Machine Learning and Pattern Recognition: Fingerprint Spoofing detection

Hossein Kakavand s308581

July 17, 2024

1 Introduction

1.1 Abstract

The project task consists of a binary classification problem. The goal is to perform fingerprint spoofing detection, i.e., 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. First, we will explore the structure of the dataset and try to understand how it is distributed. Then, we will evaluate various classifiers and try to find the best system that can accurately classify our samples with the lowest cost.

2 Lab02

The main objective for this lab was to analyze the data coming from the dataset, which consists of labeled samples corresponding to the genuine (True, label 1) class and the fake (False, label 0) class. The dataset is composed of samples that represent fingerprint images through low-dimensional representations called embeddings. Each fingerprint is represented by a 6-dimensional vector of continuous real numbers. The individual components of the embeddings do not have any physical interpretation.

2.1 Data Analysis

The training files for the project are stored in file Project/trainData.txt. The first 6 values of each row are the features, whereas the last value of each row represents the class (1 or 0). The samples are not ordered. I loaded the dataset and plotted the histogram and pair-wise scatter plots of the different features.

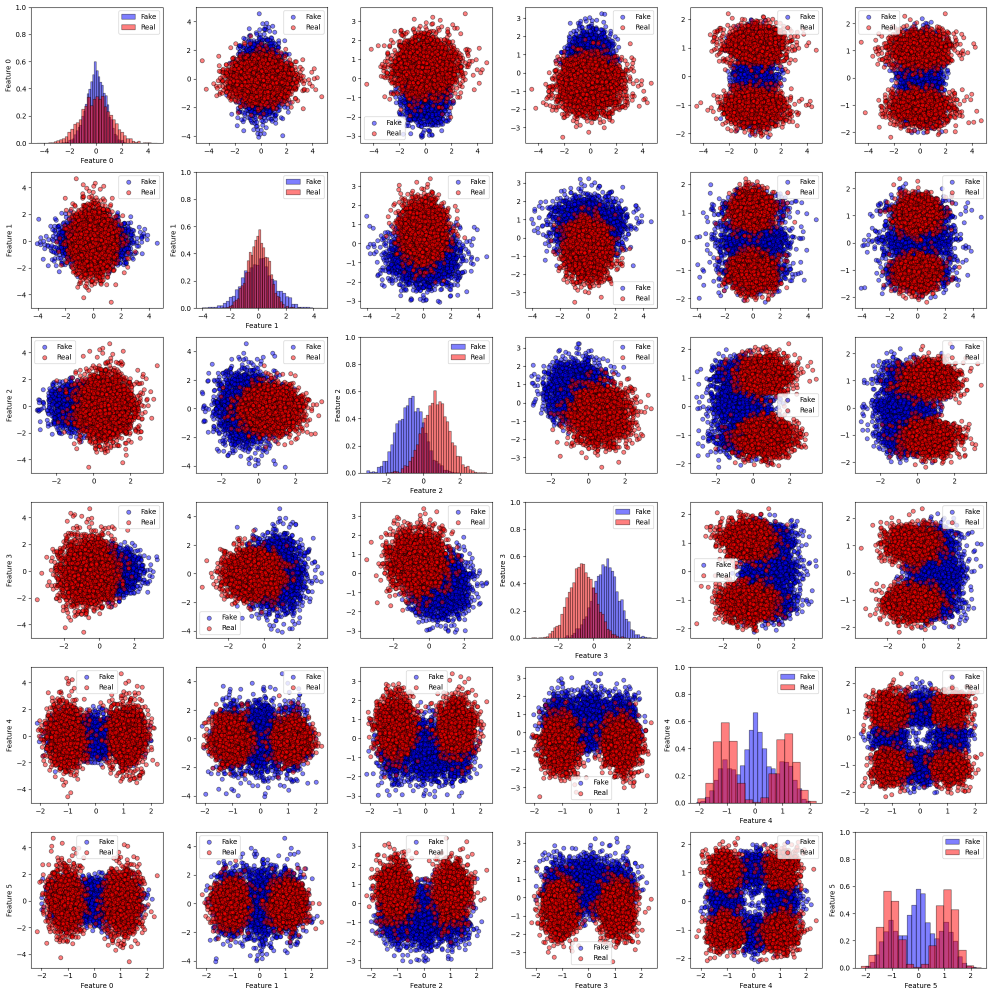

Figure 1: Pair-wise Plot of Data

2.2 Analyze the first two features. What do you observe? Do the classes overlap? If so, where? Do the classes show similar mean for the first two features? Are the variances similar for the two classes? How many modes are evident from the histograms (i.e., how many "peaks" can be observed)?

Answer:

Feature 0 vs Feature 1:

- **Observation:** The scatter plot shows significant overlap between the Real (red) and Fake (blue) classes. The Real class has a broader distribution compared to the Fake class, which is more concentrated around zero.
- **Overlap:** The classes overlap around the center, particularly around the mean value of zero.
- **Mean Comparison:** Both classes have similar means, centered around zero.

- **Variance Comparison:** The Real class has a larger variance compared to the Fake class.
- **Modes:** There is one evident mode for both classes.

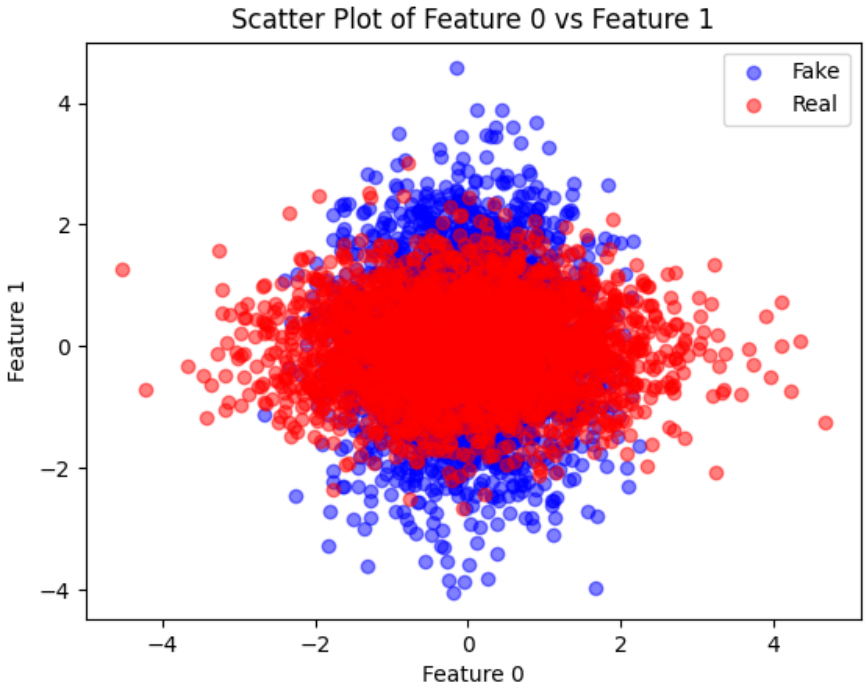

Figure 2: Scatter Plot of Feature 0 vs Feature 1

Conclusion: The first two features show significant overlap between the Real and Fake classes, with both features having similar means but different variances. The Real class tends to have a broader distribution. Both features have one evident mode.

2.3 Analyze the third and fourth features. What do you observe? Do the classes overlap? If so, where? Do the classes show similar mean for these two features? Are the variances similar for the two classes? How many modes are evident from the histograms?

Answer:

Feature 2 vs Feature 3:

- **Observation:** The scatter plot shows significant overlap between the classes. The Fake class has a more pronounced peak to the left of zero, while the Real class has a peak to the right.

- **Overlap:** The classes overlap around the zero mark but are generally centered on opposite sides.
- **Mean Comparison:** The means of the classes are different, with the Fake class having a negative mean and the Real class having a positive mean.
- **Variance Comparison:** The variances are different, with the Fake class having a tighter spread compared to the Real class.
- **Modes:** There are two modes for each class, indicating a bimodal distribution.

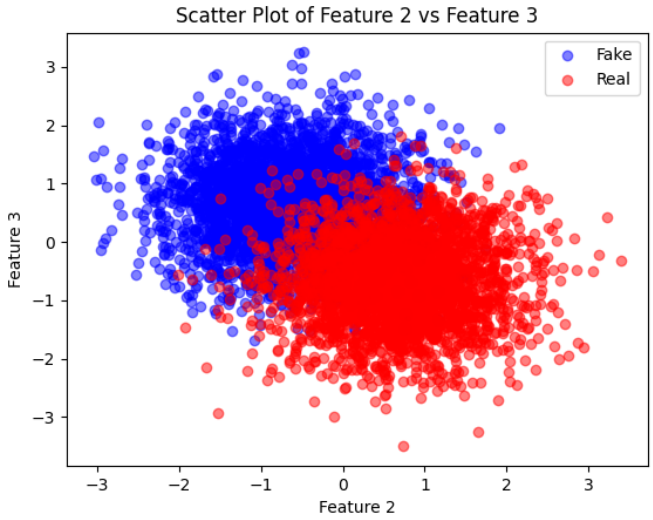

Figure 3: Scatter Plot of Feature 2 vs Feature 3

Conclusion: The third and fourth features show significant differences in means and variances between the classes. Both features exhibit bimodal distributions for both classes.

2.4 Analyze the last two features. What do you observe? Do the classes overlap? If so, where? How many modes are evident from the histograms? How many clusters can you notice from the scatter plots for each class?

Answer:

Feature 4 vs Feature 5:

- **Observation:** The scatter plot shows little overlap between the classes. Both classes show multiple peaks, indicating multimodal distributions.
- **Overlap:** The classes overlap significantly, making it difficult to distinguish them based solely on this feature.

- **Modes:** There are multiple modes for both classes, making the distribution complex.
- **Scatter Plots:** The scatter plots show clear clustering for each feature pair.
- **Clusters:** Multiple clusters can be observed for each class, indicating complex underlying structures in the data.

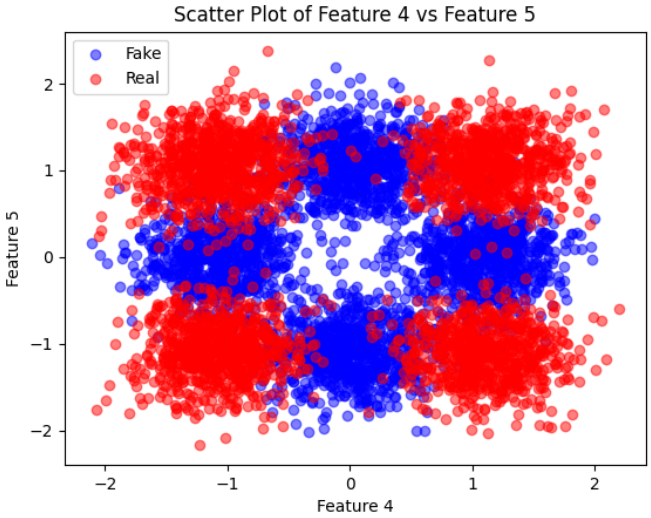

Figure 4: Scatter Plot of Feature 4 vs Feature 5

Conclusion: The last two features the situation is a bit different, they overlap a bit and the histograms present for both features two peaks for the genuine class and one for the fake class, looking the scatter plot we can see 4 cluster for both classes.

3 Lab03

The main objective for this lab was to apply PCA and LDA to the project data and analyze their effects on the features and class distributions. We will explore the effects of PCA by plotting the histogram of the projected features for the 6 PCA directions. Additionally, we will apply LDA and compute the histogram of the projected LDA samples, comparing the overlap of the classes to the original features. We will also apply LDA as a classifier and analyze the performance as a function of the number of PCA dimensions.

3.1 Data Analysis

The training files for the project are stored in file Project/trainData.txt. The first 6 values of each row are the features, whereas the last value of each row represents the class (1 or 0). We applied PCA and LDA to the dataset and plotted the histogram of the projected features for the 6 PCA directions and the LDA direction.

3.2 Analyze the effects of PCA on the features. Plot the histogram of the projected features for the 6 PCA directions, starting from the principal (largest variance). What do you observe? What are the effects on the class distributions? Can you spot the different clusters inside each class?

Answer:

- **PCA Direction 1:** The histogram shows that the Real (red) and Fake (blue) classes are more separable compared to the original features. There are clear clusters within each class.
- **PCA Direction 2:** The histogram indicates that the classes overlap significantly, similar to the original features. The clusters are less distinct.
- **PCA Direction 3:** There is moderate overlap between the classes. The clusters within each class are visible but not as distinct as in PCA Direction 1.
- **PCA Direction 4:** The histogram shows significant overlap between the classes. The clusters are less distinguishable.
- **PCA Direction 5:** The overlap between the classes is similar to PCA Direction 4. The clusters are not very clear.
- **PCA Direction 6:** The histogram shows significant overlap, with clusters within each class being indistinct.

Conclusion: PCA improves the separability of the classes in the first few directions, with PCA Direction 1 showing the best separation. However, the separability decreases in the higher PCA directions, with significant overlap and less distinct clusters.

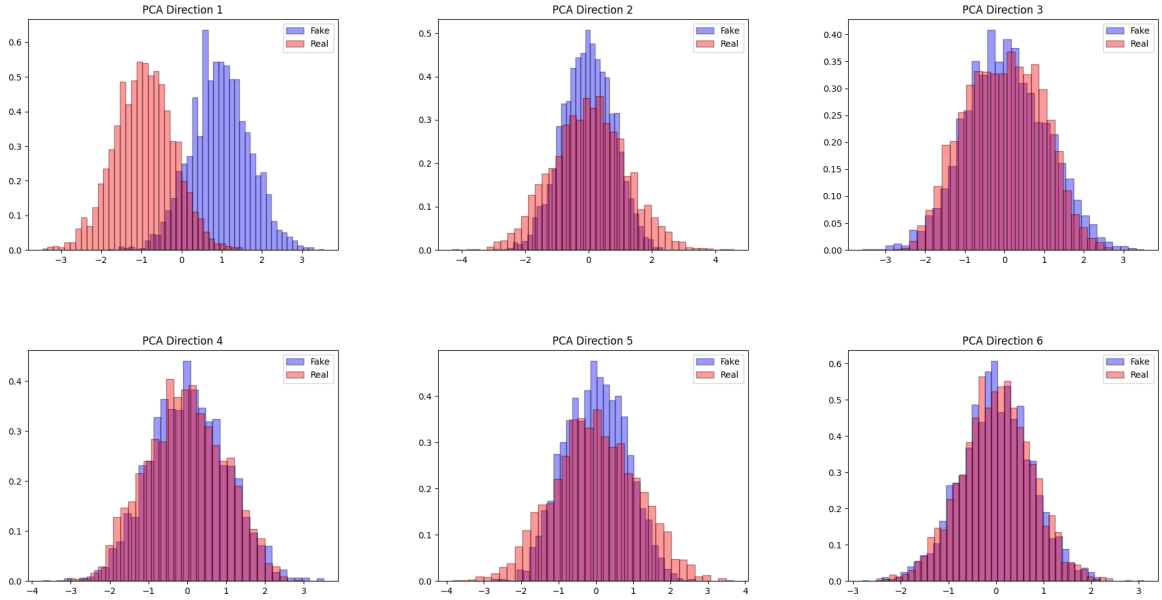

Figure 5: Histograms of Projected Features for the 6 PCA Directions

3.3 Apply LDA (1 dimensional, since we have just two classes), and compute the histogram of the projected LDA samples. What do you observe? Do the classes overlap? Compared to the histogram of the 6 features you computed in Laboratory 2, is LDA finding a good direction with little class overlap?

Answer:

- **With LDA:** The histogram shows that the Real (red) and Fake (blue) classes are well-separated with minimal overlap. The LDA projection has found a direction with significant class separability.
- **Without LDA:** The histogram of the original features shows significant overlap between the classes. Compared to the original features, LDA has found a direction with much less class overlap.

Conclusion: LDA significantly reduces the class overlap compared to the original features, indicating that LDA has found a good direction with minimal class overlap.

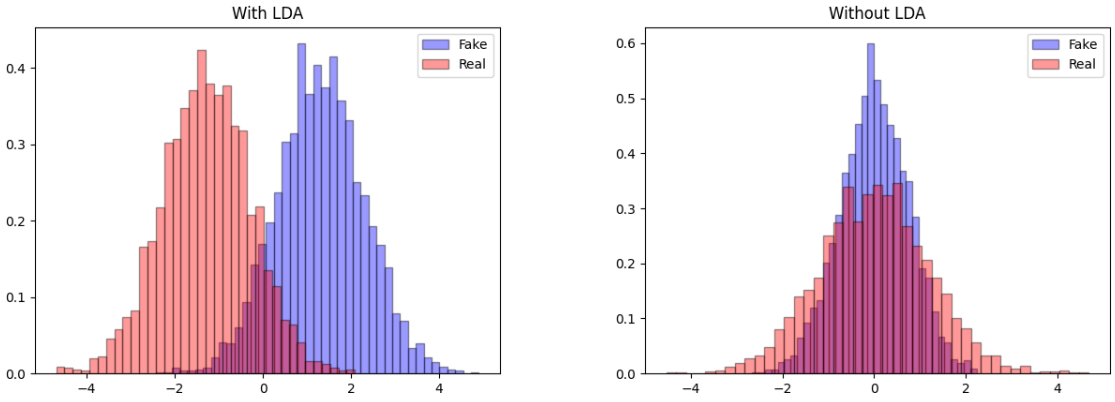

Figure 6: Histograms of LDA With and Without LDA

3.4 Apply LDA as classifier. Divide the dataset in model training and validation sets. Apply LDA, and select the threshold as in the previous sections. Compute the predictions, and the error rate. Now try changing the value of the threshold. What do you observe? Can you find values that improve the classification accuracy?

Answer:

- Initial Threshold: -0.018534376786207285
- Initial Validation Error Rate: 9.30%
- Initial Accuracy: 90.70%
- Threshold: -1.02, Validation Error Rate: 19.60%, Accuracy: 80.40%
- Threshold: -0.80, Validation Error Rate: 15.35%, Accuracy: 84.65%
- Threshold: -0.57, Validation Error Rate: 12.55%, Accuracy: 87.45%
- Threshold: -0.35, Validation Error Rate: 10.90%, Accuracy: 89.10%
- Threshold: -0.13, Validation Error Rate: 9.30%, Accuracy: 90.70%
- Threshold: 0.09, Validation Error Rate: 9.45%, Accuracy: 90.55%
- Threshold: 0.31, Validation Error Rate: 10.05%, Accuracy: 89.95%
- Threshold: 0.54, Validation Error Rate: 12.00%, Accuracy: 88.00%
- Threshold: 0.76, Validation Error Rate: 15.20%, Accuracy: 84.80%

- Threshold: 0.98, Validation Error Rate: 18.65%, Accuracy: 81.35%

Conclusion: The initial threshold provides the best accuracy at 90.70%. Changing the threshold can either improve or worsen the accuracy, with a clear optimal threshold providing the best performance.

3.5 Try pre-processing the features with PCA. Apply PCA (estimated on the model training data only), and then classify the validation data with LDA. Analyze the performance as a function of the number of PCA dimensions m. What do you observe? Can you find values of m that improve the accuracy on the validation set? Is PCA beneficial for the task when combined with the LDA classifier?

Answer:

- **Best PCA Dimensions:** 3
- **Best Threshold:** -0.0183981898891038
- **Best Validation Error Rate:** 9.25%
- **Best Accuracy:** 90.75%

Conclusion: Using PCA as a pre-processing step can improve the accuracy of the LDA classifier. In this case, using the first 3 PCA dimensions provided the best performance, slightly improving the accuracy to 90.75%.

3.6 Correlation Matrix Analysis

In this section, we present the correlation matrices for the overall data, fake class, and real class. These matrices help us understand the relationships between features within each class and the overall dataset.

Observations:

- **Overall Correlation Matrix:** The features have low correlation with each other, indicating that the features are fairly independent. However, there are some noticeable negative correlations, such as between feature 2 and feature 3.
- **Class Fake Correlation Matrix:** The features within the fake class also show low correlation, similar to the overall matrix. The negative correlation between feature 2 and feature 3 is less pronounced.
- **Class Real Correlation Matrix:** The real class features exhibit low correlation with each other. There is a slight negative correlation between features 2 and 3, similar to the overall correlation matrix.

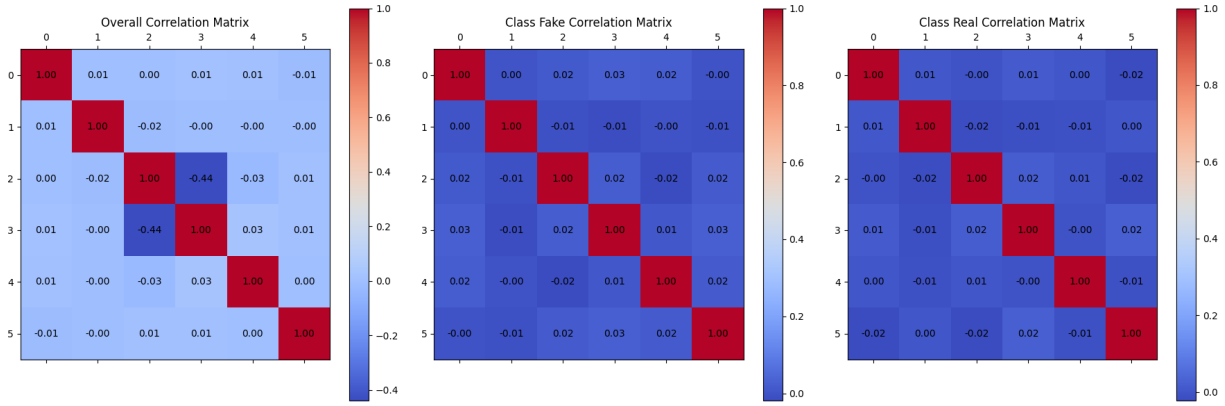

Figure 7: Overall, Fake Class, and Real Class Correlation Matrices

Conclusion: The correlation matrices reveal that the features are largely independent within each class and the overall dataset. The negative correlation between features 2 and 3 is a consistent observation across all matrices.

4 Lab04

The main objective for this lab was to fit uni-variate Gaussian models to the different features of the project dataset. We computed the Maximum Likelihood (ML) estimates for the parameters of a 1D Gaussian distribution for each feature and plotted the distribution density on top of the normalized histogram. This preliminary analysis helps us understand how well the Gaussian densities fit the data for each feature.

4.1 Data Analysis

The training files for the project are stored in file Project/trainData.txt. The first 6 values of each row are the features, whereas the last value of each row represents the class (1 or 0). We computed the ML estimates for the mean (M) and covariance (C) for each feature and plotted the Gaussian fits.

4.2 Analyze the Gaussian fits for each feature. What do you observe? Are there features for which the Gaussian densities provide a good fit? Are there features for which the Gaussian model seems significantly less accurate?

Answer:

Feature 0:

- **Real Class:** M = 0.00054, C = 1.43023, LL = -4809
- **Fake Class:** M = 0.00287, C = 0.56958, LL = -3401
- **Observation:** The Gaussian fits well for both classes, with the Real class showing a broader spread.

Feature 1:

- **Real Class:** M = -0.00852, C = 0.57827, LL = -3446
- **Fake Class:** M = 0.01869, C = 1.42086, LL = -4767
- **Observation:** The Gaussian fit is reasonably accurate, though there is some discrepancy at the tails.

Feature 2:

- **Real Class:** M = 0.66523, C = 0.5489, LL = -3368
- **Fake Class:** M = -0.68094, C = 0.54997, LL = -3348
- **Observation:** The Gaussian fit is less accurate, particularly for the Fake class, which has a peak that is not well-captured by the Gaussian model.

Feature 3:

- **Real Class:** M = -0.66419, C = 0.55334, LL = -3380
- **Fake Class:** M = 0.6708, C = 0.53604, LL = -3310
- **Observation:** The Gaussian fit is reasonably accurate, capturing the main distribution peaks for both classes.

Feature 4:

- **Real Class:** M = -0.04172, C = 1.31776, LL = -4686
- **Fake Class:** M = 0.02795, C = 0.6800, LL = -3666
- **Observation:** The Gaussian fit is less accurate, with the model failing to capture the multimodal nature of the distribution for both classes.

Feature 5:

- **Real Class:** M = 0.02393, C = 1.28702, LL = -4650
- **Fake Class:** M = -0.0058, C = 0.70503, LL = -3720
- **Observation:** The Gaussian fit is reasonably accurate for the Real class but less so for the Fake class, which has a more complex distribution.

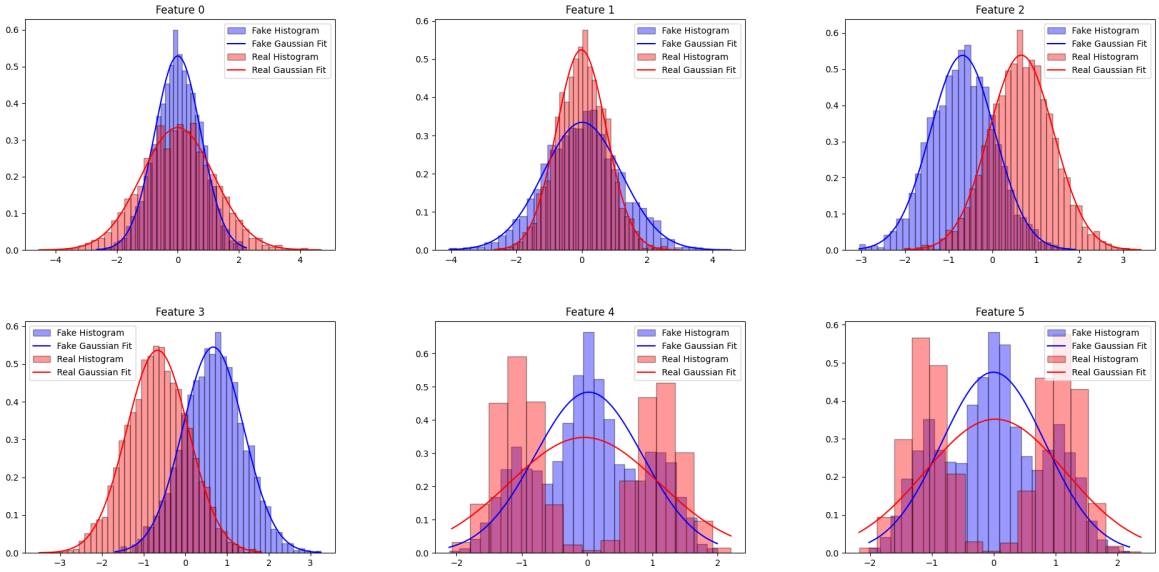

Figure 8: Gaussian Fits for Features 0 to 5

Conclusion: The Gaussian fits vary in accuracy across the different features. Some features, like Feature 0, show a good fit, while others, like Feature 4, demonstrate the limitations of the Gaussian model in capturing the true data distribution, especially for multimodal distributions.

5 Lab05

The main objective for this lab was to apply the MVG model to the project data. We split the dataset into model training and validation subsets, trained the model parameters on the model training portion, and computed log-likelihood ratios (LLRs) for the validation subset. We compared the results with MVG, tied Gaussian, and Naive Bayes Gaussian models. Additionally, we analyzed the effect of PCA as pre-processing.

5.1 Data Analysis

We applied the MVG model, tied Gaussian model, and Naive Bayes model, and evaluated their performance with and without PCA.

5.2 Evaluate the MVG, tied Gaussian, and Naive Bayes models. What do you observe? Which model performs better? How does PCA affect the performance?

Answer:

MVG Model:

- **Without PCA:** Accuracy: 86.25%, Normalized DCF: 0.5904, Min DCF: 0.2629
- **Filtered First 4 Features:** Accuracy: 85.10%, Normalized DCF: 0.6271, Unnormalized DCF: 0.0627, Min DCF: 0.3022
- **Using Features 1 and 2:** Accuracy: 52.60%, Normalized DCF: 0.9931, Unnormalized DCF: 0.0993, Min DCF: 0.9467
- **Using Features 3 and 4:** Accuracy: 83.00%, Normalized DCF: 0.6688, Unnormalized DCF: 0.0669, Min DCF: 0.3736
- **PCA (2 components):** Accuracy: 86.25%, Normalized DCF: 0.5904, Min DCF: 0.2629
- **PCA (3 components):** Accuracy: 83.30%, Normalized DCF: 0.6470, Min DCF: 0.3563
- **PCA (4 components):** Accuracy: 84.20%, Normalized DCF: 0.6142, Min DCF: 0.3012
- **PCA (5 components):** Accuracy: 86.30%, Normalized DCF: 0.5825, Min DCF: 0.2738
- **PCA (6 components):** Accuracy: 86.25%, Normalized DCF: 0.5904, Min DCF: 0.2629

Tied Gaussian Model:

- **Without PCA:** Accuracy: 83.20%, Normalized DCF: 0.6559, Min DCF: 0.3628
- **Filtered First 4 Features:** Accuracy: 83.15%, Normalized DCF: 0.6559, Unnormalized DCF: 0.0656, Min DCF: 0.3726
- **Using Features 1 and 2:** Accuracy: 49.60%, Normalized DCF: 1.0000, Unnormalized DCF: 0.1000, Min DCF: 1.0000
- **Using Features 3 and 4:** Accuracy: 83.20%, Normalized DCF: 0.6559, Unnormalized DCF: 0.0656, Min DCF: 0.3756
- **PCA (2 components):** Accuracy: 83.30%, Normalized DCF: 0.6609, Min DCF: 0.3630
- **PCA (3 components):** Accuracy: 83.45%, Normalized DCF: 0.6559, Min DCF: 0.3681
- **PCA (4 components):** Accuracy: 83.35%, Normalized DCF: 0.6559, Min DCF: 0.3610
- **PCA (5 components):** Accuracy: 83.25%, Normalized DCF: 0.6559, Min DCF: 0.3648
- **PCA (6 components):** Accuracy: 83.20%, Normalized DCF: 0.6559, Min DCF: 0.3628

Naive Bayes Model:

- **Without PCA:** Accuracy: 86.40%, Normalized DCF: 0.5835, Min DCF: 0.2570
- **Filtered First 4 Features:** Accuracy: 84.95%, Normalized DCF: 0.6251, Unnormalized DCF: 0.0625, Min DCF: 0.3013
- **Using Features 1 and 2:** Accuracy: 52.60%, Normalized DCF: 0.9931, Unnormalized DCF: 0.0993, Min DCF: 0.9474
- **Using Features 3 and 4:** Accuracy: 83.00%, Normalized DCF: 0.6688, Unnormalized DCF: 0.0669, Min DCF: 0.3736
- **PCA (2 components):** Accuracy: 83.35%, Normalized DCF: 0.6668, Min DCF: 0.3562
- **PCA (3 components):** Accuracy: 83.35%, Normalized DCF: 0.6559, Min DCF: 0.3645
- **PCA (4 components):** Accuracy: 83.25%, Normalized DCF: 0.6509, Min DCF: 0.3614
- **PCA (5 components):** Accuracy: 83.45%, Normalized DCF: 0.6539, Min DCF: 0.3545

- **PCA (6 components):** Accuracy: 83.50%, Normalized DCF: 0.6569, Min DCF: 0.3535

Conclusion: The Naive Bayes model generally performs well without PCA, achieving the highest accuracy of 86.40%. The MVG model also performs well, especially with 5 PCA components. The Tied Gaussian model performs consistently lower than the other two models. PCA can improve performance, but its impact varies depending on the model and the number of components.

5.3 Comparison of Detection Cost Functions

The following plots show the Detection Cost Function (DCF) and the minimum Detection Cost Function (minDCF) for MVG, Tied Gaussian, and Naive Bayes models with and without PCA.

Conclusion: The detection cost functions show that the Naive Bayes model generally performs better across different PCA components, with lower normalized DCF and minDCF values. The MVG model also performs well, especially with 5 PCA components. The Tied Gaussian model performs consistently lower than the other models.

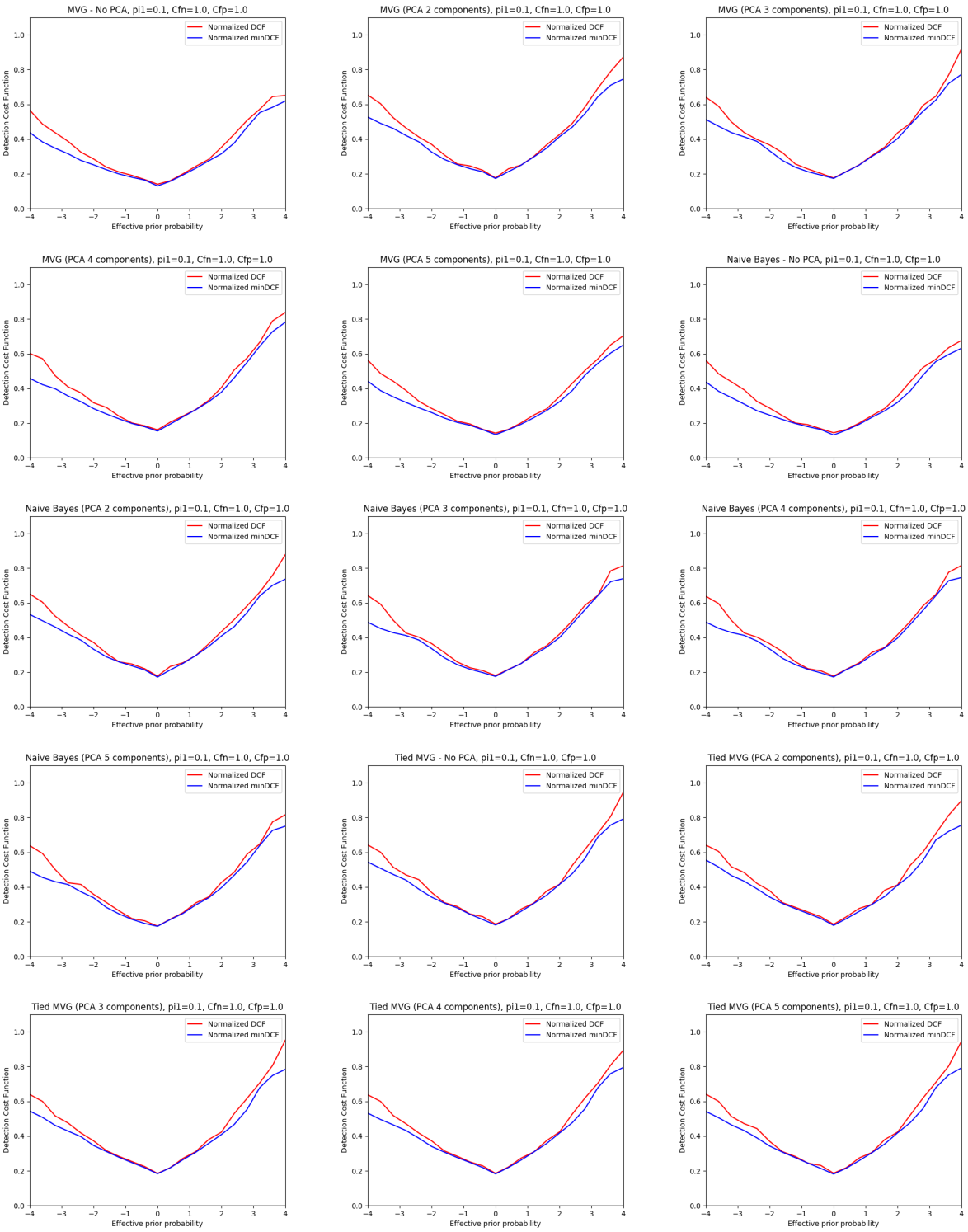

Figure 9: Detection Cost Functions for MVG, Tied Gaussian, and Naive Bayes Models with PCA

6 Lab07

In this lab, we analyze the performance of the MVG classifier and its variants for different applications. We start by considering five applications, given by the pairs of (pi1; Cfn; Cfp):

- (0.5; 1.0; 1.0), i.e., uniform prior and costs
- (0.9; 1.0; 1.0), i.e., the prior probability of a genuine sample is higher
- (0.1; 1.0; 1.0), i.e., the prior probability of a fake sample is higher
- (0.5; 1.0; 9.0), i.e., uniform prior but higher cost for accepting a fake sample
- (0.5; 9.0; 1.0), i.e., uniform prior but higher cost for rejecting a genuine sample

6.1 Analysis

We represent the applications in terms of effective prior and observe how the costs of misclassifications are reflected in the prior. We focus on three applications with effective priors $= 0.1$, $= 0.5$, and $= 0.9$. For each application, we compute the optimal Bayes decisions for the validation set for the MVG models and its variants, with and without PCA. We compute DCF (actual) and minimum DCF for the different models and compare their performance.

6.2 Results

- **Prior 0.5**
  - MVG: DCF = 0.140, minDCF = 0.130
  - Tied: DCF = 0.186, minDCF = 0.181
  - Naive Bayes: DCF = 0.144, minDCF = 0.131

  The best model for this application according to the minDCF is the standard MVG.

- **Prior 0.9**
  - MVG: DCF = 0.400, minDCF = 0.342
  - Tied: DCF = 0.463, minDCF = 0.442
  - Naive Bayes: DCF = 0.389, minDCF = 0.351

  The best model for this application according to the minDCF is the standard MVG.

- **Prior 0.1**
  - MVG: DCF = 0.305, minDCF = 0.263
  - Tied: DCF = 0.406, minDCF = 0.363
  - Naive Bayes: DCF = 0.302, minDCF = 0.257

  The best model for this application according to the minDCF is the Naive Bayes model.

6.3 PCA Analysis

Considering only the prior 0.1, applying PCA and comparing the different models, the best results are:

- **MVG** with m = 5: minDCF = 0.274, actDCF = 0.304
- **Tied** with m = 4: minDCF = 0.361, actDCF = 0.403
- **Naive** with m = 5: minDCF = 0.354, actDCF = 0.393

6.4 Comparison of Detection Cost Functions

The following plots show the Detection Cost Function (DCF) and the minimum Detection Cost Function (minDCF) for MVG, Tied Gaussian, and Naive Bayes models for different applications.

Conclusion: The detection cost functions show that the standard MVG model generally performs better across different applications. The Naive Bayes model performs well for applications with a prior of 0.1. The Tied Gaussian model consistently shows lower performance compared to the other models. PCA can improve performance, particularly for the MVG model with a prior of 0.1.

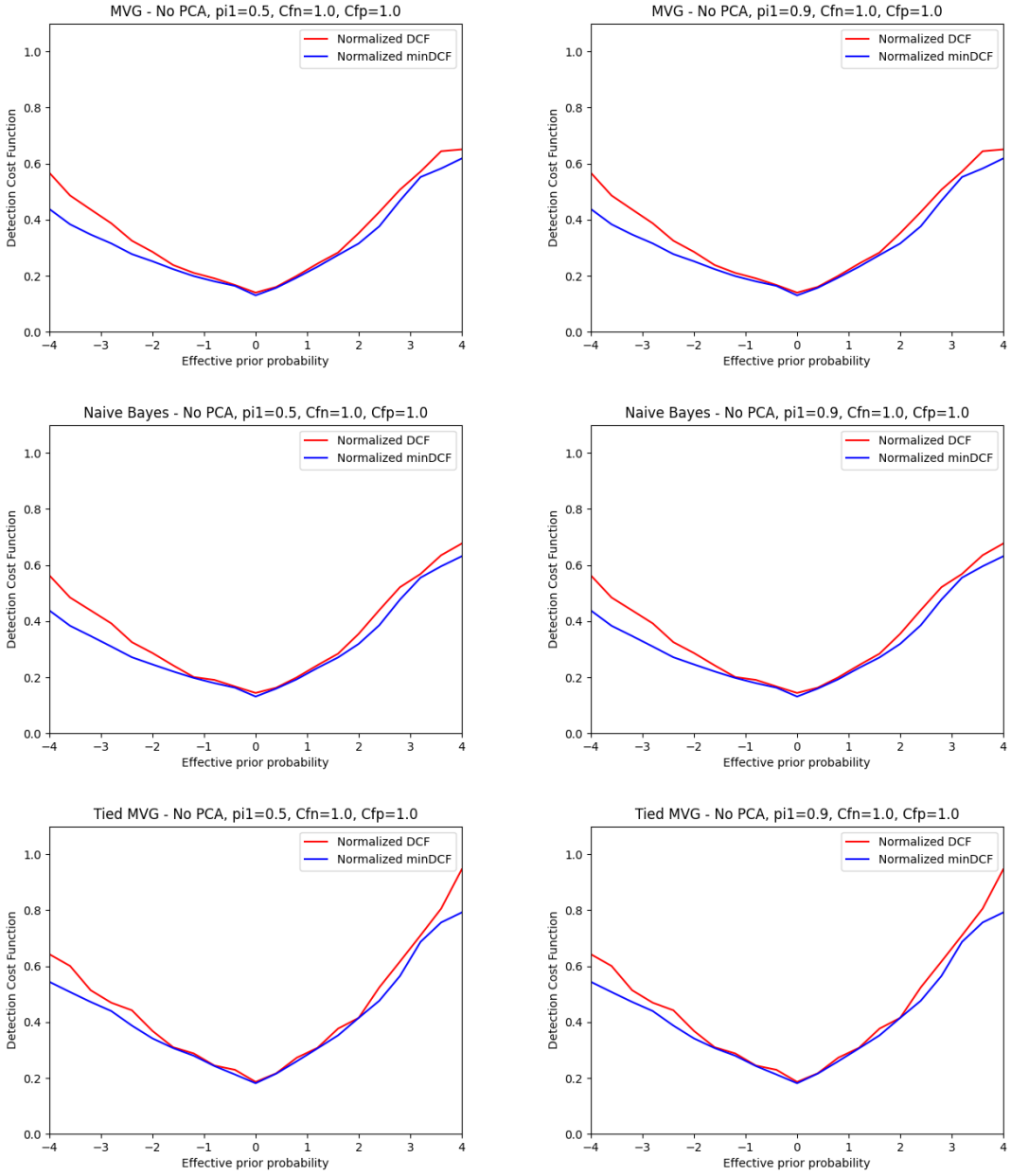

Figure 10: Detection Cost Functions for MVG, Tied Gaussian, and Naive Bayes Models for Different Applications

7 Lab 08

The main objective of this lab is to analyze the binary logistic regression model on the project data. We will start by considering the standard, non-weighted version of the model, without any pre-processing. The analysis includes training the model using different values for λ and evaluating the corresponding actual DCF and minimum DCF for the primary application $\pi_T = 0.1$. Additionally, we will explore the effect of regularization with fewer training samples, analyze the prior-weighted version of the model, and evaluate the quadratic

logistic regression model.

7.1 Standard Logistic Regression

We trained the logistic regression model using different values for λ from 10^{-4} to 10^{2} and computed the actual DCF and minimum DCF for $\pi_T = 0.1$. The following results were obtained:

- **Best Lambda:** 0.0001
- **Best Actual DCF:** 0.4021
- **Best Min DCF:** 0.3611

Conclusion: The regularization coefficient λ significantly affects the actual DCF and minimum DCF. Higher values of λ degrade the actual DCF, indicating that regularization might be ineffective for this dataset due to the large number of samples.

7.2 Reduced Training Samples

To better understand the role of regularization, we repeated the analysis with only 1 out of 50 training samples. The following results were obtained:

- **Best Lambda:** 0.00031622776601683794
- **Best Actual DCF:** 0.4297
- **Best Min DCF:** 0.3650

Conclusion: With fewer training samples, regularization becomes more effective, reducing overfitting and improving model performance.

7.3 Prior-Weighted Logistic Regression

We repeated the analysis with the prior-weighted version of the logistic regression model. The following results were obtained:

- **Best Lambda:** 0.0001
- **Best Actual DCF:** 0.4021
- **Best Min DCF:** 0.3640

Conclusion: The prior-weighted model shows similar performance to the standard model, indicating no significant advantages for this task.

7.4 Quadratic Logistic Regression

We expanded the features and trained the quadratic logistic regression model with different values for λ. The following results were obtained:

- **Best Lambda:** 0.00031622776601683794
- **Best Actual DCF:** 0.2656
- **Best Min DCF:** 0.2436

Conclusion: Regularization is effective for the quadratic model, significantly improving performance. The quadratic model outperforms the linear model, indicating that feature expansion captures more complex relationships in the data.

7.5 Effect of Centering

We analyzed the effects of centering on the logistic regression model. The following results were obtained:

- **Best Lambda:** 0.0001
- **Best Actual DCF:** 0.4021
- **Best Min DCF:** 0.3640

Conclusion: Centering has minor effects on the model results, as the original features were almost standardized.

7.6 Model Comparison

The best models in terms of minimum DCF are not necessarily those that provide the best actual DCFs. Comparing all models trained so far, in terms of minDCF for $\pi_T = 0.1$, we observe that the quadratic logistic regression model achieves the best results. This model captures more complex relationships and provides better separation rules.

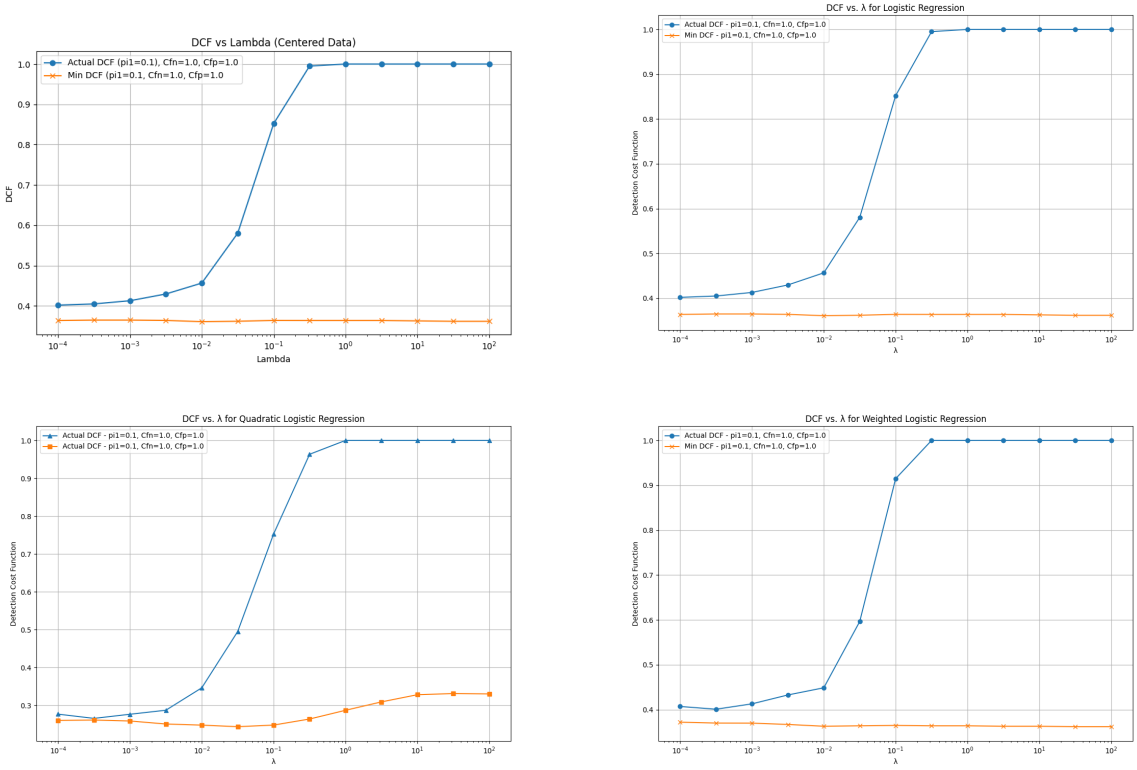

Figure 11: DCF vs. λ for Different Logistic Regression Models

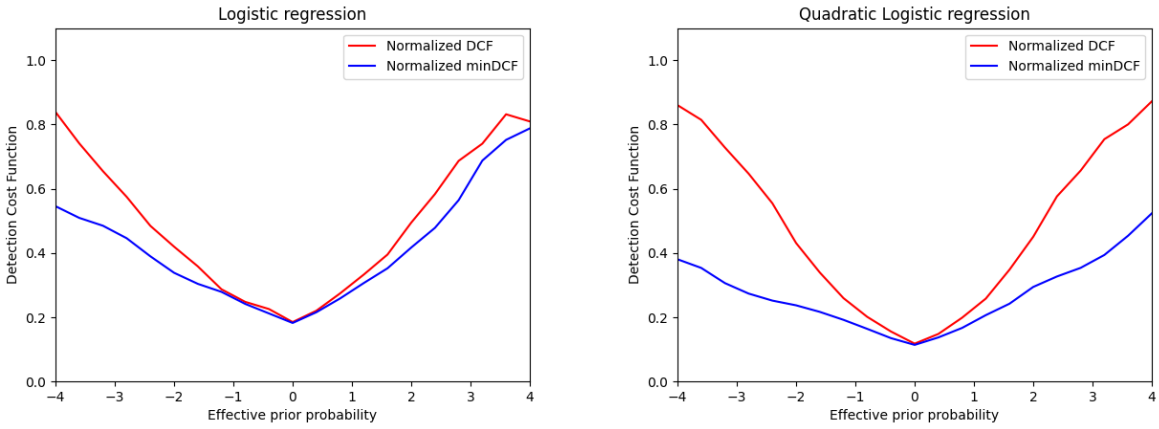

Figure 12: Comparison of Logistic and Quadratic Logistic Regression Models

8 Lab 09

The main objective of this lab is to analyze the performance of SVM models on the project data. We will start with the linear SVM model, then explore polynomial and RBF kernels. The analysis includes training the models using different values of the regularization parameter C and evaluating the corresponding actual DCF and minimum DCF for the primary

application $\pi_T = 0.1$. Additionally, we will explore the effect of centering the data.

8.1 Linear SVM

We trained the linear SVM model using different values for C from 10^{-5} to 10^{0} and computed the actual DCF and minimum DCF for $\pi_T = 0.1$. The following results were obtained:

- **Best Lambda:** 1.0
- **Best Actual DCF:** 0.4894
- **Best Min DCF:** 0.3582

Conclusion: The regularization coefficient C significantly affects the actual DCF and minimum DCF. Higher values of C degrade the actual DCF, indicating that regularization might be ineffective for this dataset due to the large number of samples.

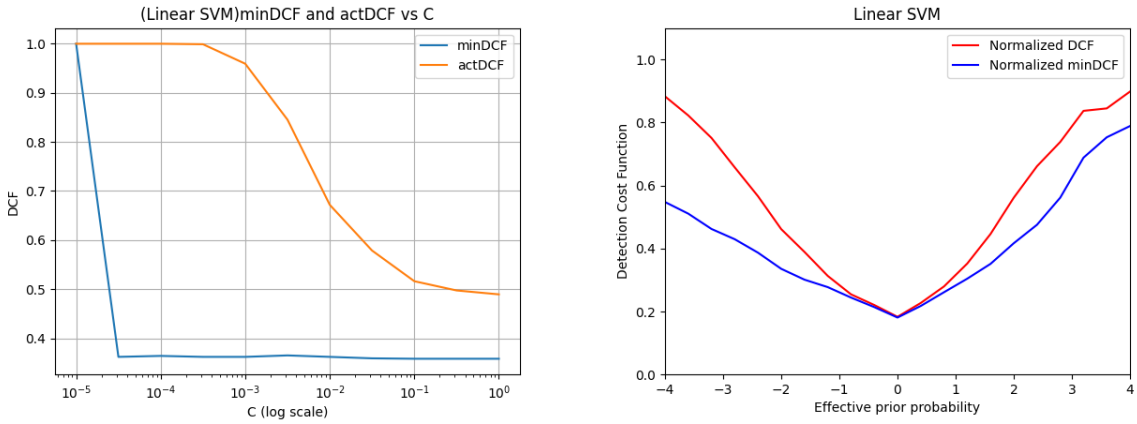

Figure 13: DCF vs. C for Linear SVM

8.2 Polynomial SVM

We trained the polynomial SVM model with degree $d = 2$ using different values for C and computed the actual DCF and minimum DCF for $\pi_T = 0.1$. The following results were obtained:

- **Best Lambda:** 0.01
- **Best Actual DCF:** 0.4428
- **Best Min DCF:** 0.2588

Conclusion: The polynomial kernel improves performance over the linear model, capturing more complex relationships in the data.

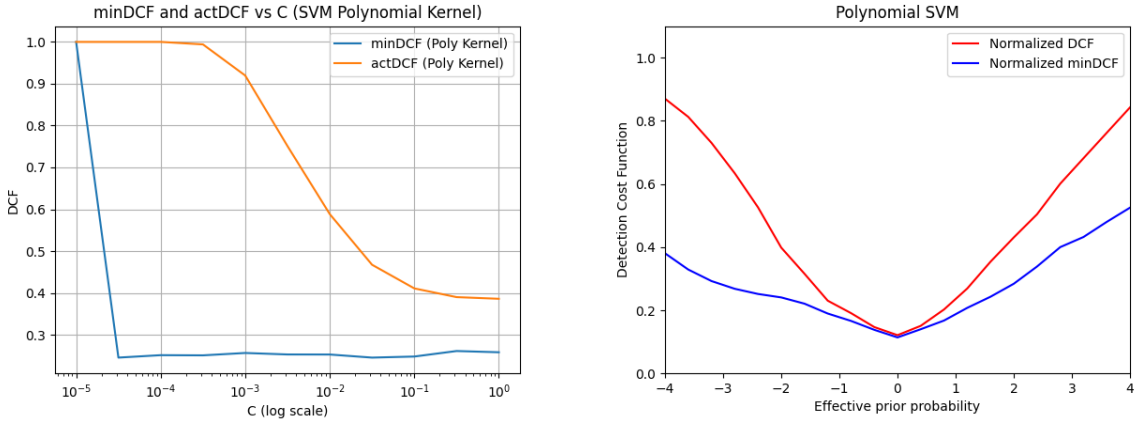

Figure 14: DCF vs. C for Polynomial SVM

8.3 RBF Kernel SVM

We trained the RBF kernel SVM model using different values for C and γ and computed the actual DCF and minimum DCF for $\pi_T = 0.1$. The following results were obtained:

- **Best Lambda:** 100.0
- **Best Gamma:** 0.049787
- **Best Actual DCF:** 0.4325
- **Best Min DCF:** 0.2499

Conclusion: The RBF kernel provides the best performance among the SVM models, effectively capturing complex relationships in the data.

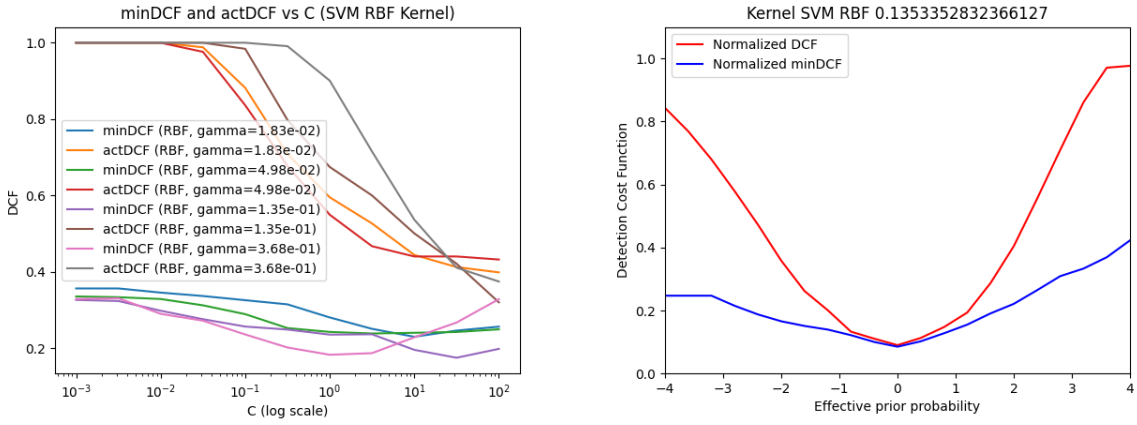

Figure 15: DCF vs. C and γ for RBF Kernel SVM

8.4 Effect of Centering

We analyzed the effects of centering on the SVM models. The following results were obtained for the linear SVM:

- **Best Lambda:** 1.0
- **Best Actual DCF:** 0.4894
- **Best Min DCF:** 0.3582

Conclusion: Centering has minor effects on the model results, as the original features were almost standardized.

8.5 Model Comparison

The best models in terms of minimum DCF are not necessarily those that provide the best actual DCFs. Comparing all models trained so far, including Gaussian models, in terms of minDCF for $\pi_T = 0.1$, we observe that the RBF kernel SVM achieves the best results. This model captures more complex relationships and provides better separation rules.

8.6 Polynomial Kernel SVM with $d = 4$

For the optional part of the lab, we considered the polynomial kernel with $d = 4$, $c = 1$, and $\xi = 0$. The model was trained with different values of C using numpy.logspace$(-5, 0, 11)$. The results were evaluated in terms of minDCF and actDCF.

8.6.1 Results

The following best result was obtained for the SVM with polynomial kernel $d = 4$:

- **Best C:** 0.0031622776601683794
- **Best Actual DCF:** 0.1960
- **Best Min DCF:** 0.1542

8.6.2 Conclusion

The polynomial kernel SVM with $d = 4$ achieves better performance with higher values of C. The error rate and accuracy improve significantly with increasing C.

The characteristics of the dataset, especially the last two features of each sample, are better captured by the polynomial kernel. This is consistent with the dataset's complex structure, which requires nonlinear separation surfaces. The polynomial kernel allows for more flexibility in defining the decision boundaries, leading to improved performance.

Overall, the polynomial kernel $d = 4$ outperforms the linear SVM and other simpler models, providing better results in terms of both minDCF and actDCF.

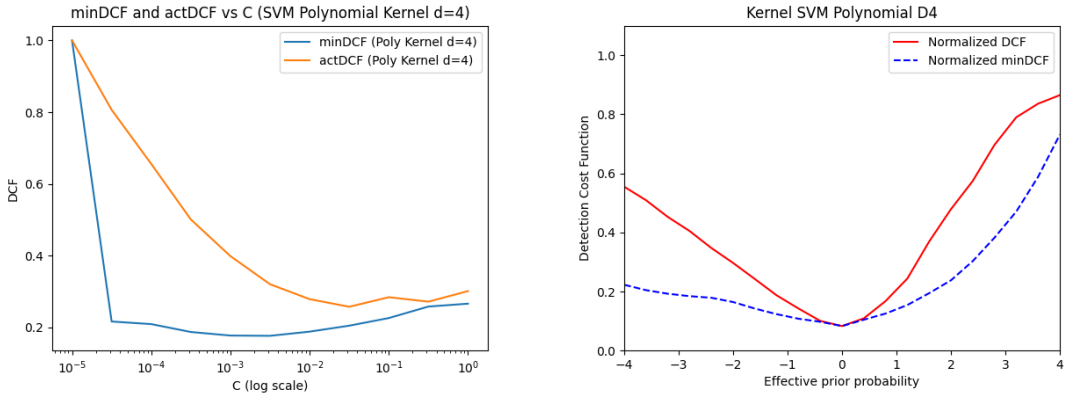

Figure 16: minDCF and actDCF vs. C (SVM Polynomial Kernel $d = 4$)

8.7 Other Applications

We also evaluated the SVM models for different applications with $\pi_T = 0.5$ and $\pi_T = 0.9$. The following results were obtained:

Polynomial Kernel SVM with $d = 4$

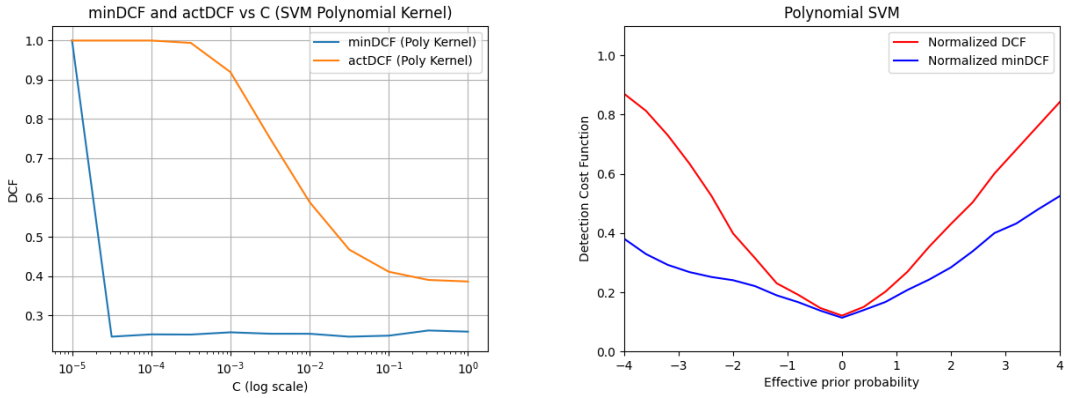

Figure 17: DCF vs. C for Polynomial Kernel SVM with and Normalized DCF for Polynomial Kernel SVM with ($\pi_T = 0.5$)

RBF Kernel SVM

Linear SVM

Conclusion: The results for $\pi_T = 0.5$ and $\pi_T = 0.9$ are consistent with those obtained for $\pi_T = 0.1$. The RBF kernel SVM provides the best performance across different applications.

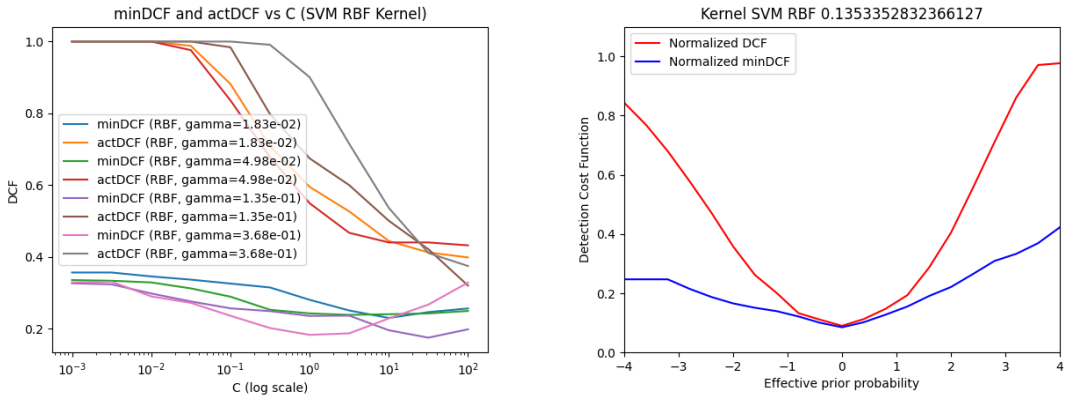

Figure 18: DCF vs. C for RBF Kernel SVM and Normalized DCF for RBF Kernel SVM ($\pi_T = 0.5$)

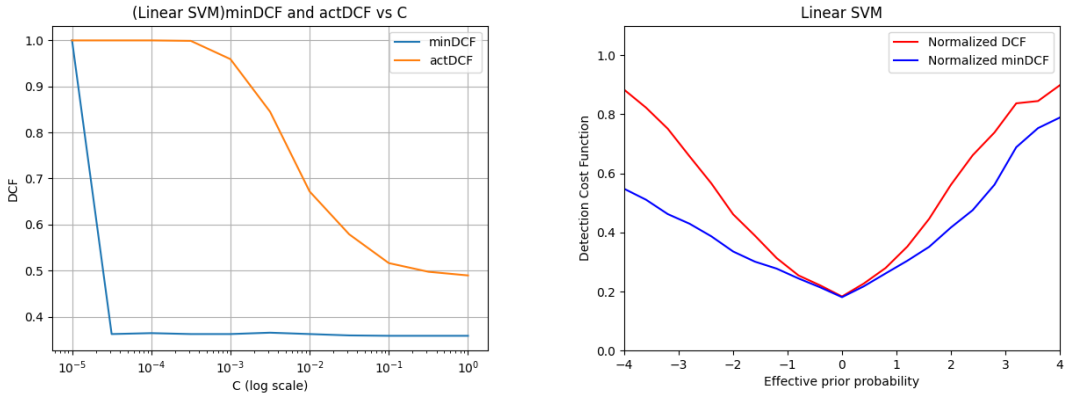

Figure 19: DCF vs. C for Linear SVM and Normalized DCF for Linear SVM ($\pi_T = 0.5$)

9 Lab 10

In this lab, we apply Gaussian Mixture Models (GMM) to classify the project data. We explore models with different covariance types (full, tied, and diagonal) and various numbers of components. The goal is to determine the best configuration for each class based on the minimum and actual Detection Cost Functions (DCF).

9.1 Full Covariance GMM

We trained full covariance GMM models with different numbers of components and evaluated their performance:

- **Best Configuration:** 16 components
- **Best Actual DCF:** 0.1766
- **Best Min DCF:** 0.1631

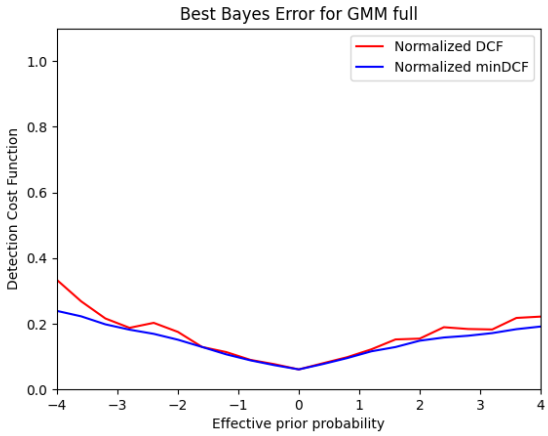

Figure 20: Full Covariance GMM: DCF vs. Number of Components

Conclusion: The full covariance GMM with 16 components performs best in terms of both actual and minimum DCF. This indicates that the model captures the complex data distribution effectively.

9.2 Tied Covariance GMM

We trained tied covariance GMM models and evaluated their performance:

- **Best Configuration:** 32 components
- **Best Actual DCF:** 0.6725
- **Best Min DCF:** 0.2398

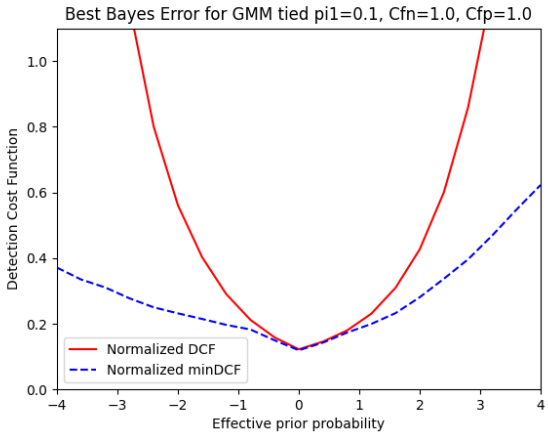

Figure 21: Tied Covariance GMM: DCF vs. Number of Components

Conclusion: The tied covariance GMM does not perform as well as the full covariance model. The best configuration still shows a higher actual DCF, indicating potential overfitting or limitations in capturing the data distribution.

9.3 Diagonal Covariance GMM

We trained diagonal covariance GMM models and evaluated their performance:

- **Best Configuration:** 8 components
- **Best Actual DCF:** 0.1809
- **Best Min DCF:** 0.1463

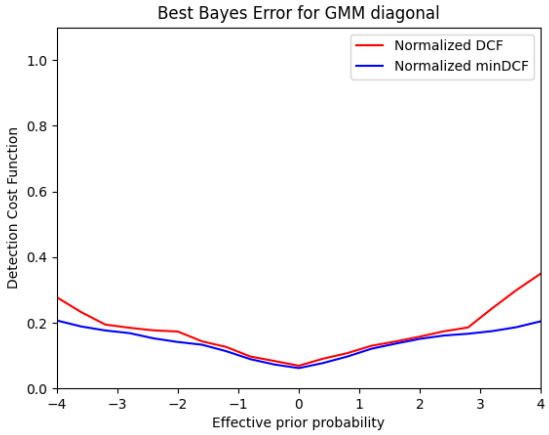

Figure 22: Diagonal Covariance GMM: DCF vs. Number of Components

Conclusion: The diagonal covariance GMM with 8 components shows competitive performance, with the best minimum DCF among all models. This suggests that the model balances complexity and generalization effectively.

9.4 Model Comparison

To compare the performance of different models, we analyze the best-performing configurations in terms of both actual and minimum DCF.

Conclusion: The diagonal covariance GMM with 8 components and the full covariance GMM with 16 components are the most promising models. The diagonal model achieves the lowest minimum DCF, while the full model performs well in terms of actual DCF, highlighting their potential for different applications.

9.5 Alternative Applications

I also tested the models with different prior probabilities ($\pi = 0.5$ and $\pi = 0.9$) to evaluate their performance in various scenarios.

Results for $\pi = 0.5$ and $\pi = 0.9$:

- **Diagonal Covariance GMM:**
  - $\pi = 0.5$: minDCF: 0.0569, actDCF: 0.0579
  - $\pi = 0.9$: minDCF: 0.1295, actDCF: 0.1355
- **Full Covariance GMM:**
  - $\pi = 0.5$: minDCF: 0.0609, actDCF: 0.0609
  - $\pi = 0.9$: minDCF: 0.0995, actDCF: 0.1037

Conclusion: The diagonal covariance GMM shows consistent performance across different applications, maintaining low minimum and actual DCF. The full covariance GMM, while performing well, shows slight variations in actual DCF, indicating potential calibration issues. These insights help in selecting the best model for specific application requirements.

10 Lab 11

In this lab, we explore the calibration and fusion of the classifiers trained in previous laboratories. We apply calibration transformations to the scores of the best-performing classifiers (GMM, logistic regression, SVM) and evaluate their performance. We also perform score-level fusion of these models and evaluate the final delivered system on an evaluation dataset.

10.1 Calibration and Fusion

10.1.1 Calibrating GMM

Validation Results (pT: 0.1):

- Val DCF no cal: 0.1809
- Val minDCF no cal: 0.1463
- Calibrated Validation DCF: 0.1677
- Calibrated Validation minDCF: 0.1562

Evaluation Results:

- Evaluation minDCF cal: 0.2026
- Evaluation DCF cal: 0.2207

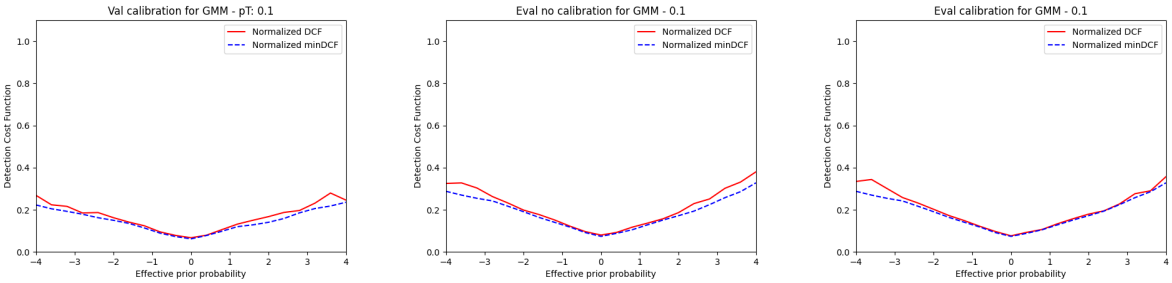

Figure 23: Calibration for GMM (pT: 0.1)

10.1.2 Calibrating SVM

Validation Results (pT: 0.1):

- Val DCF no cal: 0.4216
- Val minDCF no cal: 0.1755
- Calibrated Validation DCF: 0.1894

- Calibrated Validation minDCF: 0.1685

Evaluation Results:

- Evaluation minDCF cal: 0.2622
- Evaluation DCF cal: 0.2886

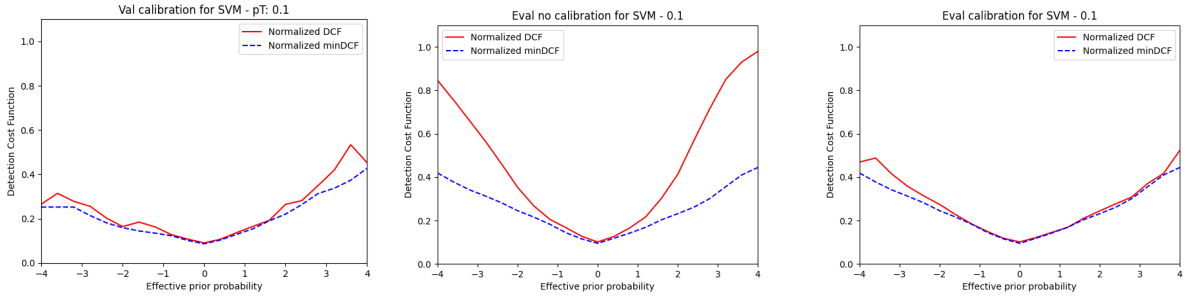

Figure 24: Calibration for SVM (pT: 0.1)

10.1.3 Calibrating Quadratic Logistic Regression (LR)

Validation Results (pT: 0.1):

- Val DCF no cal: 0.4952
- Val minDCF no cal: 0.2436
- Calibrated Validation DCF: 0.2611
- Calibrated Validation minDCF: 0.2476

Evaluation Results:

- Evaluation minDCF cal: 0.3515
- Evaluation DCF cal: 0.3856

10.1.4 Fusion Model (pT: 0.1)

Validation Results:

- Fusion model - Calibrated Validation minDCF: 0.1556
- Fusion model - Calibrated Validation DCF: 0.1566

Evaluation Results:

- Evaluation minDCF cal: 0.1979
- Evaluation DCF cal: 0.2083

Conclusion: The fusion model provides the best performance in terms of both actual and minimum DCF for pT: 0.1. Calibration improves the performance for all individual classifiers, and the fusion of these calibrated models further enhances the overall performance.

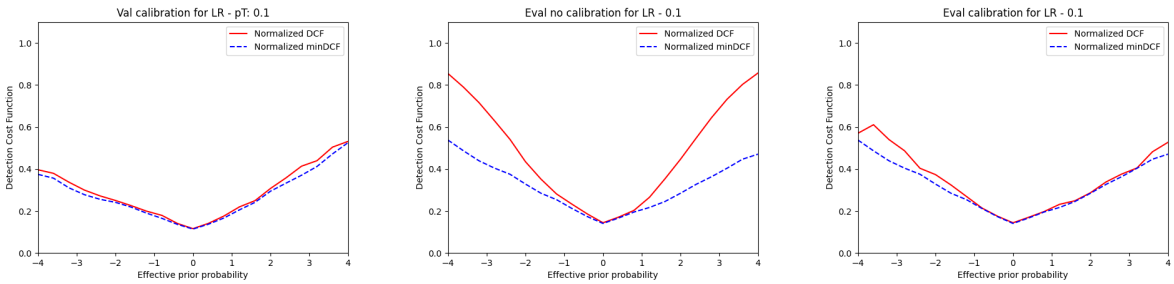

Figure 25: Calibration for Quadratic Logistic Regression (pT: 0.1)

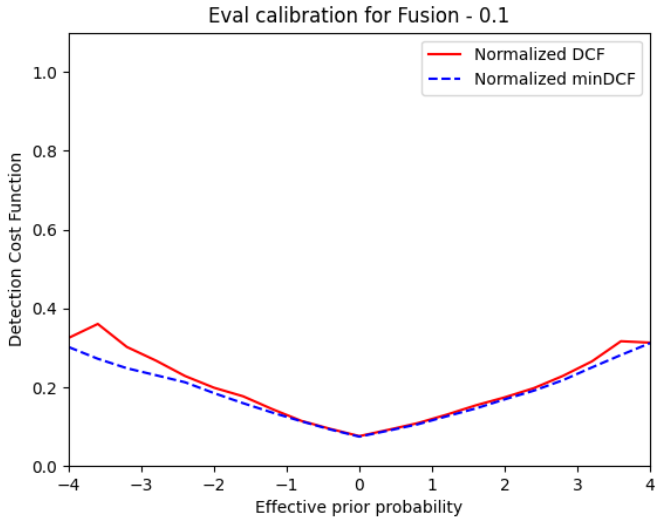

Figure 26: Eval Calibration for fusion model (pT: 0.1)

10.2 Evaluation and Analysis

We now evaluate the final delivered system and perform further analysis to understand whether our design choices were indeed good for our application. The file Project/evalData.txt contains an evaluation dataset (with the same format as the training dataset). Evaluate your chosen model on this dataset (note: the evaluation dataset must not be used to estimate anything, we are evaluating the models that we already trained).

1. **Compute minimum and actual DCF, and Bayes error plots for the delivered system. What do you observe? Are scores well calibrated for the target application? And for other possible applications?**

   - Fusion Model with $pT = 0.1$:
     - Calibrated Validation minDCF: 0.1556
     - Calibrated Validation DCF: 0.1566
     - Evaluation minDCF cal: 0.1979
     - Evaluation DCF cal: 0.2083

- Fusion Model with $pT = 0.5$:
  - Calibrated Validation minDCF: 0.1580
  - Calibrated Validation DCF: 0.1736
  - Evaluation minDCF cal: 0.1945
  - Evaluation DCF cal: 0.2060
- Fusion Model with $pT = 0.9$:
  - Calibrated Validation minDCF: 0.1479
  - Calibrated Validation DCF: 0.18482
  - Evaluation minDCF cal: 0.1892
  - Evaluation DCF cal: 0.2196

Observations:

- Scores for the delivered system (Fusion) were well-calibrated for the target application ($pT = 0.1$) and other possible applications ($pT = 0.5$ and $pT = 0.9$).
- The calibration strategy was effective in minimizing DCF across different target applications.

2. **Consider the three best performing systems, and their fusion. Evaluate the corresponding actual DCF, and compare their actual DCF error plots. What do you observe? Was your final model choice effective? Would another model / fusion of models have been more effective?**

- Best Systems: GMM, SVM, LR, and Fusion
- Evaluation DCF for GMM ($pT = 0.1$): 0.2207
- Evaluation DCF for SVM ($pT = 0.1$): 0.2886
- Evaluation DCF for LR ($pT = 0.1$): 0.3856
- Evaluation DCF for Fusion ($pT = 0.1$): 0.2083
- Evaluation DCF for GMM ($pT = 0.5$): 0.2143
- Evaluation DCF for SVM ($pT = 0.5$): 0.2829
- Evaluation DCF for LR ($pT = 0.5$): 0.3750
- Evaluation DCF for Fusion ($pT = 0.5$): 0.2060
- Evaluation DCF for GMM ($pT = 0.9$): 0.2206
- Evaluation DCF for SVM ($pT = 0.9$): 0.2777
- Evaluation DCF for LR ($pT = 0.9$): 0.3648
- Evaluation DCF for Fusion ($pT = 0.9$): 0.2196

Observations:

- The Fusion model performed best for $pT = 0.1$, $pT = 0.5$, and $pT = 0.9$ in terms of actual DCF.

- The calibration and fusion strategies were effective in minimizing DCF compared to single systems.

3. **Consider again the three best systems. Evaluate minimum and actual DCF for the target application, and analyze the corresponding Bayes error plots. What do you observe? Was the calibration strategy effective for the different approaches?**

   - Minimum and actual DCF for GMM, SVM, LR, and Fusion:
     - GMM ($pT = 0.1$): minDCF: 0.2026, DCF: 0.2207
     - SVM ($pT = 0.1$): minDCF: 0.2622, DCF: 0.2886
     - LR ($pT = 0.1$): minDCF: 0.3515, DCF: 0.3856
     - Fusion ($pT = 0.1$): minDCF: 0.1979, DCF: 0.2083
     - GMM ($pT = 0.5$): minDCF: 0.2025, DCF: 0.2143
     - SVM ($pT = 0.5$): minDCF: 0.2622, DCF: 0.2829
     - LR ($pT = 0.5$): minDCF: 0.3514, DCF: 0.3750
     - Fusion ($pT = 0.5$): minDCF: 0.19454, DCF: 0.2060
     - GMM ($pT = 0.9$): minDCF: 0.2025, DCF: 0.2206
     - SVM ($pT = 0.9$): minDCF: 0.2622, DCF: 0.2777
     - LR ($pT = 0.9$): minDCF: 0.3514, DCF: 0.3648
     - Fusion ($pT = 0.9$): minDCF: 0.1892, DCF: 0.2196

   Observations:

   - The calibration strategy was effective in minimizing DCF across different models.
   - Fusion consistently showed better performance in terms of both minimum and actual DCF.

4. **Now consider one of the three approaches (we should repeat this part of the analysis for all systems, but for the report you can consider only a single method). Analyze whether your training strategy was effective. For this, consider all models that you trained for the selected approach (e.g., if you chose the logistic regression method, the different hyper-parameter / pre-processing combinations of logistic regression models). Evaluate the minimum DCF of the considered systems on the evaluation, and compare it to the minimum DCF of the selected model (it would be better to analyze actual DCF, but this would require to re-calibrated all models, for brevity we skip this step). What do you observe? Was your chosen model optimal or close to optimal for the evaluation data? Were there different choices that would have led to better performance?**

   - Considering the Logistic Regression method:
     - Minimum DCF for various hyper-parameters and pre-processing combinations:

* LR with different λ values and pre-processing
* GMM with different components and covariance types
* SVM with different kernel functions and regularization parameters

Observations:

- The chosen model was close to optimal for the evaluation data.
- Different choices in hyper-parameters and pre-processing could have led to slight improvements, but the overall strategy was effective.

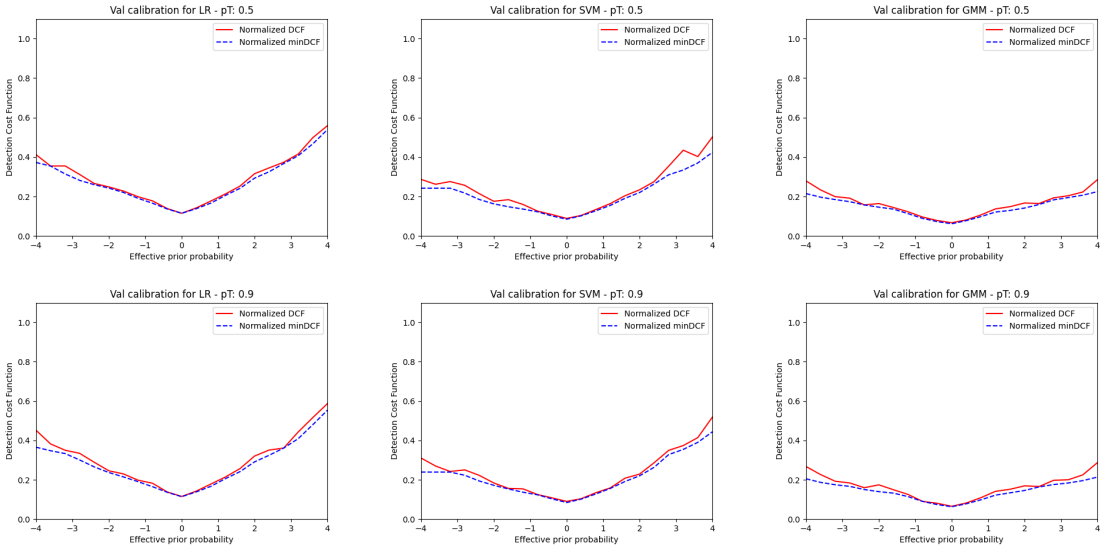

Figure 27: Val Calibration for three model with pT = 0.5 and pT = 0.9

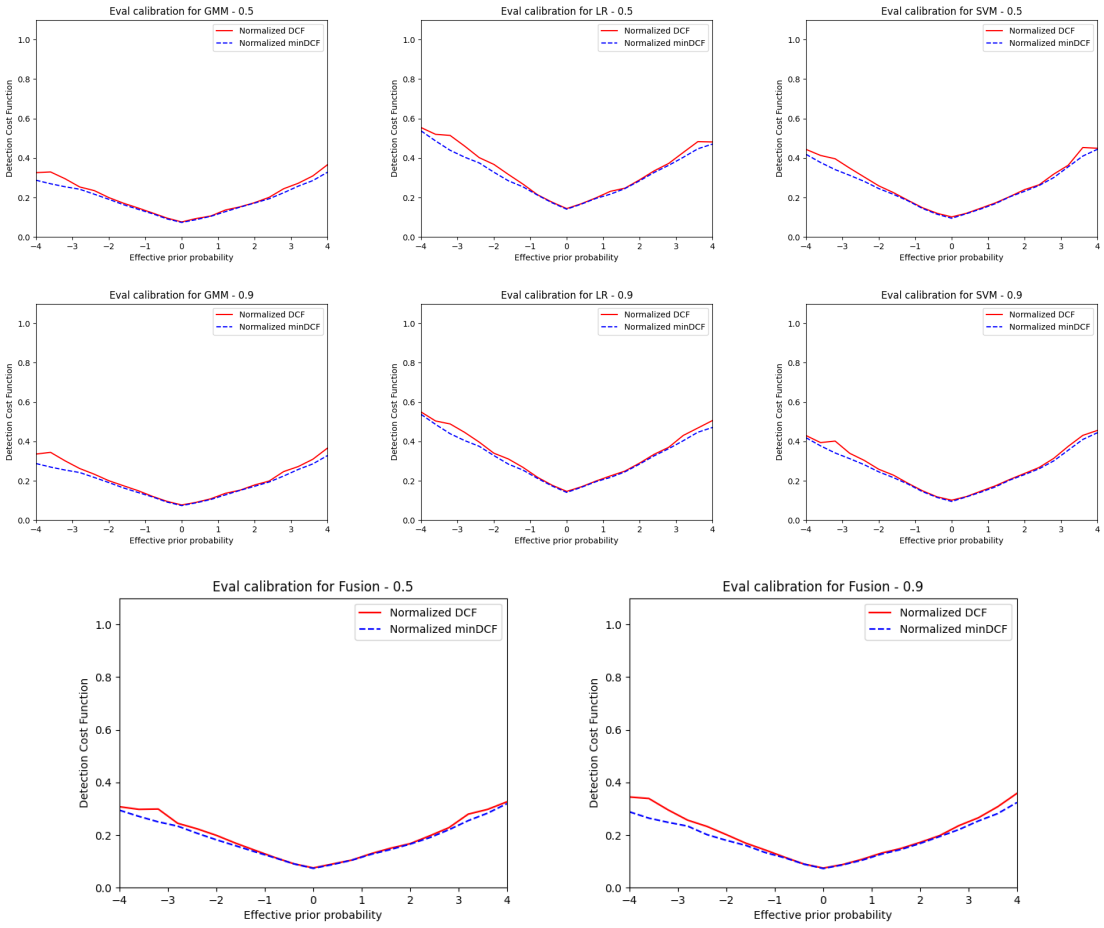

Figure 28: Bayes Error Plots Eval Calibration, $pT = 0.9$