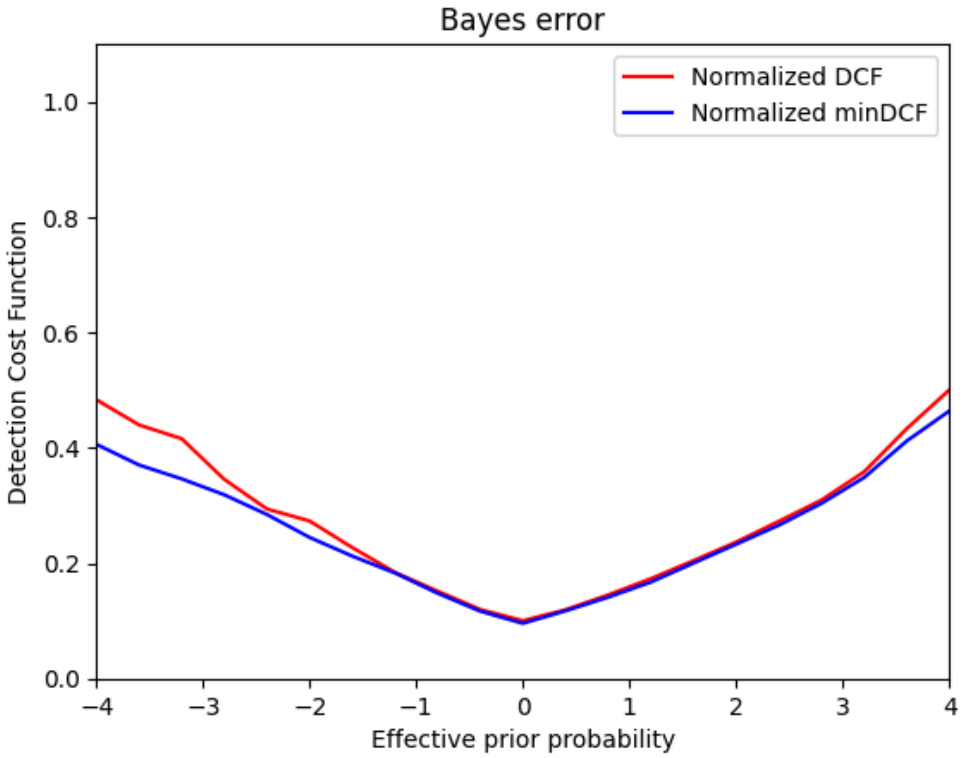

fig 39: Bayes error plot for SVM RBF model on evaluation data after calibration.

MODEL	SETTINGS	MIN DCF	DCF NORM	ACCURACY
FUSION	RAW DATA	0.1961	0.2137	94.12%

table 34: Fusion model minDCF, DCF normalized after calibration on the evaluation set

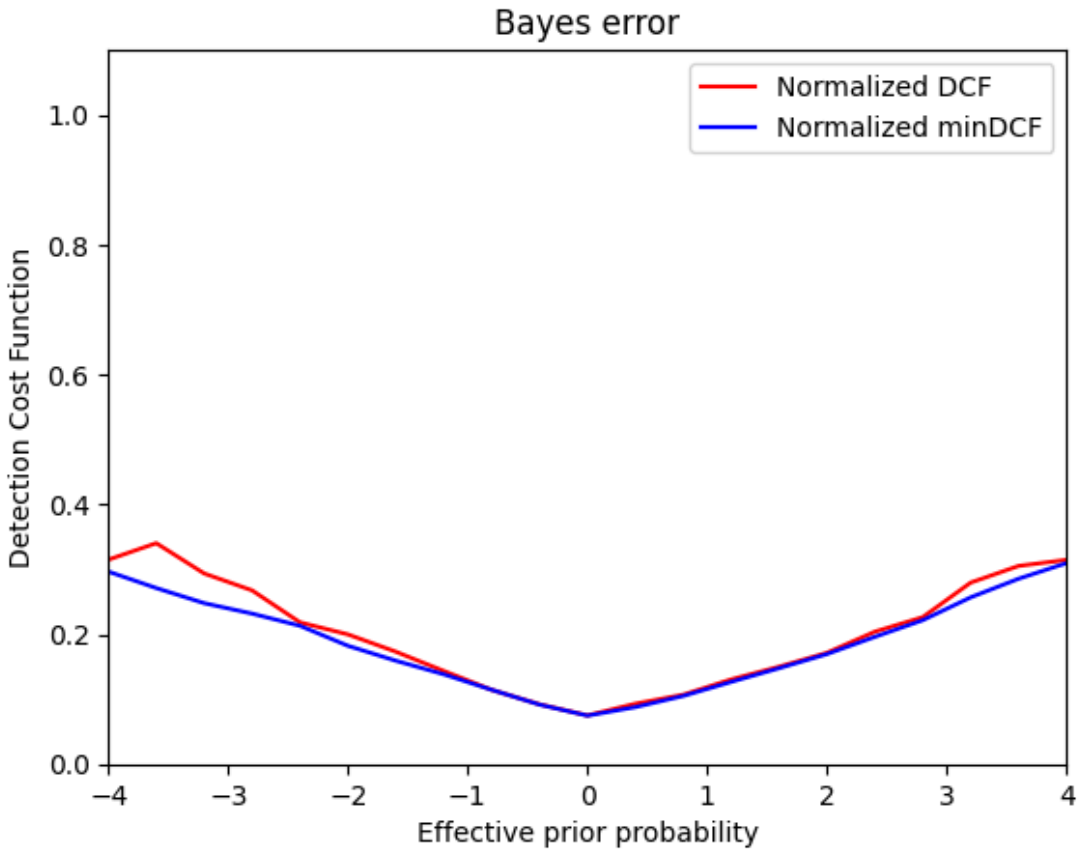

fig 39: Bayes error plot for FUSION model on evaluation data after calibration.

We can see that all the other models perform worse than the GMM DIAG 8 (beside fusion that performed slightly better than the calibrated version, but still worse than the not calibrated version) in terms of accuracy for the evaluation dataset. As for minimum DCF and DCF normalization, the fusion model is the one performing slightly better than the model we choose, but for the consideration we stated before (time and need to train three different models) GMM may still be the best choice.

As for calibration, the strategy we implemented succeeded in calibrating all the models we trained.

For the last test we will take the GMM Models and train it on the evaluation dataset to see if other variations of the model would have given us better results in terms of minimum DCF and DCF normalization.

MODEL	TYPE	PARAMS	MIN DCF	DCF NORM
GMM	FULL	COMP = 1	0.4073	0.4334
GMM	FULL	COMP = 2	0.2968	0.3032
GMM	FULL	COMP = 4	0.2462	0.2551
GMM	FULL	COMP = 8	0.1802	0.1937
GMM	FULL	COMP = 16	0.2025	0.2063
GMM	FULL	COMP = 32	0.2294	0.2443

table 35: Performance for GMM Full with different components

MODEL	TYPE	PARAMS	MIN DCF	DCF NORM
GMM	DIAGONAL	COMP = 1	0.4098	0.4301
GMM	DIAGONAL	COMP = 2	0.3657	0.3723
GMM	DIAGONAL	COMP = 4	0.1947	0.2043
GMM	DIAGONAL	COMP = 8	0.2026	0.2209
GMM	DIAGONAL	COMP = 16	0.1895	0.2040
GMM	DIAGONAL	COMP = 32	0.1955	0.2067

table 36: Performance for GMM Diagonal with different components trained on the evaluation dataset

During our previous test we selected GMM DIAG 8 COMPONENTS as our best model, as we can see from these two tables, this is not the best model we could

have picked. There are 5 different settings that give us lower min DCF value, the best one would have been GMM FULL DIAG 8.

So indeed 8 components was enough to train the model, but a full covariance matrix would have picked slightly better details from the dataset leading to a better performance.

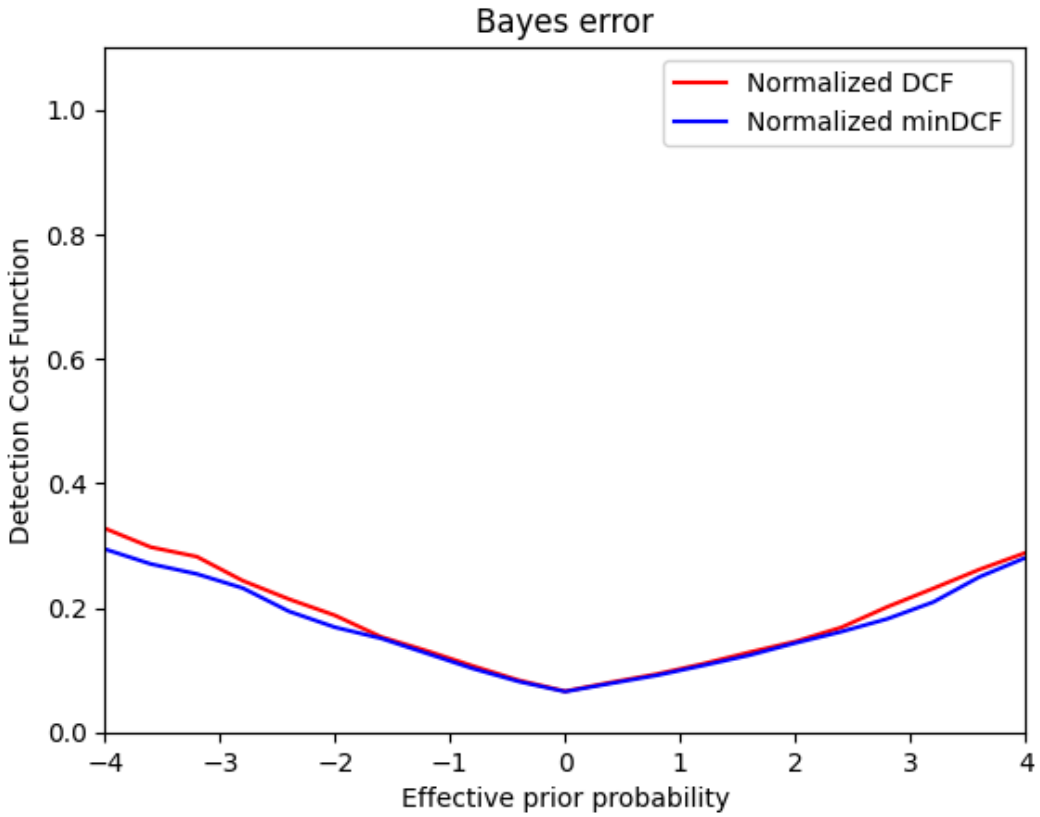

fig 40: Bayes error plot for GMM FULL 8 COMPONENT model on evaluation data.

Plotting the bayes error also seems to align with the idea that this model performs better for the evaluation dataset, as we can see the data is well calibrated, especially from -2 to 2.

Conclusion

We started from a dataset and trained several models on a training split (⅔ of the dataset) and tested it on the remaining part of the dataset (⅓ of the dataset).

We trained Gaussian models, Logistic regression models, Support vector machines models and Gaussian Mixture models. All of them have weak points and strength points but only one made itself stand on the other, the Gaussian Mixture model from the beginning gave us satisfying results.

In the end we saw that the best model would have been GMM FULL 8 COMPONENTS, but our choice was GMM DIAG 8 COMPONENTS, given that we could not use the evaluation data for anything during testing (this would have "corrupt" the results) our choice was the best we could have made with the data at our disposition.