Machine Learning & Pattern Recognition

Project - Fingerprint spoofing

Santiago Romero[1]

[1]S314233

11/09/2024

Politecnico di Torino

Abstract

This project addresses the challenge of fingerprint spoofing detection by utilizing a variety of machine learning techniques to differentiate between genuine and counterfeit fingerprint images. The study involves a detailed analysis of a balanced dataset comprising 6,000 samples through data visualization, principal component analysis (PCA), linear discriminant analysis (LDA), and the application of several classifiers including Gaussian models, logistic regression, and support vector machines. The effectiveness of these methods is evaluated based on their classification accuracy, and the results highlight the potential of combining these techniques to enhance the reliability of spoofing detection systems. This approach highlights the complexities of fingerprint verification and shows the practical importance of advanced pattern recognition in security applications.

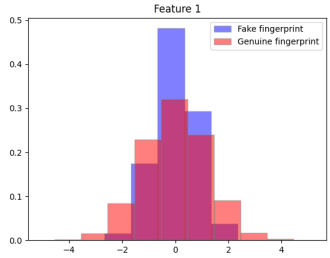
Contents

1 Introduction

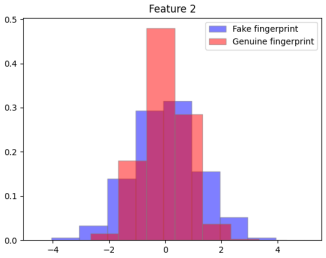
This paper outlines a project done in a sequence of laboratories on the Machine learning and Pattern Recognition course follow in the Politecnico di Torino. This project was focused on developing a classifier for fingerprint spoofing detection, which involves distinguishing between genuine and counterfeit fingerprint images. The dataset comprises labeled samples, with genuine fingerprints labeled as 1 (True) and fake fingerprints labeled as 0 (False).Each sample is represented by a 6-dimensional, continuous-valued vector, summarizing high-level characteristics of the fingerprint image. The training set is contained in the file "Train.txt," and the evaluation set is in "Test.txt." Both files consist of rows where the first six columns represent the fingerprint image components and the last column contains the corresponding label.The training dataset includes 6,000 samples, with 3,010 genuine fingerprints and 2,990 fake fingerprints, making the dataset generally balanced. Approximately 49.83% of the samples represent fake fingerprints.

2 Data Visualization and analysis

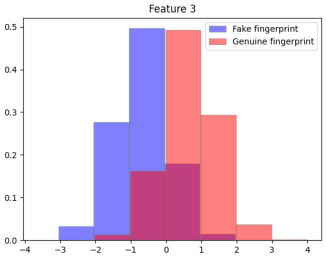
Beginning with the second laboratory, we were tasked with creating visualizations of the dataset to analyze the means and variances of different features using the input data provided in the Train.txt file. To achieve this, we developed a Python script named 'visualization_data.py', which generates histograms for all the features and pair wise scatter plots for the different features. Fro answering the related questions only the scatter plots for the pairs 1&2, 3&4, and 5&6 where presented. The results of these visualizations were as follows:



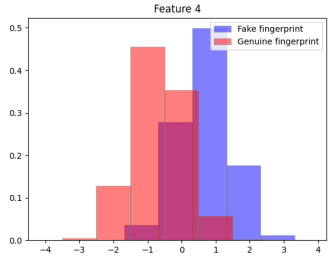
(a) Histogram Feature 1



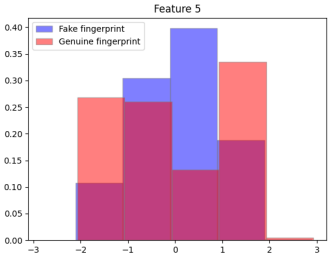
(b) Histogram Feature 2



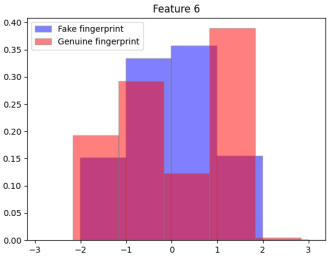
(c) Histogram Feature 3



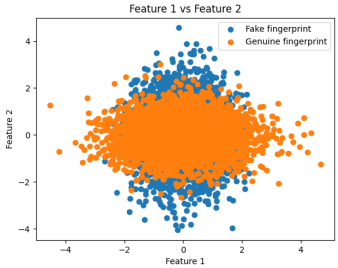
(d) Histogram Feature 4



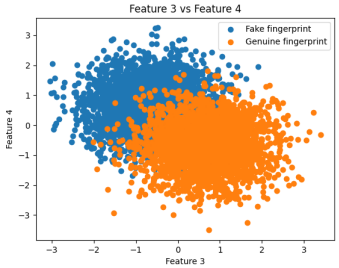
(e) Histogram Feature 5



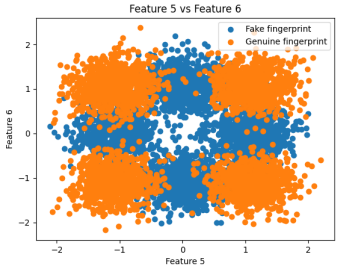
(f) Histogram Feature 6



(g) Scatter plot Feature 1 vs 2



(h) Scatter plot Feature 3 vs 4



(i) Scatter plot Feature 5 vs 6

The scatter plot for Feature 1 vs. Feature 2 shows significant overlap between the fake and genuine fingerprint classes. This overlap is especially prominent in the central region of the plot, indicating that the classification task between these two classes using only these features would be challenging. Both classes show similar means and variances for these features, which further complicates classification. The histograms for Feature 1 and Feature 2 show a unimodal distribution for each class, with single prominent peaks, suggesting that values for each feature cluster around a central value with few outliers.

When examining the histograms and scatter plot for Feature 3 vs. Feature 4, it is observed that there is still some degree of overlap between the classes, though it is less pronounced than in the first two features. The means of Feature 3 and Feature 4 differ slightly between the genuine and fake fingerprints, indicating better discriminative potential. The variances for these features are also different. Genuine fingerprints exhibit a more spread out distribution in Feature 4, while fake fingerprints are more concentrated around certain values. The histograms for these features reveal a bimodal distribution for fake fingerprints, suggesting the presence of two groups with Gaussian patterns, whereas genuine fingerprints maintain a more unimodal distribution.

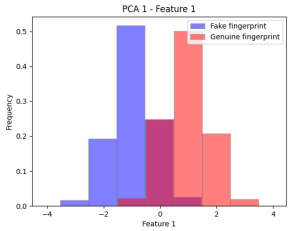
The scatter plot for Feature 5 vs. Feature 6 shows considerable overlap between the classes, but more than one clusters are also noticeable. In general 4 clusters where distinguish. The histograms for Feature 5 and Feature 6 show the presence of several peaks with Gaussian patterns. This suggests a multimodal distribution for both fake and genuine fingerprints. Two peaks where seen in both features for the genuine fingerprint class, and 1 for the fake fingerprint class. The distinct clusters observed in the scatter plot may correspond to different subtypes or variations within the classes, which could be used to improve classification performance.

In general, the first two features exhibit significant overlap and similar statistical properties between the classes, making them less effective for distinguishing between genuine and fake fingerprints. In contrast, the third and fourth features show better discriminative potential due to noticeable differences in means and variances. The last two features, despite their overlap, reveal distinct clusters and multimodal distributions, suggesting that a combination of features or more advanced classification techniques could be necessary for effective fingerprint spoofing detection.

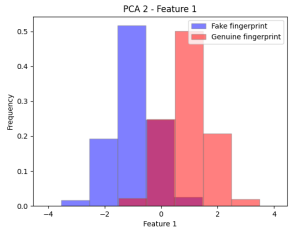
3 PCA and LDA

3.1 PCA implementation and analysis

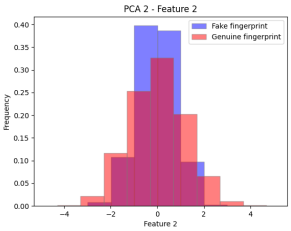
We subsequently implemented the PCA dimension reduction technique, which effectively removes the most correlated dimensions from the data set, as suggested in the laboratory 3. This where the generated histograms from this application of PCA:



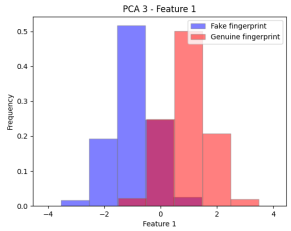
(a) PCA 1 Feature 1



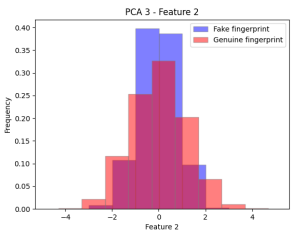
(b) PCA 2 Feature 1



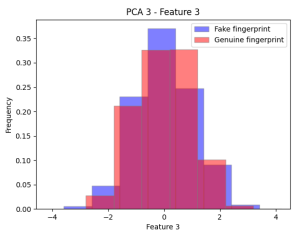
(c) PCA 2 Feature 2



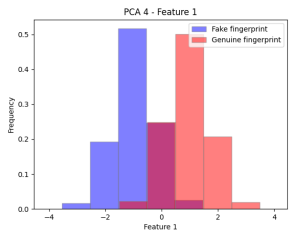
(d) PCA 3 Feature 1



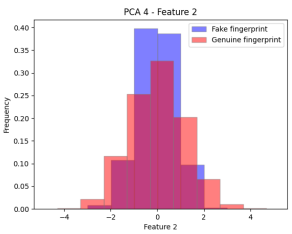
(e) PCA 3 Feature 2



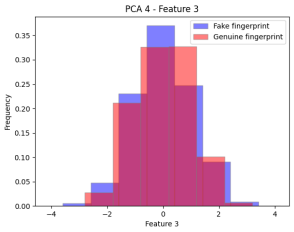
(f) PCA 3 Feature 3



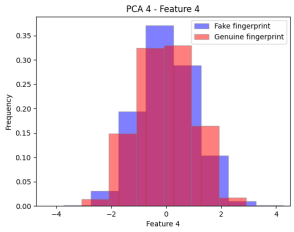
(g) PCA 4 Feature 1



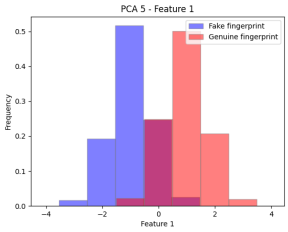
(h) PCA 4 Feature 2



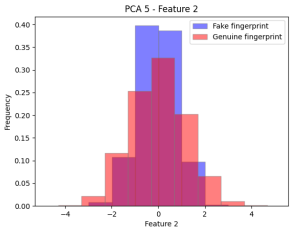
(a) PCA 4 Feature 3



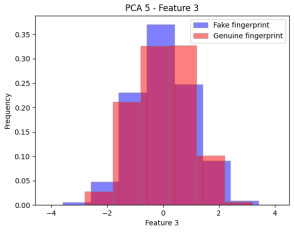
(b) PCA 4 Feature 4



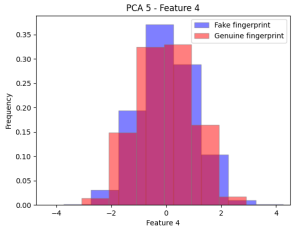
(c) PCA 5 Feature 1



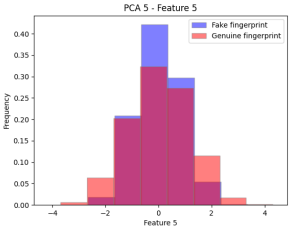
(d) PCA 5 Feature 2



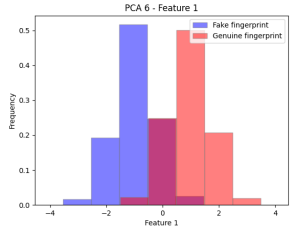
(e) PCA 5 Feature 3



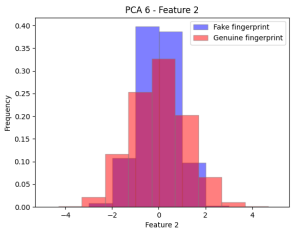
(f) PCA 5 Feature 4



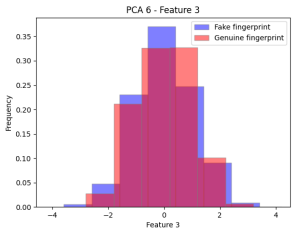
(g) PCA 5 Feature 5



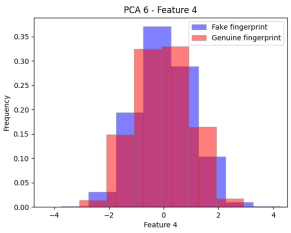
(h) PCA 6 Feature 1



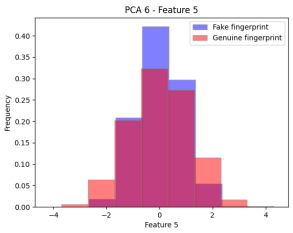
(i) PCA 6 Feature 2



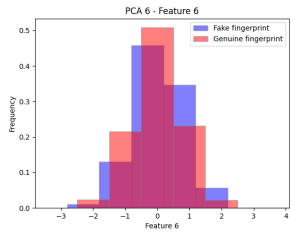
(j) PCA 6 Feature 3



(k) PCA 6 Feature 4



(l) PCA 6 Feature 5



(m) PCA 6 Feature 6

The PCA histograms illustrate the distribution of fake and genuine fingerprints across different principal components (PCA 1 through PCA 6). The analysis reveals that PCA 1 and PCA 2 provide better separation between the classes, particularly in the first feature, while subsequent features (PCA 3 through PCA 6) show increased overlap. This overlapping suggests that PCA alone may not be sufficient for distinguishing the classes without further processing. Although PCA is effective in reducing dimensionality and retaining variance, it does not achieve complete class separation. The overlaps in class distributions indicate the need for additional processing or more dimensions to improve classification accuracy. The first principal component often captures the most variance, leading to a histogram where one class might dominate a certain region, and the other class occupies a different region. This indicates that PCA is effective at separating the classes along the first few components. PCA can reveal sub-clusters within each class, particularly in the components with higher variance. These sub-clusters might indicate different subgroups or patterns within the class, which could be of interest for further analysis.

As an extra, a PCA variability analysis was performed to look how much fraction of the reduce variance of the whole data set is retained. As recommended, this retained variance should be over 97%.

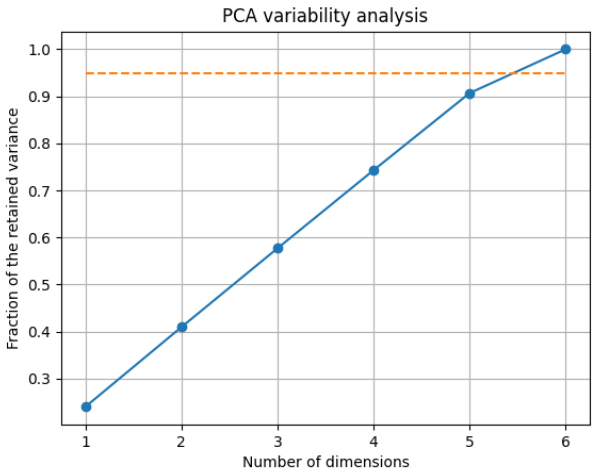


Figure 1: PCA Analysis

This resulted in the accompanying figure, illustrating the variance difference in the data set for each Dimension Reduction. Our findings indicate that we can retain a percentage near 95% of the data set's variance using a maximum of 5 dimensions.

3.2 LDA analysis

LDA was applied also to the data. This was 1 dimensional thanks that we only have 2 classes. A single projection of the data onto the LDA axis. This works by finding a linear combination of features that best separates two or more classes by maximizing the ratio of between-class variance to within-class variance, thus ensuring maximum class separability. We then created the next histogram:

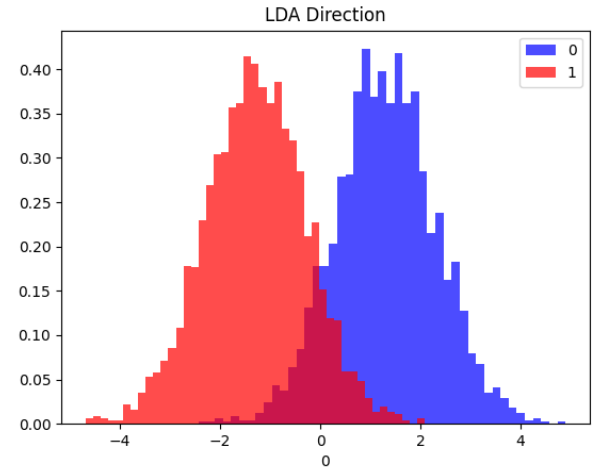


Figure 2: LDA implementation

Compared to the histograms of the 6 PCA features computed earlier, LDA generally finds a direction that maximizes the separation between the classes. This is evident in the histogram where the peaks of the two distributions are more distinct, with less overlap compared to the PCA projections. LDA explicitly aims to maximize the ratio of between-class variance to within-class variance, which typically results in better separation of the classes along the LDA direction. Therefore, the overlap between the classes is often reduced when compared to the PCA projections.

3.3 LDA as classifier

Because this separation of data, LDA works as a classifier by projecting data onto a lower-dimensional space where the classes are most separable, then classifies new data points based on which side of a decision boundary they fall on in this reduced space.

The Linear Discriminant Analysis (LDA) results, both with and without PCA pre-processing and with different thresholds, are summarized below. Without PCA pre-processing and using a threshold of 0.0, the LDA classifier resulted in 186 errors out of 2000 samples, yielding an error rate of 9.3%. When PCA pre-processing with a dimension of $m = 5$ was applied and the threshold remained at 0.0, the number of errors slightly decreased to 185, resulting in an error rate of 9.2%. When the threshold was adjusted to 0.018534376786207174, the number of errors remained at 186 for both with and without PCA pre-processing, maintaining an error rate of 9.3%.

The effectiveness of PCA in conjunction with LDA was evident, as PCA pre-processing with a dimension of $m = 5$ slightly improved the error rate from 9.3% to 9.2% when the threshold was 0.0. However, changing the threshold to 0.018534376786207174 did not affect the error rate, which stayed at 9.3% regardless of PCA pre-processing. This minimal impact of threshold adjustment suggests that the classifier's performance was relatively robust to this specific threshold change.

Overall, LDA as a classifier achieved an error rate of approximately 9.3% without PCA pre-processing. The slight reduction in error rate to 9.2% with PCA pre-processing indicates that PCA can provide a modest improvement in classification performance by reducing dimensionality. However, the threshold adjustment had minimal effect on the error rate, highlighting LDA's robustness. Combining PCA with LDA offers some enhancement in classification tasks, with PCA serving as a useful pre-processing step to boost LDA's effectiveness in class separation.

Lastly for data, an extra analysis was performed.The three correlation heatmaps provided illustrate the relationships between different attributes within the dataset: genuine fingerprints, fake fingerprints, and the entire dataset.

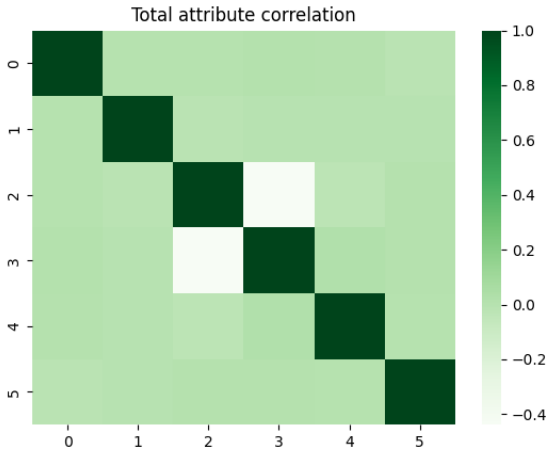


Figure 3: Full Data Set Correlation

The genuine fingerprint attributes tend to have stronger correlations with each other compared to the fake fingerprints, indicating more consistent patterns in the genuine data. The total dataset, on the other hand, shows a more mixed pattern of correlations, reflecting the combined influence of both genuine and fake fingerprints, which contributes to the overall lower and more varied correlations between attributes. These heatmaps provide valuable insights into the consistency and variability of the data, which can guide further analysis or modeling efforts.

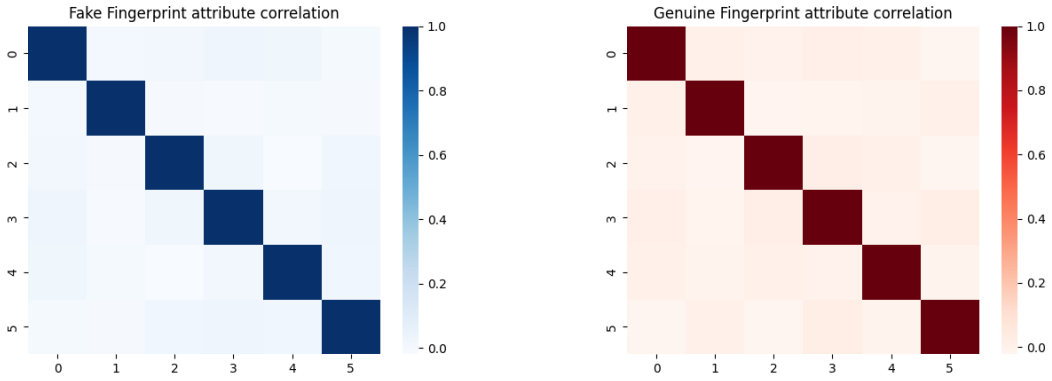
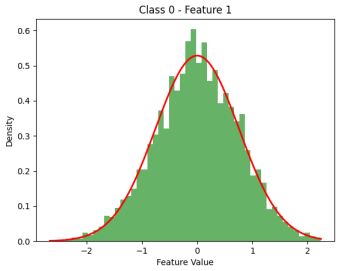


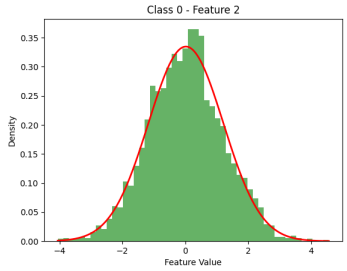
Figure 4: Genuine and fake fingerprint correlations

4 Gaussian Models

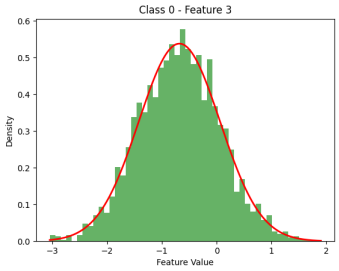
To begin using Gaussian models as a classifier, the first step is to create univariate Gaussian models for the various features corresponding to each class in the project dataset. This involves computing a Maximum Likelihood (ML) estimate for each feature-class combination to determine the parameters of a 1D Gaussian distribution. The resulting distribution densities are as follows:



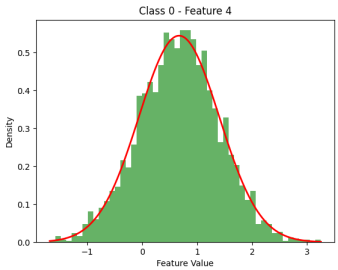
(a) Distribution density Class(0)-Feature(1)



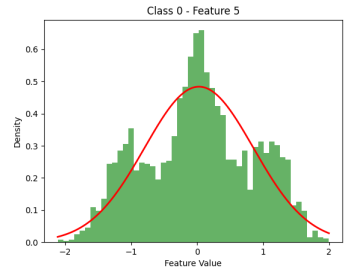
(b) Distribution density Class(0)-Feature(2)



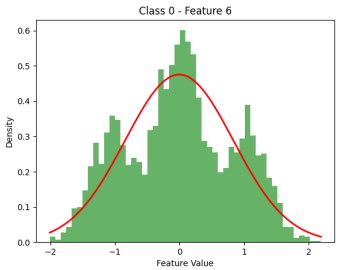
(c) Distribution density Class(0)-Feature(3)



(d) Distribution density Class(0)-Feature(4)



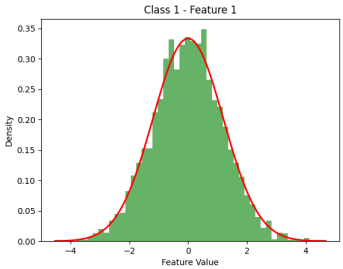
(e) Distribution density Class(0)-Feature(5)



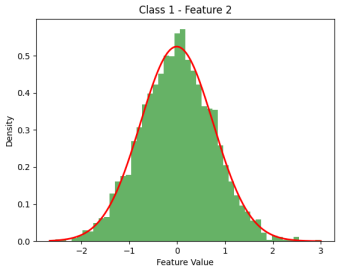
(f) Distribution density Class(0)-Feature(6)

For most features in Class 0, the Gaussian model provides a good fit. Specifically, for Features 1, 2, 3, and 4 the histogram of the data aligns well with the fitted Gaussian density curve. This suggests that these features have distributions that are approximately normal, with the Gaussian model capturing the central tendency and spread of the data. In the analysis of Features 5 and 6, it's evident that the Gaussian model does not provide an accurate fit for the data. Specifically, these features exhibit characteristics that diverge significantly from the assumptions of a Gaussian distribution, which leads to a poor match between the fitted Gaussian density curve and the actual data histogram. This seem

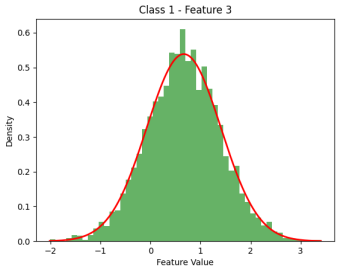
to be 3 Gaussian distribution together in one. Maybe for this later in GMM this can be resolved.



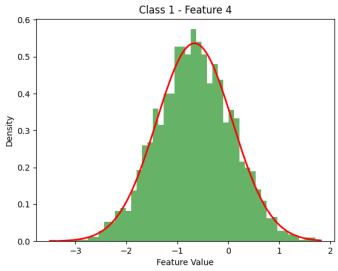
(a) Distribution density Class(1)-Feature(1)



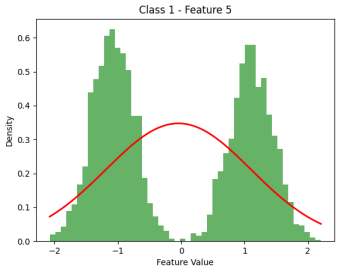
(b) Distribution density Class(1)-Feature(2)



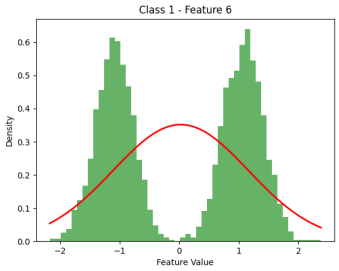
(c) Distribution density Class(1)-Feature(3)



(d) Distribution density Class(1)-Feature(4)



(e) Distribution density Class(1)-Feature(5)



(f) Distribution density Class(1)-Feature(6)

In Class 1, the Gaussian model also fits well for some features, such as Features 1, 2, 3, and 4. The histograms for these features show a good alignment with the Gaussian density curves, suggesting that the data is symmetrically distributed around the mean and follows a normal distribution pattern. However, significant deviations are observed in Features 5 and 6 of Class 1. Both features exhibit a clear bimodal distribution, with two distinct peaks in the histogram. The Gaussian model, which assumes a single peak, fails to capture this bimodality, resulting in a poor fit for these features.

4.1 Gaussians as classifiers

The following section presents the results and analysis from the project work, where various Gaussian models were applied to the dataset. The section will cover the implementation, comparison, and evaluation of different classifiers, including the Multivariate Gaussian (MVG) model, the Naive Bayes Gaussian model, and the Tied Covariance Gaussian model. We will also discuss the impact of feature selection, correlations between features, and the use of Principal Component Analysis (PCA) on the classification accuracy.

First, the data was divided in a 2:1 ratio, consistent with previous procedures (this method will be applied across all subsequent classifier implementations). Assuming equal class priors of $P(C = 1) = P(C = 0) = 1/2$, predictions were generated using the log-likelihood ratios (LLRs). We then calculated the error rates for the three classifiers, with the results as follows:

Table 1: Model Error Rates and Covariance/Correlation Matrices

Model	**Error Rate**
MVG Model	7.00%
Tied Gaussian Model	9.30%
Naive Bayes Gaussian Model	7.20%

The results from the classification models—Multivariate Gaussian (MVG), Tied Gaussian, and Naive Bayes Gaussian—show that the MVG model achieves the lowest error rate of 7.00%, indicating it performs best among the three for this dataset. The Naive Bayes Gaussian model follows closely

with a 7.20% error rate, while the Tied Gaussian model has the highest error rate at 9.30%. This suggests that the MVG model is most effective in capturing the data's distribution, likely due to its ability to model class-specific covariance matrices. The Naive Bayes model's relatively strong performance, despite assuming feature independence, suggests that the dataset's features might be weakly correlated, which aligns with the Naive Bayes assumptions. The Tied Gaussian model, which shares a single covariance matrix across all classes, seems less effective, perhaps because it fails to capture the specific variations within each class.

We then printed the covariance matrices for each class. This where the generated tables:

Table 2: Class 0 - Covariance Matrix

0.60095651	0.00005159	0.01905891	0.01925299	0.01280394	-0.01347216
0.00005159	1.44722543	-0.01613401	-0.01585615	-0.02645291	0.02291398
0.01905891	-0.01613401	0.56534890	-0.00184344	-0.00691446	0.01689283
0.01925299	-0.01585615	-0.00184344	0.54161520	0.00525171	0.01357178
0.01280394	-0.02645291	-0.00691446	0.00525171	0.69606764	0.01584384
-0.01347216	0.02291398	0.01689283	0.01357178	0.01584384	0.68651971

Table 3: Class 1 - Covariance Matrix

1.44809527	-0.01472224	0.00557010	0.01574159	0.01949712	-0.00017668
-0.01472224	0.55339080	-0.01121687	-0.00906473	-0.01465899	0.01634920
0.00557010	-0.01121687	0.55748023	0.02756097	-0.00376966	-0.01459769
0.01574159	-0.00906473	0.02756097	0.56965701	-0.01169834	0.03499319
0.01949712	-0.01465899	-0.00376966	-0.01169834	1.34201767	0.01694541
-0.00017668	0.01634920	-0.01459769	0.03499319	0.01694541	1.30371880

The covariance matrices for both classes reveal that the variances of individual features are significantly larger than the covariances between different features. This is evident from the small off-diagonal values in the covariance matrices, indicating that the features do not vary together strongly. In other words, changes in one feature are not strongly associated with changes in another. The dominance of diagonal elements (variances) over off-diagonal elements (covariances) suggests that each feature's variability is more independent and not influenced much by the other features. This characteristic supports the effectiveness of the Naive Bayes model, as it assumes that features are independent, which seems to be approximately true for this dataset.

After this, to better visualize the strength of covariance the Pearson Correlations was calculated. This correlation values are presented in the following tables:

Table 4: Class 0 - Correlation Matrix

1.00000000	0.00005532	0.03269779	0.03374669	0.01979686	-0.02097438
0.00005532	1.00000000	-0.01783676	-0.01790953	-0.02635601	0.02298825
0.03269779	-0.01783676	1.00000000	-0.00333140	-0.01102236	0.02711550
0.03374669	-0.01790953	-0.00333140	1.00000000	0.00855323	0.02225691
0.01979686	-0.02635601	-0.01102236	0.00855323	1.00000000	0.02291966
-0.02097438	0.02298825	0.02711550	0.02225691	0.02291966	1.00000000

The correlation matrices provide further insight into the relationships between features. The off-diagonal elements in these matrices are mostly close to zero, indicating weak correlations between features. Correlation measures the strength and direction of the linear relationship between pairs of features, normalized by their variances. The near-zero values suggest that knowing the value of one feature gives little information about the value of another, giving the idea that the features are largely independent. This weak correlation helps explain why the Naive Bayes model, which assumes feature independence, performs relatively well on this dataset.

Table 5: Class 1 - Correlation Matrix

1.00000000	-0.01644597	0.00619940	0.01733178	0.01398597	-0.00012859
-0.01644597	1.00000000	-0.02019486	-0.01614479	-0.01701018	0.01924814
0.00619940	-0.02019486	1.00000000	0.04890722	-0.00435822	-0.01712291
0.01733178	-0.01614479	0.04890722	1.00000000	-0.01337945	0.04060549
0.01398597	-0.01701018	-0.00435822	-0.01337945	1.00000000	0.01281094
-0.00012859	0.01924814	-0.01712291	0.04060549	0.01281094	1.00000000

To assess how different features impact the performance of Gaussian classifiers, we analyzed the Gaussian density plots discussed earlier. Features 1 through 4 exhibit strong Gaussian characteristics, while features 5 and 6 deviate from this distribution. This suggests that the first four features are well-suited for Gaussian-based classification. To test this, we excluded the last two features and recorded the results in the following table. Additionally, we computed results using Tied Gaussian and MVG classifiers for paired features (1 and 2, 3 and 4, 5 and 6). We also included the outcomes after applying PCA for preprocessing. The final table is presented below:

Table 6: Error Rates for Different Feature Sets and PCA

Feature Set	**Error Rate**
MVG Model (first 4 features)	7.95%
Tied Gaussian Model (first 4 features)	9.50%
Naive Bayes Gaussian Model (first 4 features)	7.65%
MVG Model (features 1-2)	36.50%
Tied Gaussian Model (features 1-2)	49.45%
MVG Model (features 3-4)	9.45%
Tied Gaussian Model (features 3-4)	9.40%
MVG Model with PCA(5)	7.10%
Tied Gaussian Model with PCA(5)	9.30%
Naive Bayes Gaussian Model with PCA(5)	8.75%

When analyzing the models using only the first four features, there is a slight increase in error rates for all three models. The MVG model's error rate rises to 7.95%, the Tied Gaussian model's error rate increases to 9.50%, and the Naive Bayes model's error rate increases to 7.65%. This suggests that the last two features, while not individually dominant, contribute useful information that enhances classification accuracy. Removing these features slightly degrades performance, indicating that they provide complementary information that helps the models better distinguish between classes.

Further analysis using only features 1-2 and features 3-4 separately reveals stark differences in the models' effectiveness. When using only features 1-2, the error rates are significantly higher—36.50% for the MVG model and 49.45% for the Tied Gaussian model. This likely occurs because these features have similar means but different variances across classes, making it difficult for the models to effectively separate the classes. In contrast, using only features 3-4 results in much lower error rates of around 9.45% for both models. This suggests that features 3 and 4 are more informative and better suited for distinguishing between classes, likely due to their more pronounced mean differences across classes.

Finally, the application of PCA as a pre-processing step did not improve model performance; in fact, it slightly increased the error rates. The MVG model's error rate increased to 8.05%, the Naive Bayes model's error rate rose to 8.85%, and the Tied Gaussian model's error rate remained roughly the same at 9.25%. This suggests that PCA might have removed some critical information or altered the data structure in a way that was less favorable for classification (maybe as it was seen in the variability analysis). While PCA is useful for reducing dimensionality, in this case, the original features seem to have been more effective for the classification task without dimensionality reduction. Overall, the MVG model provided the best accuracy on the validation set, both with and without PCA.

To continue as suggested in lab 7, we will analyze the performance of the MVG classifier in different applications.

Best PCA Setup: m=6 with DCF=0.263

Based on the analysis of the MVG classifier and its variants for different applications, the results

Table 7: Classifier Performance Comparison

Classifier	Application	DCF	min DCF
MVG	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 1.0$	0.140	0.130
	$\pi = 0.9, c_{fn} = 1.0, c_{fp} = 1.0$	0.400	0.342
	$\pi = 0.1, c_{fn} = 1.0, c_{fp} = 1.0$	0.305	0.263
	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 9.0$	0.140	0.263
	$\pi = 0.5, c_{fn} = 9.0, c_{fp} = 1.0$	0.140	0.342
Naive	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 1.0$	0.144	0.131
	$\pi = 0.9, c_{fn} = 1.0, c_{fp} = 1.0$	0.389	0.351
	$\pi = 0.1, c_{fn} = 1.0, c_{fp} = 1.0$	0.302	0.257
	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 9.0$	0.144	0.257
	$\pi = 0.5, c_{fn} = 9.0, c_{fp} = 1.0$	0.144	0.351
Tied	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 1.0$	0.186	0.181
	$\pi = 0.9, c_{fn} = 1.0, c_{fp} = 1.0$	0.463	0.442
	$\pi = 0.1, c_{fn} = 1.0, c_{fp} = 1.0$	0.406	0.363
	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 9.0$	0.186	0.363
	$\pi = 0.5, c_{fn} = 9.0, c_{fp} = 1.0$	0.186	0.442

Table 8: PCA Analysis

Classifier	Application	DCF	min DCF
MVG	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 1.0$	0.140	0.130
	$\pi = 0.9, c_{fn} = 1.0, c_{fp} = 1.0$	0.400	0.342
	$\pi = 0.1, c_{fn} = 1.0, c_{fp} = 1.0$	0.305	0.263
	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 9.0$	0.140	0.263
	$\pi = 0.5, c_{fn} = 9.0, c_{fp} = 1.0$	0.140	0.342
Naive	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 1.0$	0.178	0.173
	$\pi = 0.9, c_{fn} = 1.0, c_{fp} = 1.0$	0.451	0.436
	$\pi = 0.1, c_{fn} = 1.0, c_{fp} = 1.0$	0.392	0.353
	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 9.0$	0.178	0.353
	$\pi = 0.5, c_{fn} = 9.0, c_{fp} = 1.0$	0.178	0.436
Tied	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 1.0$	0.186	0.181
	$\pi = 0.9, c_{fn} = 1.0, c_{fp} = 1.0$	0.463	0.442
	$\pi = 0.1, c_{fn} = 1.0, c_{fp} = 1.0$	0.406	0.363
	$\pi = 0.5, c_{fn} = 1.0, c_{fp} = 9.0$	0.186	0.363
	$\pi = 0.5, c_{fn} = 9.0, c_{fp} = 1.0$	0.186	0.442

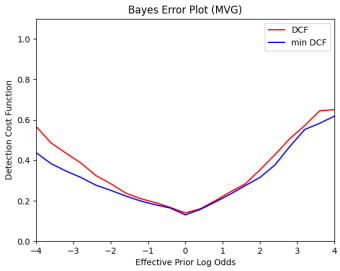
demonstrate distinct behaviors depending on the specific application scenario. The applications are represented in terms of effective prior probabilities, which show different assumptions about the likelihood of genuine versus fake samples. For example, an application with a uniform prior and equal costs ($\pi = 0.5$, cfn = 1.0, cfp = 1.0) represents a balanced scenario, whereas an application with a higher prior probability for genuine samples ($\pi = 0.9$, cfn = 1.0, cfp = 1.0) assumes most results are genuine fingerprints. Also, an application with a higher prior for fake samples ($\pi = 0.1$, cfn = 1.0, cfp = 1.0) suggests most results are fake. Applications with asymmetric costs, such as ($\pi = 0.5$, cfn = 1.0, cfp = 9.0), heavily penalize the acceptance of impostors, while ($\pi = 0.5$, cfn = 9.0, cfp = 1.0) minimizes the cost of rejecting genuine users.

For each application, the optimal Bayes decisions were computed for the MVG, Naive Bayes, and Tied covariance classifiers, both with and without Principal Component Analysis (PCA). The number of components(m) for PCA was decided by testing one single application with all PCAs and mantaining the one that achivied better results. As before m=6 or in other words no PCA gives the better results. Nonetheless, both results are presented. The results show that the MVG classifier consistently performs well across all applications, achieving low Detection Cost Function (DCF) values. The Naive Bayes classifier also performs well, albeit with slightly higher DCF values compared to the MVG classifier. The Tied covariance classifier, while useful, generally exhibits the highest DCF values among the three models.

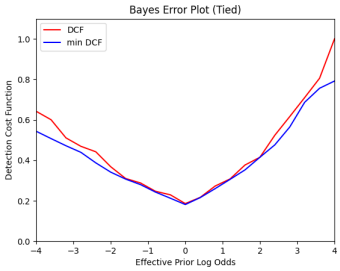
The PCA analysis revealed that the best setup was with $m = 6$ components, yielding a DCF of 0.263. For this PCA configuration, the MVG classifier continued to demonstrate strong performance with the lowest DCF values across various applications. The Naive and Tied classifiers, while competitive, showed slightly worse performance and larger calibration losses compared to the MVG classifier. The Bayes error plots, which display the performance of the classifiers over a range of effective prior log odds, confirmed that the MVG classifier consistently outperforms the others, maintaining robustness and good calibration across different scenarios.

The MVG classifier is the best-performing model among those considered, providing consistent and reliable performance across various applications with minimal calibration loss. The PCA setup with $m = 6$ components proved to be optimal, but non beneficial as preprocessing. Although the Naive and Tied classifiers have generally good results, they generally exhibit larger calibration losses, making them less effective than the MVG classifier in most scenarios.

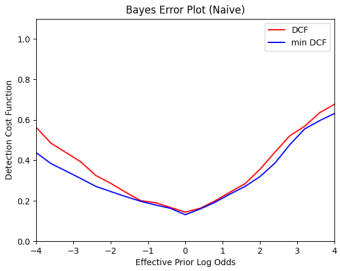
The results for the different prior log-odds in the Bayes plot, with the application of $\pi = 0.1$ over the range $[-4, +4]$, for both the minimum and actual DCF are as follows:



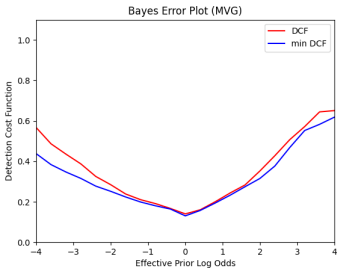
(a) Bayes error plot MVG



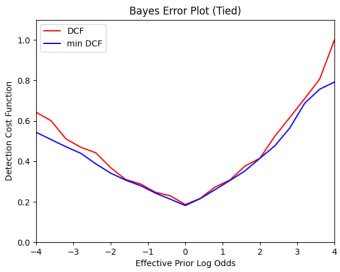
(b) Bayes error plot Tied Gaussian



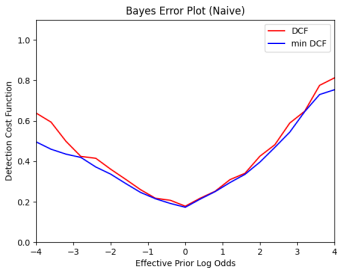
(c) Bayes error plot Naive Bayes



(d) Bayes error plot MVG PCA m=6



(e) Bayes error plot Tied Gaussian PCA m=5



(f) Bayes error plot Naive Bayes PCA m=5

The Bayes Error plots display both the minimum DCF (minDCF), representing the lowest possible DCF achievable with an optimal decision threshold, and the actual DCF, which indicates the classifier's performance without further calibration. In these plots, the minDCF is plotted in blue, while the actual DCF is shown in red. A smaller gap between these lines suggests better calibration of the model. Without PCA, the MVG classifier generally shows superior performance, evidenced by lower minDCF values compared to Naive and Tied Gaussian classifiers. With PCA applied, the performance of Naive and Tied classifiers improves, sometimes approaching that of the MVG model.

Regarding calibration, the Naive Gaussian classifier typically exhibits a larger gap between DCF and minDCF than the MVG, indicating it may be less well-calibrated. The Tied Gaussian classifier performs similarly to the MVG model when PCA is used, showing competitive results across various applications. Overall, the MVG classifier demonstrates the best calibration, with the smallest difference between actual DCF and minDCF across different scenarios.

The consistency of minimum DCF values across various applications suggests that the relative rankings of these models remain stable. The actual DCF values also indicate that models like the MVG classifier maintain consistent performance, with minimal calibration loss, across different settings. The MVG classifier generally outperforms others, particularly in high-security scenarios where false positives carry a high cost. In contrast, the Naive Bayes model, especially without PCA, tends to show higher DCF values, suggesting greater sensitivity to data dimensionality and correlations.

5 Logistic and Quadratic Regression Models/ Discriminative approaches

To analyze the logistic and quadratic regression models introduced in Lab 8, we considered both standard and prior-weighted models. These models were constructed in logarithmic space using various values of λ generated by 'numpy.logspace(-4, 2, 13)'. The corresponding actual DCF and minDCF values are presented. All analyses were conducted in the primary application setting with $\pi = 0.1$.

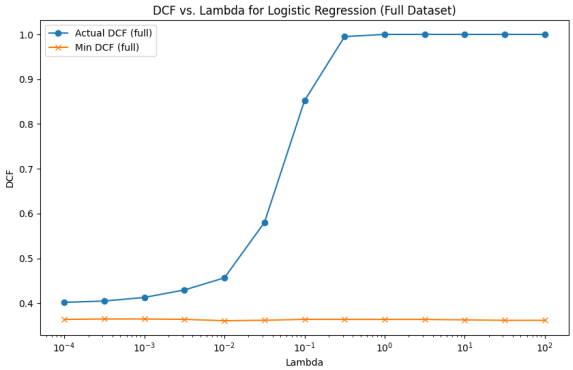


Figure 5: Full Data Logistic Regression

The analysis of the binary logistic regression model, using various values of the regularization coefficient (λ), reveals significant insights into how regularization affects model performance, particularly in terms of the Decision Cost Function (DCF). When using the full dataset, as shown in the first graph, it is evident that as λ increases, the Actual DCF also increases sharply after a certain threshold (around 10^{-2}). This suggests that with a large dataset, applying too much regularization leads to underfitting, where the model becomes overly simplistic and fails to capture the underlying data distribution effectively. In contrast, the minimum DCF remains relatively stable, indicating that while the model's calibration (as reflected by Actual DCF) is adversely affected by high regularization, the theoretical optimal decision performance (minDCF) is less sensitive to changes in λ.

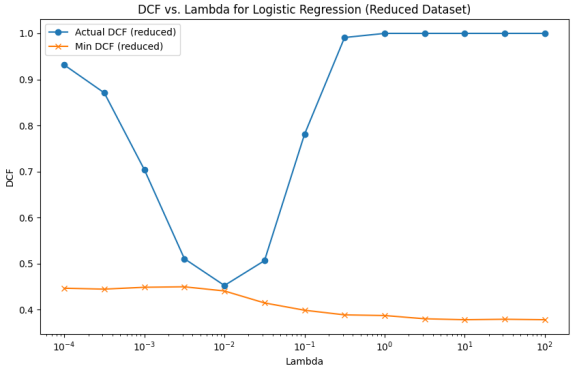


Figure 6: Reduced dataset (1 of 50 model samples) Logistic Regression

When analyzing the reduced dataset, as shown in the second graph, a different pattern emerges. Here, the Actual DCF initially decreases as λ increases, reaching its lowest point around 10^{-2} , before increasing again. This behavior indicates that with fewer training samples, some degree of regularization is beneficial as it helps prevent overfitting, which is more likely in smaller datasets. However, similar to the full dataset, excessive regularization eventually leads to underfitting, where the model loses its ability to generalize well, resulting in a higher Actual DCF.

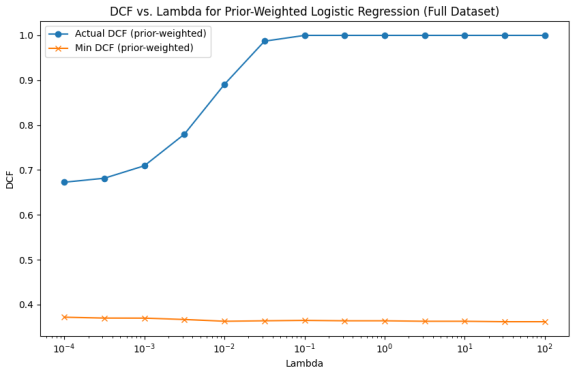


Figure 7: Prior Weighted Logistic Regression

The analysis with the prior-weighted logistic regression model, depicted in the graph, shows a similar trend to the non-weighted model when using the full dataset. The Actual DCF increases with larger λ values, indicating that regularization still leads to underfitting in this scenario. Interestingly, the prior-weighted model does not exhibit significant improvements in Actual DCF compared to the non-weighted model, suggesting that the choice of prior weighting might not provide substantial advantages in this particular application.

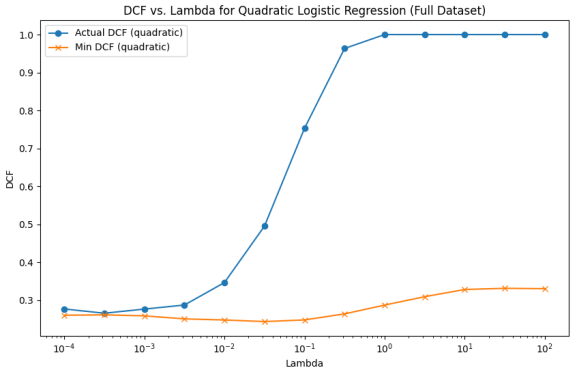


Figure 8: Full dataset Quadratic Regression

When applying quadratic logistic regression, as shown in the graph, the pattern of the DCF metrics as a function of λ remains consistent with the linear model. The Actual DCF increases with larger λ values, reinforcing the idea that excessive regularization, even in the presence of quadratic features, leads to underfitting. This consistency across different model types suggests that while feature expansion (from linear to quadratic) might change the model's complexity, the fundamental impact of regularization on DCF metrics remains similar.

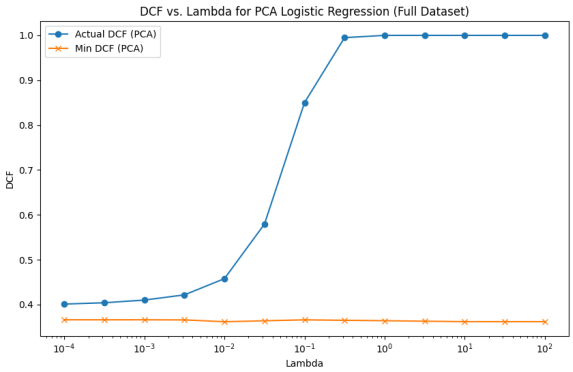


Figure 9: PCA with m=5 Logistic Regression

The PCA graph offers an interesting comparison to the standard logistic regression model without Principal Component Analysis (PCA). In this graph, the trends observed for both Actual DCF and minDCF as a function of the regularization parameter (λ) are similar to those seen in the non-PCA model. the PCA graph reinforces the idea that regularization plays a critical role in balancing the trade-off between overfitting and underfitting. Based on the results, PCA does not seem to significantly alter the impact of regularization on the model's performance in terms of DCF. Actual DCF with higher λ values still indicates a loss of the probabilistic interpretation of the scores, leading to underfitting. Even with PCA applied, the effects of regularization remain prominent, emphasizing the need to carefully select an appropriate λ value that minimizes Actual DCF while maintaining good generalization performance.

This is a table with the best results of each Logistic regression graphs:

Type	**Best Actual DCF**	**Min DCF for Best Actual**	**Best Lambda**	**Best Min DCF**
Full	0.4021	0.3640	0.0001	0.3611
Reduced	0.4526	0.4407	0.01	0.3783
Prior-weighted	0.6728	0.3721	0.0001	0.3620
Quadratic	0.2656	0.2612	0.000316	0.2436
PCA = 5	0.4011	0.3661	0.0001	0.3618

Table 9: Best Actual DCF for different configurations

6 Support Vector Machines

In this section, we will calculate three different types of Support Vector Machines (SVMs), starting with the linear model. Unlike other models, SVMs operate differently by focusing on creating a hyperplane that separates data into classes. However, rather than providing a probabilistic interpretation of the score, the SVM optimizes by maximizing the minimum distance between the hyperplane and the classes, known as the margin. To reduce training time, we set K=1 and calculated the Detection Cost Function (DCF) using various values of C within the log space of (-5, 0, 11). The results for the Linear SVM are shown in the following graph:

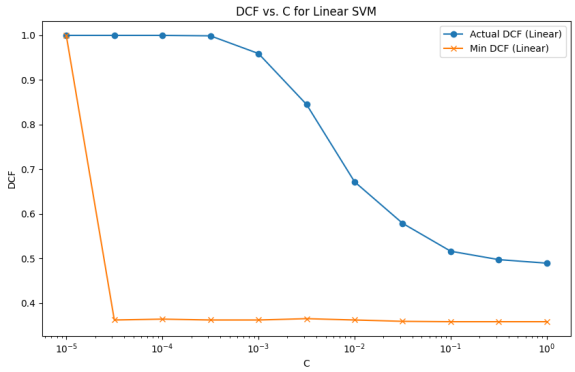


Figure 10: Linear SVM with various C values results for minDCF and actual DCF

This Linear SVM, the graph depicting the relationship between the Decision Cost Function (DCF) and the regularization parameter C shows that as C increases, the Actual DCF decreases significantly. This indicates improved performance with weaker regularization, as lower values of C (which imply stronger regularization) result in higher Actual DCF, suggesting poorer model performance. Interestingly, the Min DCF remains low and stable across all values of C, which suggests that the minimum achievable cost is relatively unaffected by the strength of regularization. This observation implies that while the regularization coefficient C significantly influences the Actual DCF, it has little impact on the Min DCF. The proximity of Actual DCF to Min DCF at higher C values also indicates that the model's scores are well-calibrated. Overall, the Linear SVM performs well with less regularization and shows comparable results to other linear models when optimally tuned.

This next graph corresponds to Polynomial kernel with d=2 and c=1 for simplicity:

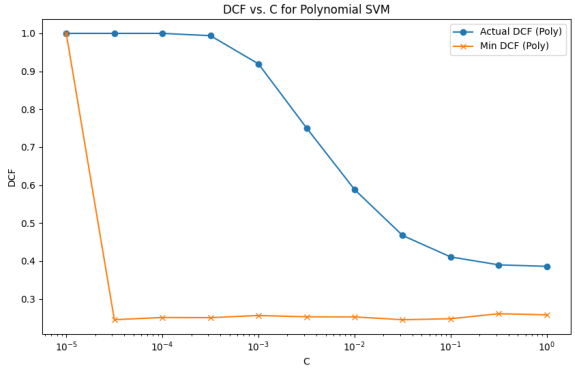


Figure 11: Polynomial SVM with various C values results for minDCF and actual DCF

Moving to the Polynomial SVM with a degree of 2, the graph similarly shows a significant decrease in Actual DCF as C increases, followed by stabilization. Like the Linear SVM, the Min DCF remains consistently low, indicating that the minimum achievable cost is stable across different regularization strengths. The behavior of the Polynomial SVM aligns closely with that of the Linear SVM, demonstrating that Polynomial SVM with a degree of 2 offers good performance, particularly with less regularization. This performance is consistent with the results obtained from logistic regression and Multivariate Gaussian (MVG) models, particularly in terms of Min DCF, confirming that the Polynomial SVM can effectively model the dataset's characteristics.

Lastly for SVM, the RBF SVM is more complex with the inclusion of the γ parameter, which controls the influence of individual training samples. A grid search aproach was used with $\gamma = \{e^{-4}, e^{-3}, e^{-2}, e^{-1}\}$ and C used in numpy.logspace(-3,2,11) to avoid excessive time. This was the result:

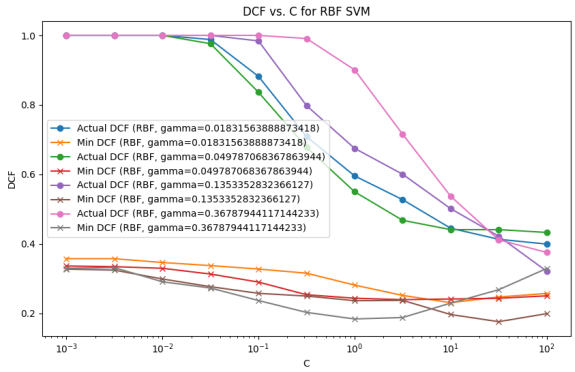


Figure 12: RBF SVM with various C and γ values results for minDCF and actual DCF

The DCF vs. C graph for the RBF SVM shows multiple lines corresponding to different γ values, illustrating the interaction between γ and C in determining model performance. For each γ, the Actual DCF decreases as C increases, though the rate of decrease varies depending on the γ value. Notably, the Min DCF remains low across different C values, similar to the Linear and Polynomial SVMs. The RBF SVM's performance is highly dependent on the appropriate tuning of both γ and C, with certain combinations yielding notably lower Actual DCF. This indicates that RBF SVM can capture more complex patterns in the data due to the flexibility offered by the γ parameter. Compared to the Linear and Polynomial SVMs, the RBF SVM offers the potential for better performance.

The best results of SVM are the following:

SVM Type	**Best Actual DCF**	**Best C**	**Best Gamma**	**Min DCF for Best Actual DCF**	**Best Min DCF**
Linear SVM	0.4894	1.0	N/A	0.3582	0.3582
Polynomial SVM	0.3861	1.0	N/A	0.2582	0.2455
RBF SVM	0.3206	$10^{1.5}$	0.1353	0.1987	0.1755

Table 10: Best SVM configurations

7 Gaussian Mixture Models

Finally, we utilized the Gaussian Mixture Model (GMM) classifier, which is designed to associate datasets with a combination of one or more Gaussian distributions. We varied the number of Gaussian components, training full and diagonal covariance models with up to 32 components, and selected the best models based on the minimum Detection Cost Function (DCF) on the validation set. The next results are for full and diagonal models:

As observed, the plot comparing full and diagonal covariance models reveals that both minimum DCF (minDCF) and actual DCF (actDCF) values generally decrease as the number of components increases, particularly around 4 to 8 components. However, after this point, the DCF values begin to rise, which indicates that while increasing the number of components initially improves model performance, it eventually leads to overfitting, especially in the full covariance model. The diagonal covariance model often outperforms the full covariance model, especially when we look at actual DCF.When it comes to selecting the best model, the diagonal covariance GMM, particularly with 4 components, seems to be a more reliable choice. This model consistently shows lower or comparable DCF values compared to the full covariance model, indicating that it may be less prone to overfitting and more robust overall. For this specific application, diagonal covariance GMMs appear to strike a good balance between simplicity and performance.

The best results of GMM are the following:

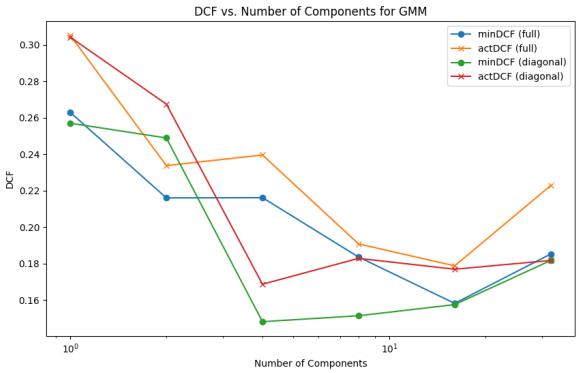


Figure 13: GMM implementation for Full and Diagonal models

Covariance Type	**Best actDCF**	**Components for Best actDCF**	**Min DCF for Best actDCF**	**Best Min DCF (num components, minDCF, actDCF)**
Full Covariance	0.1766	16	0.1631	(16, 0.1631, 0.1766)
Diagonal Covariance	0.1686	4	0.1481	(8, 0.1462, 0.1808)

Table 11: Best GMM configurations

8 Best Classifier models

For each of the best methods (GMM,, logistic regression, and SVM) (we ignore MVG for bad results) this is the table with the best performing candidates:

Model Type (Configuration)	**Best Actual DCF**	**Min DCF for Best Actual DCF**
Quadratic Logistic Regression (Lambda: 0.0003)	0.2656	0.2436
Full Logistic Regression (Lambda: 0.0001)	0.402	0.361
Diagonal Covariance GMM (Components: 4)	0.1686	0.1481
Full Covariance GMM (Components: 16)	0.1766	0.1631
RBF SVM (C: 1.0, Gamma: 0.1353)	0.3206	0.1987

Table 12: Best Configurations for Actual DCF Ordered from Best to Worst

The Diagonal Covariance GMM (Components: 4) emerges as the most promising method for the given application. It achieves a relatively low actual DCF of 0.1686, which is only slightly above its minimum DCF of 0.1481. This small difference suggests that the model is well-calibrated and effectively balances performance with minimal overfitting. While the Full Covariance GMM (Components: 16) also shows good performance with a low actual DCF of 0.1766, the Diagonal Covariance GMM outperforms it in terms of both actual and minimum DCF values.

Other models, such as the Quadratic Logistic Regression and RBF SVM, show higher actual DCFs and larger discrepancies between their actual and minimum DCFs, indicating potential calibration issues or less robustness in this context. Therefore, the Diagonal Covariance GMM, with its strong balance between low actual and minimum DCFs, appears to be the most reliable and effective choice for this application.

Now, we consider alternative application of this good performing models. For this Bayes error plot where calculated for each method.

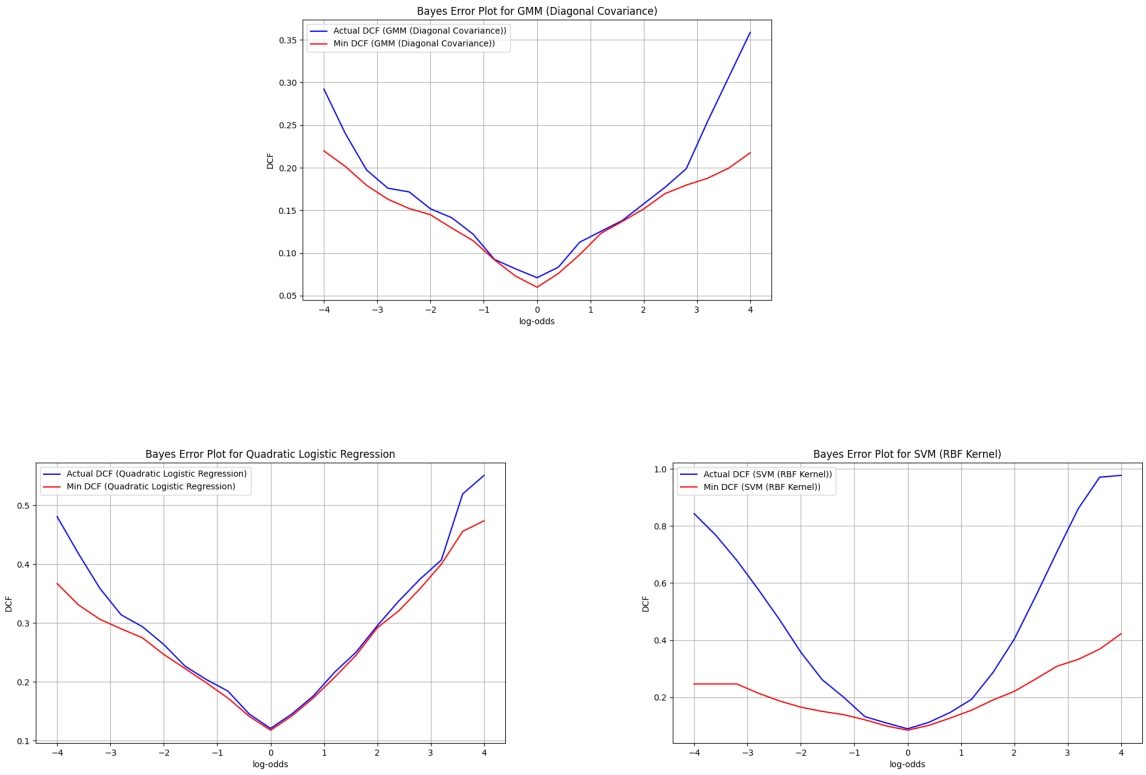


Figure 14: Bayes error plots of best performing models

The analysis of the Bayes Error Plots for Quadratic Logistic Regression, GMM (Diagonal Covariance), and SVM (RBF Kernel) reveals that in terms of minimum DCF, the relative ranking of the systems is consistent across the range of operating points, with GMM generally performing best, followed by Quadratic Logistic Regression, and then SVM. The Quadratic Logistic Regression model appears well-calibrated, as its actual DCF closely follows the minimum DCF across most log-odds values, indicating stable performance. GMM also demonstrates good calibration, though with some fluctuations. However, the SVM (RBF Kernel) displays significant miscalibration, especially at the extremes of the log-odds range, where the actual DCF diverges noticeably from the minimum DCF. This suggests that while SVM might perform well within a specific range, it could lead to poor decisions in applications where the decision threshold isn't fixed. Overall, Quadratic Logistic Regression and GMM are reliable across various scenarios, while SVM requires careful calibration to avoid potential issues in more dynamic applications.

9 Calibration and fusion

In this section, the calibration of the optimal models is carried out. We employ Logistic Regression for this task, as it operates as a posterior log-likelihood ratio estimator. By subtracting the theoretical threshold, we can obtain the calibrated score. This approach is expected to yield improvements, and these calibrations will be assessed accordingly. To enhance the results, a 5-fold cross-validation (K-fold) method was applied. Additionally, a series of prior probabilities, ranging from 0.1 to 0.9, were tested to identify which yielded the best scores. Based on this analysis, a subsequent generalization beyond the K-folds was performed.

Additionally, the calibrated best scores from each model (GMM, SVM, and LR) were fused in all possible combinations to explore the potential for improved performance. This fusion was achieved using weighted Logistic Regression to combine the scores, allowing for a more nuanced integration of the model outputs. Furthermore, the optimal training prior was determined during this process to ensure the best possible calibration of the fused models. The results are showned in the next table and graphs.

Model	Best Training Prior	act DCF
Quadratic Logistic Regression (Lambda: 0.0003)	0.9	0.2561
Diagonal Covariance GMM (Components: 4)	0.7	**0.1521**
RBF SVM (C: 1.0, Gamma: 0.1353)	0.9	0.1775
Fusion of the three models	0.8	0.1657
Fusion of Quadratic Logistic Regression and GMM	0.1	0.1586
Fusion of Quadratic Logistic Regression and RBF SVM	0.1	0.1903
Fusion of GMM and RBF SVM	0.9	0.1637

Table 13: Best Training Priors and DCF for Various Models and Fusions

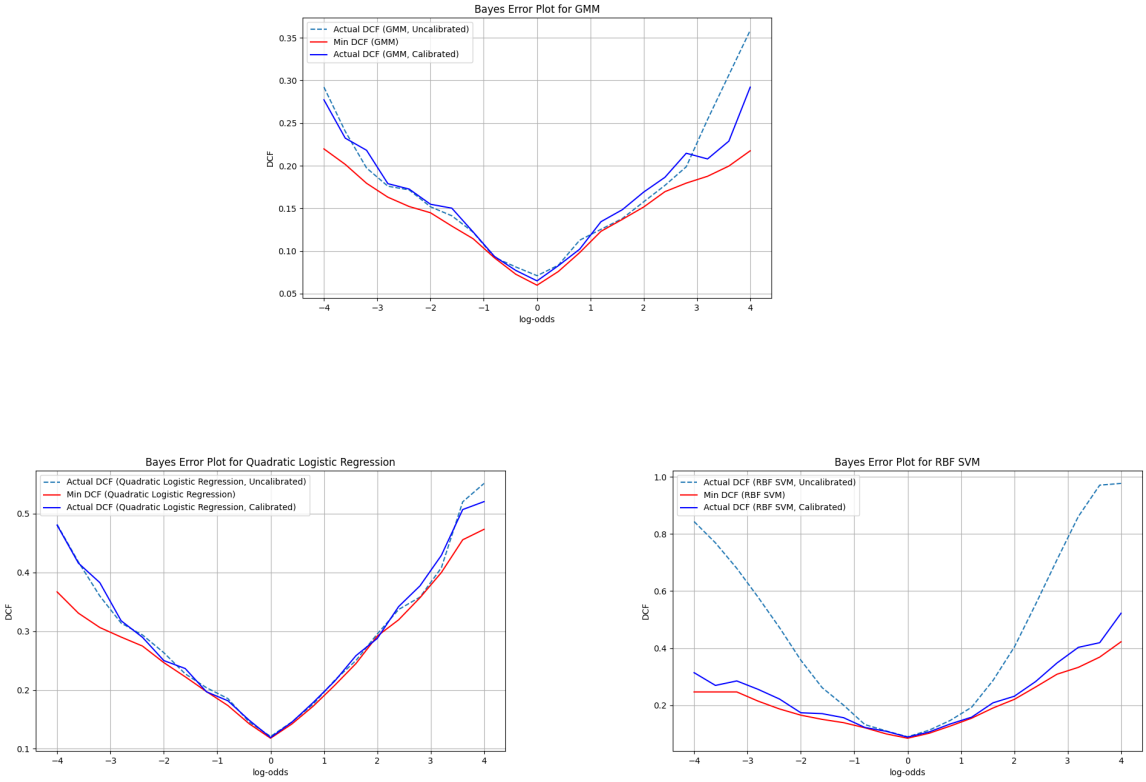
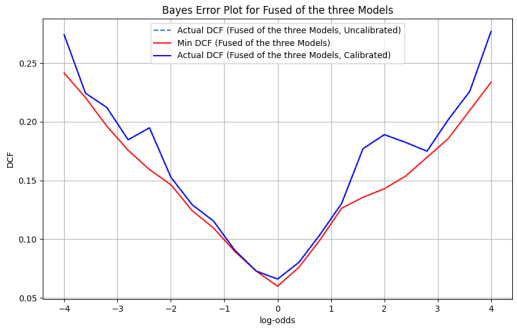


Figure 15: Calibrated Bayes error plots of best performing models

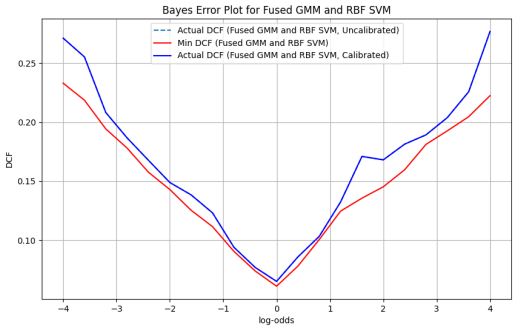
When evaluating the calibration for each model, it is clear that the GMM model performs the best. For Quadratic Logistic Regression, the best training prior is 0.9, and the actual DCF is 0.2561. While calibration improves the model's performance, as shown in the Bayes plot, the DCF remains higher than in the other models, suggesting that calibration is less effective here. RBF SVM, with a best training prior of 0.9 and actual DCF of 0.1775, shows more significant improvement after calibration, especially around the target application, confirming the effectiveness of the calibration process for this model.

However, GMM stands out with a best training prior of 0.7 and an actual DCF of 0.1521, the lowest among all models. The calibration process for GMM proves to be highly effective, as its actual DCF comes very close to the minimum DCF across a wide range of log-odds values, as observed in the Bayes plot. This confirms that GMM is well-calibrated for various applications, making it the most optimal model in terms of calibration performance.

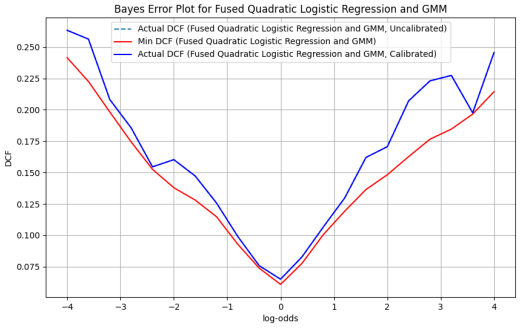
The results of fusion look in general well calibrated, but with some noise. Fusion of multiple models, while typically improving performance, did not outperform the GMM model. The fusion of Quadratic Logistic Regression, GMM, and RBF SVM has a best training prior of 0.8 and achieves an actual DCF of 0.1657. While this is an improvement over most individual models, it does not surpass



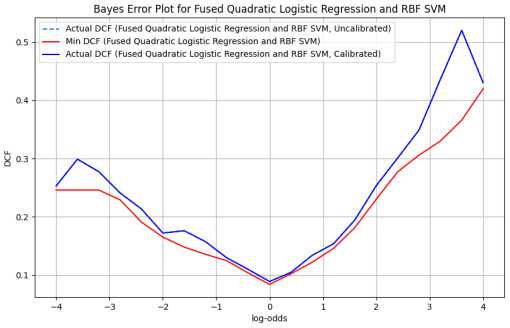
(a) Fusion of the 3 models



(b) Fusion of Diagonal GMM and RBF SVM



(c) Fusion Quadratic Logistic Regression and Diagonal GMM



(d) Fusion Quadratic Logistic regression and RBF SVM

Figure 16: Bayes error plots: Fusion of best performing models

the GMM model's performance. Similarly, the fusion of Quadratic Logistic Regression and Diagonal GMM achieves an actual DCF of 0.1586, which is better than all the other models except for GMM alone. This suggests that while fusion can enhance performance, in this case, it is not able to deliver a better actual DCF than GMM, which remains the best model.

Given the results of the calibration and fusion evaluations, GMM is selected as the best final model. Its best training prior of 0.7 leads to an actual DCF of 0.1521, which is the lowest DCF among all models and fusion combinations. This demonstrates that GMM is well-calibrated, providing consistent and reliable performance across a variety of applications. The Bayes error plot for GMM shows that it performs optimally after calibration, with minimal deviation from the minimum DCF.

10 Evaluation

After comparing all the models, the Diagonal GMM with 4 components was selected as the most suitable candidate for implementation on the evaluation dataset. This was decided thanks of the better actual DCFs performance, but also thanks that in the begining the visualization of the data show multiple Gaussian distributions (that could be a reason for the good scores). For this evaluation, the top three models were applied to the new data, and their resulting scores were optimized using the calibration parameters (w and b) derived in the previous section. The results for all the best-performing models are presented in the following table and graphs.

Initially, the GMM (Diagonal Covariance Gaussian Mixture Model) was chosen as the primary model for testing due to its ability to model data distributions effectively. The GMM achieved a minDCF of 0.1947 and an actDCF of 0.2043, with a slight improvement after calibration to a calibrated actDCF of 0.2017. The Bayes error plot for the GMM showed that the actual DCF closely followed the minimum DCF across different log-odds values, indicating effective calibration.

When comparing the GMM's performance to other models, it became clear that the GMM outperformed the alternatives. The Quadratic Logistic Regression model had a significantly higher minDCF

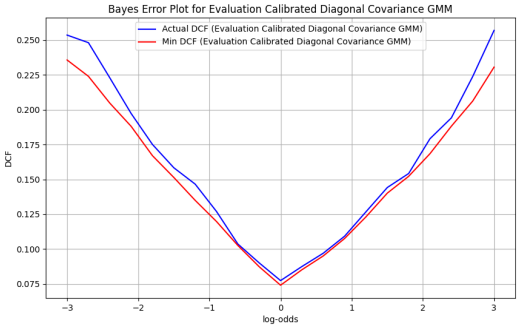
Model	minDCF	actDCF	Calibrated actDCF
Quadratic Logistic Regression	0.3549	0.3632	0.3651
Diagonal Covariance GMM	0.1947	0.2043	0.2017
RBF SVM	0.2622	0.4056	0.2761
Fusion Calibrated Models			
Fused (Quad + GMM + SVM)	0.1932	0.2129	N/A
Fused (Quad + GMM)	0.2014	0.2070	N/A
Fused (Quad + SVM)	0.2569	0.3746	N/A
Fused (GMM + SVM)	0.1885	0.2093	N/A

Table 14: Evaluation of Calibrated Models and Fused Models

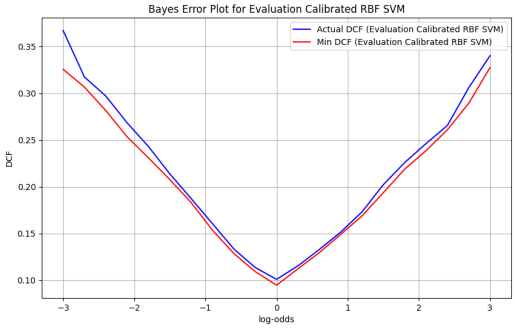
of 0.3549 and actDCF of 0.3632, with a calibrated actDCF of 0.3651, making it less effective than the GMM. The RBF SVM had an initial minDCF of 0.2622 and an actDCF of 0.4056, but calibration improved its actDCF to 0.2761. However, even after calibration, the GMM remained the better-performing model overall.

The real improvement came with model fusion. The GMM + SVM fusion achieved the best results with a minDCF of 0.1885 and an actDCF of 0.2093, outperforming all other combinations. The Bayes error plot for this fusion showed an excellent alignment between the actual and minimum DCF, demonstrating the effectiveness of this combination. The three-model fusion (Quad + GMM + SVM) achieved a minDCF of 0.1932 and actDCF of 0.2129, which, while strong, did not outperform the GMM + SVM fusion. Similarly, the Quad + GMM fusion had a minDCF of 0.2014 and actDCF of 0.2070, which was also close but not the best result.

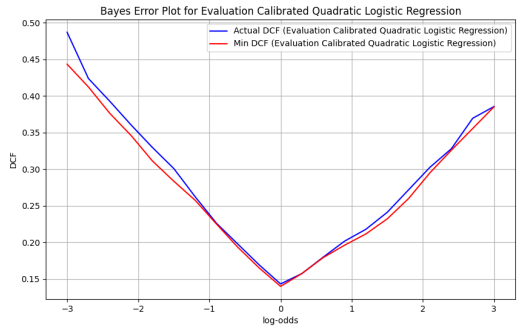
In conclusion, although the GMM model performed well individually, the fusion with the SVM resulted in the best fusion performance, with the lowest minDCF value of all the models. This shows the power of combining models to enhance performance and calibration across different log-odds values, but most importantly the strength of GMM on this specific case.



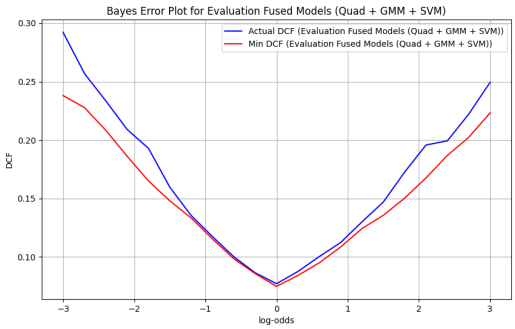
(a) Evaluation Diagonal GMM 4 components



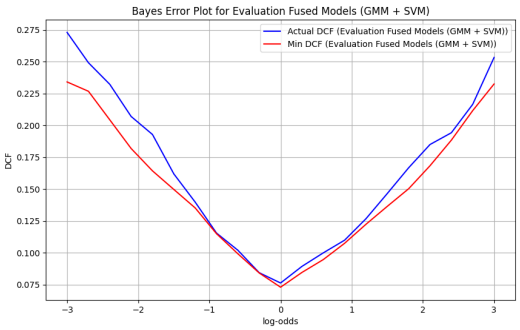
(b) Evaluation RBF SVM



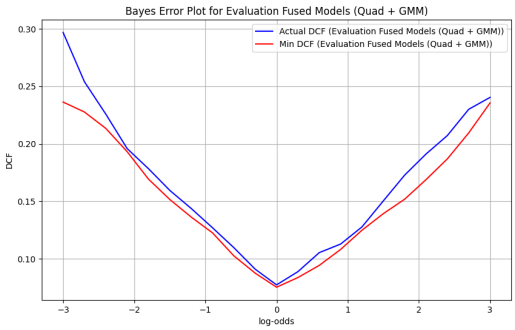
(c) Evaluation Quadratic Logistic Regression



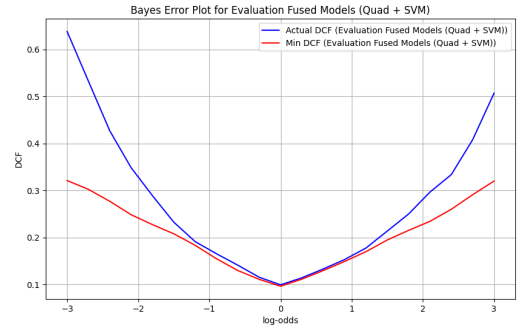
(d) Evaluation 3 models together



(e) Evaluation GMM + SVM



(f) Evaluation QLR + GMM



(g) Evaluation QLR + SVM

Figure 17: Bayes plots Evaluation of Calibrated Models and Fused Models

10.1 Analysis of other configurations of 1 method

Finally, the previously selected GMM model variations will be tested to determine whether the choice of 4 components and Diagonal was indeed optimal when applied to the evaluation dataset. Since these are uncalibrated models, the comparison will focus solely on the minDCF metric. The results are presented in the following table.

Model	**minDCF**	**actDCF**
GMM (full, 1 components)	0.4073	0.4334
GMM (full, 2 components)	0.2968	0.3032
GMM (full, 4 components)	0.2462	0.2551
GMM (full, 8 components)	**0.1858**	**0.1947**
GMM (full, 16 components)	0.2025	0.2063
GMM (full, 32 components)	0.2229	0.2403
GMM (diagonal, 1 components)	0.4098	0.4274
GMM (diagonal, 2 components)	0.3657	0.3723
GMM (diagonal, 4 components)	**0.1943**	**0.2043**
GMM (diagonal, 8 components)	0.2042	0.2196
GMM (diagonal, 16 components)	0.1885	0.2213
GMM (diagonal, 32 components)	0.1862	0.2007
Best GMM configuration: full covariance with 8 components		
Best minDCF: 0.1858		

Table 15: Evaluation of GMM Models with Full and Diagonal Covariance

The selected model, a GMM with diagonal covariance and 4 components, achieved a minDCF of 0.1943. When compared to other diagonal GMM models, the results indicate that while this configuration performed well, it was not the best model in terms of minDCF. Specifically, the GMM with diagonal covariance and 32 components achieved a slightly better minDCF of 0.1862. Similarly, Diagpnal GMMs with 16 and 8 components yielded minDCFs of 0.1885 and 0.2042, respectively. At the end, the 4-component full model was the best one, performing better than both the 1-component and 2-component models, which had significantly higher minDCFs of 0.4098 and 0.3657, respectively.

In terms of the overall effectiveness of the training strategy, the results demonstrate that the strategy was generally good. The chosen model delivered strong performance and was close to optimal compared to the other configurations. Although models with a higher number of components (16 or 32) offered marginal improvements in minDCF, the difference in performance was minimal. The selection of the 4-component model can be justified by its balance between performance and model complexity. Increasing the number of components typically introduces additional computational cost and the potential risk of overfitting, which makes the 4-component model a reasonable choice in terms of both efficiency and performance.

While the selected GMM model with diagonal covariance and 4 components was not the absolute best-performing model, it was near-optimal and performed robustly in the evaluation. The slight improvements seen in models with a higher number of components suggest that other configurations could have better results, but the trade-off in terms of computational efficiency supports the choice of the 4-component model. Therefore, the training strategy can be considered effective in achieving strong results without unnecessary complexity.

11 Conclusion

In this project, we have explored various machine learning models for the task of language detection based on speech embeddings. We have analyzed the characteristics of the data set, such as its imbalance, diversity, and non-linearity, and applied different pre-processing techniques, such as PCA and Z-score normalization. We have implemented and compared generative, discriminative, and kernel-based models, and evaluated their performance using the minDCF metric on both the training and test data sets. We have also calibrated the scores of the models using logistic regression and plotted the Bayes error, DET, and ROC curves. Our results show that the quadratic discriminant model with

$\pi_t = 0.5$ pre-processing achieved the best performance on the test data, with a minDCF of 0.265 for the average working point. This model was able to capture the non-linear patterns in the data and separate the classes more effectively than the other models. We conclude that machine learning models can be useful for language detection, but they require careful selection and tuning of the parameters and pre-processing techniques. We also suggest some possible directions for future work, such as exploring other types of embeddings, kernels, and regularization methods.

The investigation into fingerprint spoofing detection presented in this project demonstrates the effectiveness of various machine learning classifiers in distinguishing between genuine and fake fingerprints. Through meticulous data analysis, PCA and LDA provided valuable insights into feature importance and data separability, which were crucial for the application of classifiers. The comparative analysis revealed that Gaussian Mixture Models and Support Vector Machines, when calibrated properly, offer substantial potential in improving classification accuracy. Future work can focus on optimizing these models further and exploring the integration of other models that resolve the non-linear complexities of fingerprint data. The findings of this study sho the importance of the use of Machine Learning on the development of more secure biometric verification systems.