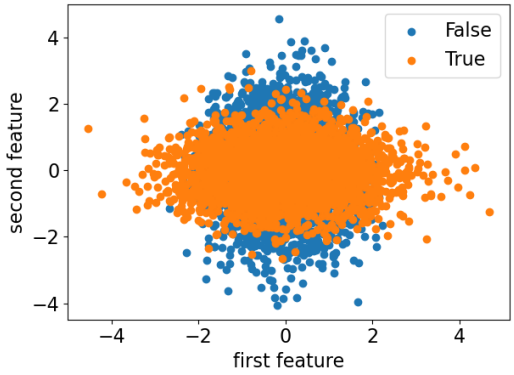
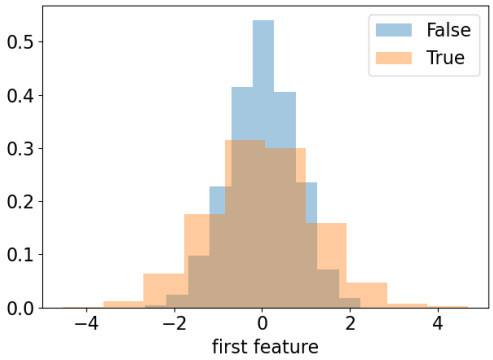
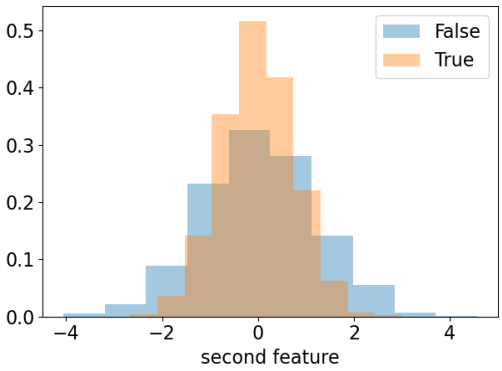
In the first phase of the project, the distribution of six features extracted from fingerprint reader data was visualized. Readings in the dataset are labeled as either 'True' or 'False' based on the authentication result from the reader. Histograms were used to show the distribution of these features in both classes. These histograms provide information on how each feature changes with respect to the 'True' and 'False' labels, offering a preliminary idea about their discrimination power. By examining these distributions, patterns that could help drive the classification model to high accuracy were identified.

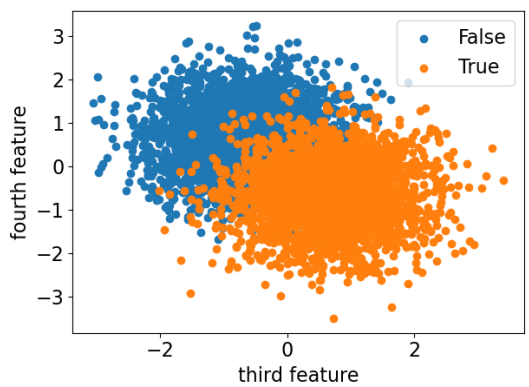
Analyzing the first two features:
The first two features exhibit substantial overlap between the two classes, as shown in the scatter plot. Both classes appear to be centered around the origin. The first feature shows very different variances between the two classes, while the second feature has more comparable variances. The means are similar but slightly different.
First feature:
Class 0: Mean = 0.002877, Variance = 0.569581, one peak
Class 1: Mean = 0.000545, Variance = 1.430233, one peak
Second feature:
Class 0: Mean = 0.018693, Variance = 1.420866, one peak
Class 1: Mean = -0.008524, Variance = 0.578278, one peak



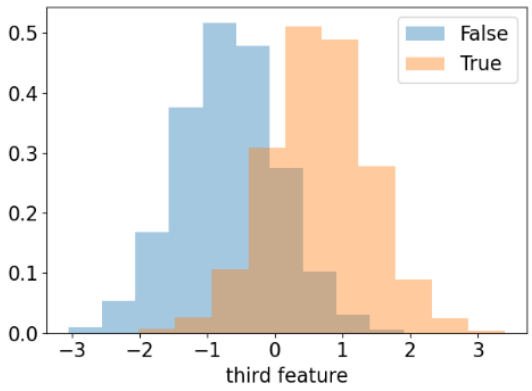


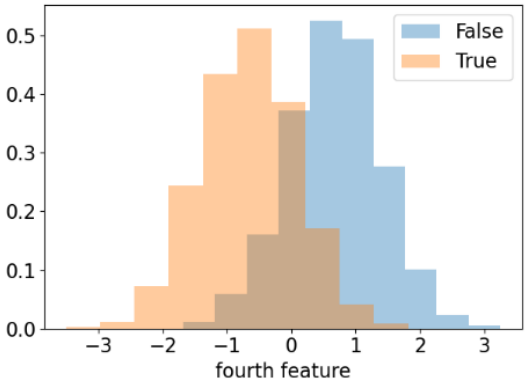


Analyzing the third and fourth features:
The third and fourth features show clear separation between the classes, with minimal overlap. Both the means and variances for the third and fourth features are quite distinct between the two classes, showing clear separation.
Third feature:
Class 0: Mean = -0.680940 Variance = 0.549977, single peak
Class 1: Mean = 0.665238, Variance = 0.548903, single peak.
Fourth feature:
Class 0: Mean = 0.670836, Variance = 0.536043, single peak.
Class 1: Mean = -0.664195, Variance = 0.553343,

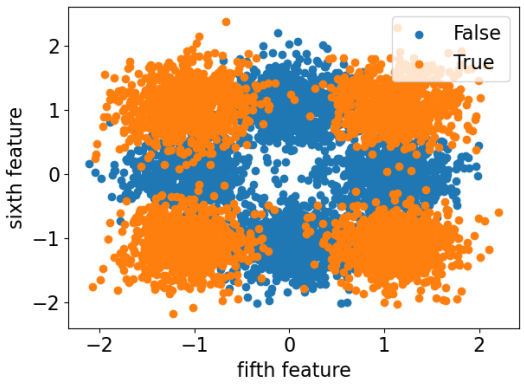


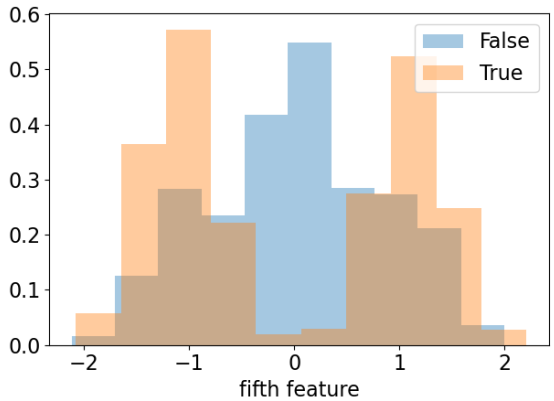
single peak.

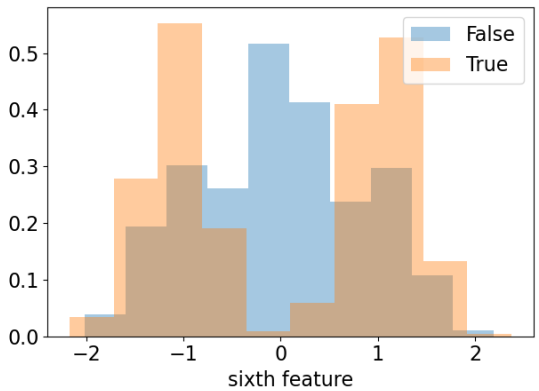




Analyzing the last two features:
The last two features have noticeable overlap between the classes, but there are distinct clustering patterns that can be observed. The variances are different, especially for the fifth feature.
Fifth feature:
Class 0: Mean = 0.027957, Variance = 0.680074, multiple peaks, indicating multiple modes
Class 1: Mean = -0.041725, Variance = 1.317768, multiple peaks, indicating multiple modes
Sixth feature:
Class 0: Mean = -0.005827, Variance = 0.705038, multiple peaks, indicating multiple modes
Class 1: Mean = 0.023938, Variance = 1.287026, multiple peaks, indicating multiple modes

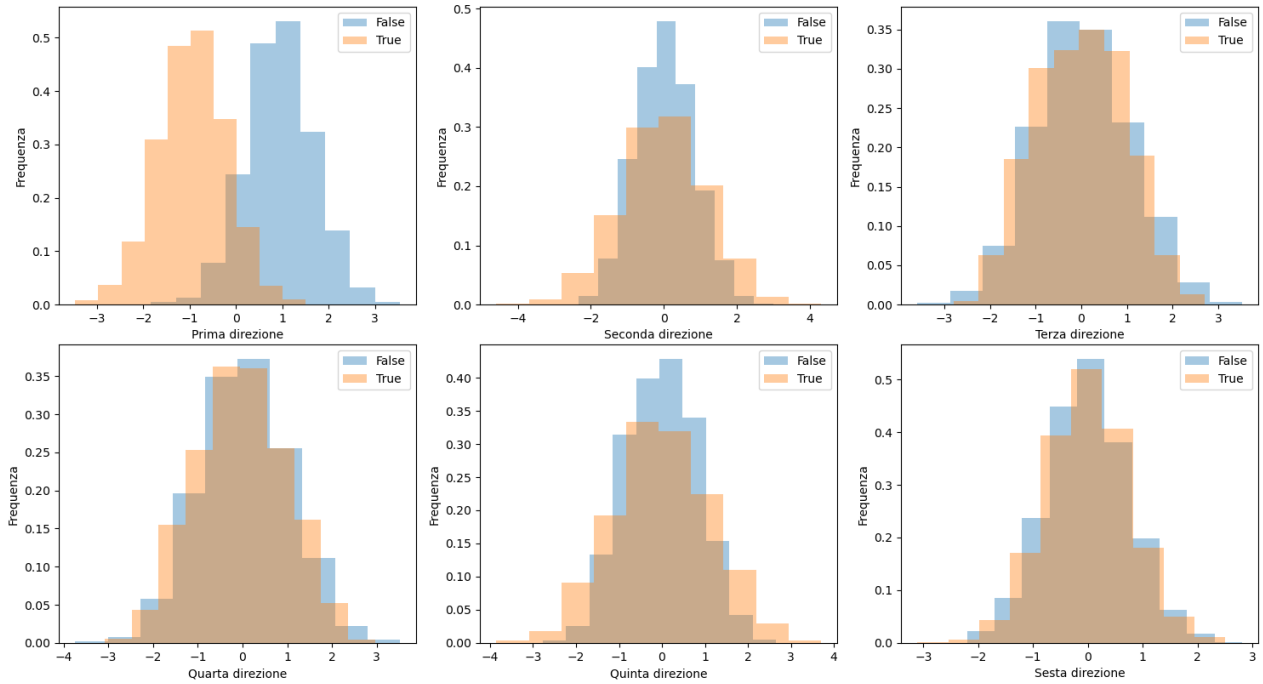




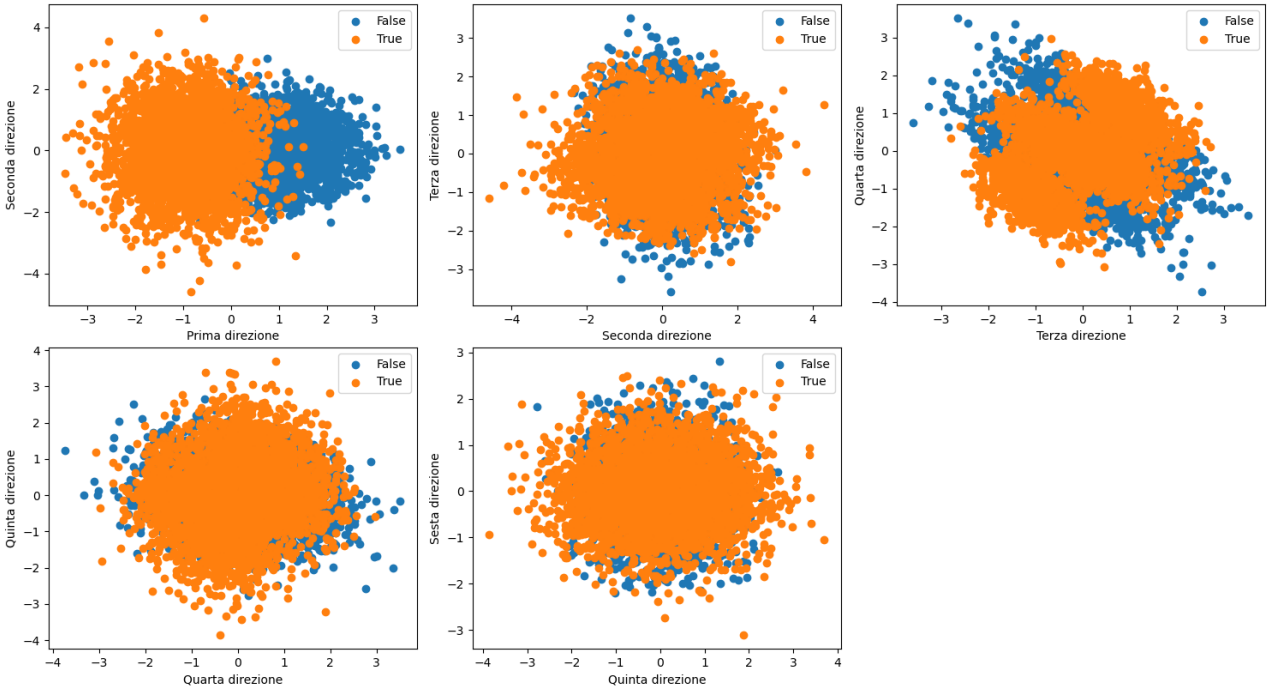


Dimensionality reduction

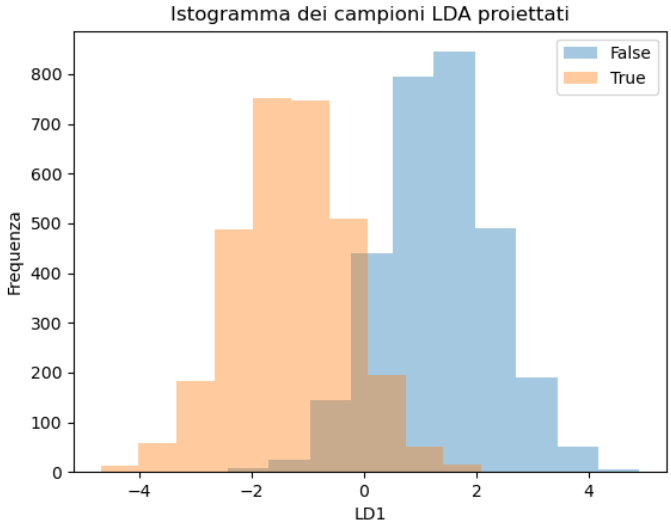
The analysis begins with the application of PCA to the project data, followed by plotting histograms of the projected features along the first 6 PCA directions. Post-PCA application, the principal components exhibit more balanced variances across the classes, suggesting that PCA has redistributed the variance among the components to maximize variance along the principal axes. Additionally, the means of the principal components are closer to zero in contrast to those of the original features. This outcome aligns with expectations, as PCA centers the data prior to transformation.



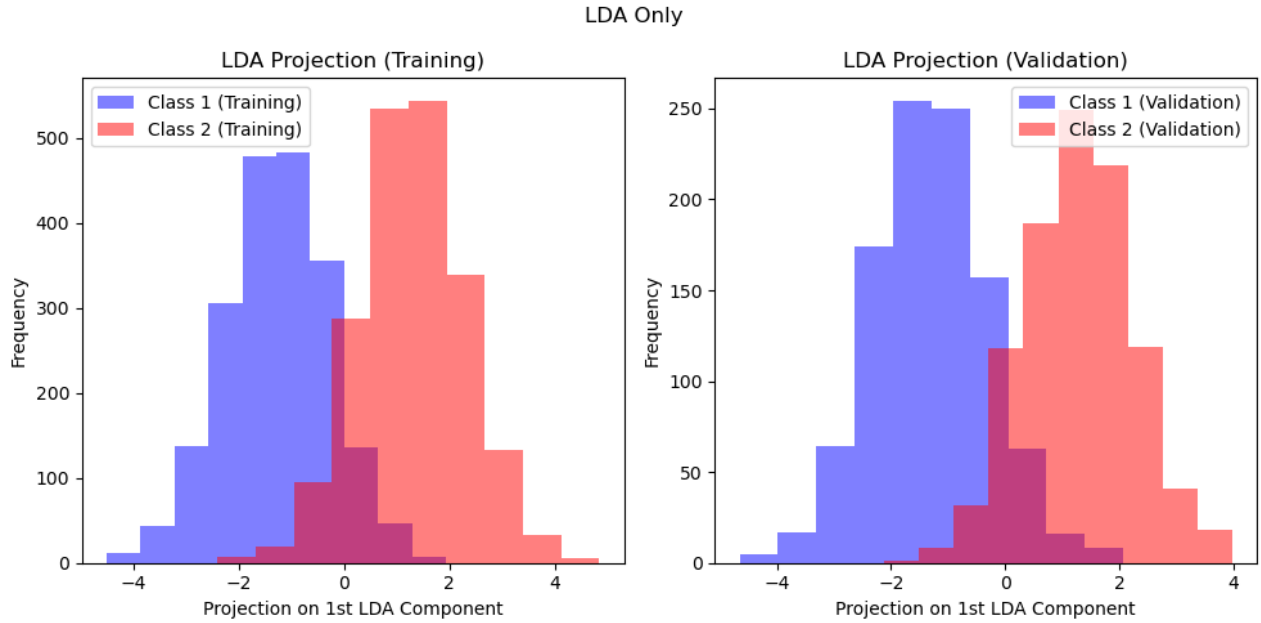
PCA helps to reveal potential clusters within each class by projecting the data onto new axes that maximize variance. By examining scatter plots of the principal components, it is possible to inspect the presence of clusters in the first scatter plot, therefore, there is a noticeable separation between the two classes (True and False). True data points are more concentrated around the negative side of the first dimension, while False points are clustered around the positive side.



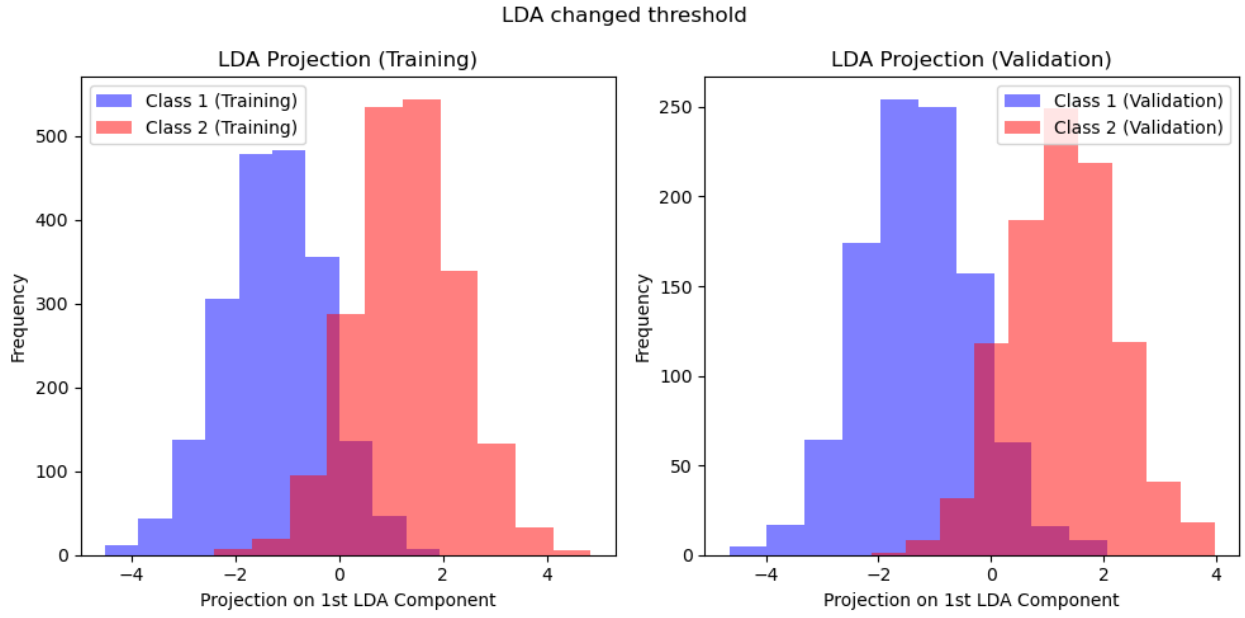
PCA has effectively reduced the dimensionality of the data while preserving the variance. This makes it easier to visualize and analyze the data, especially for identifying clusters and separating the classes. PCA has transformed the data to highlight the directions of maximum variance, helping to potentially separate the classes better and reveal underlying clusters within each class.
Applying LDA (1 dimensional) and computing the histogram of the projected LDA sample it is possible to observe that LDA has found a direction (LD1) that provides a significant separation between the two classes, as indicated by the distinct peaks in the histogram. However, the presence of some overlap around the LD1 value of 0 suggests that the separation is not perfect. This is often expected in real-world data, where classes may not be completely linearly separable. Overall, LDA appears to be effective in finding a direction with a relatively low degree of class overlap, but not perfect separation.



I applied LDA as a classifier and divided the dataset into model training and validation sets. I applied LDA and selected the orientation that resulted in the projected mean of class True (label 1) being larger than the projected mean of class False (label 0). I then selected the threshold as in the previous sections, as the average of the projected class means. Finally, I computed the predictions on the validation data and calculated the corresponding error rate.
LDA Only:
threshold: 0.018534376786207174
Error rate: 9.3%

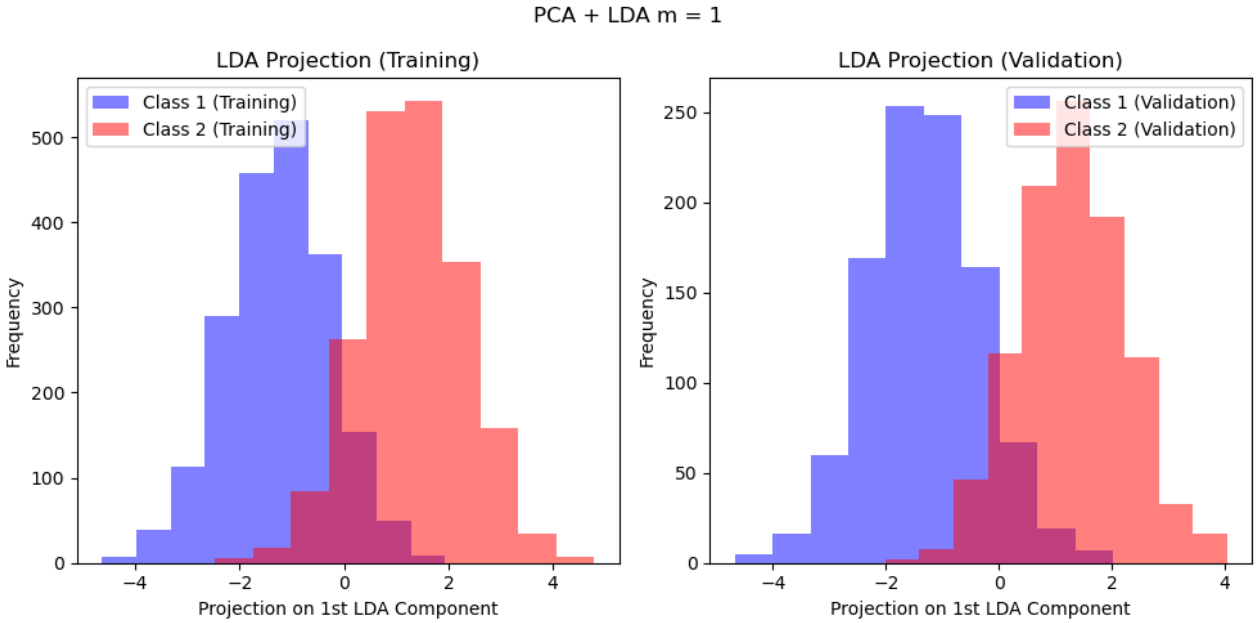


Changing the threshold, with a lower one, it is possible to observe a decreasing of the error rate
LDA changed threshold:
threshold: 0.003706875357241435
Error rate: 9.2%

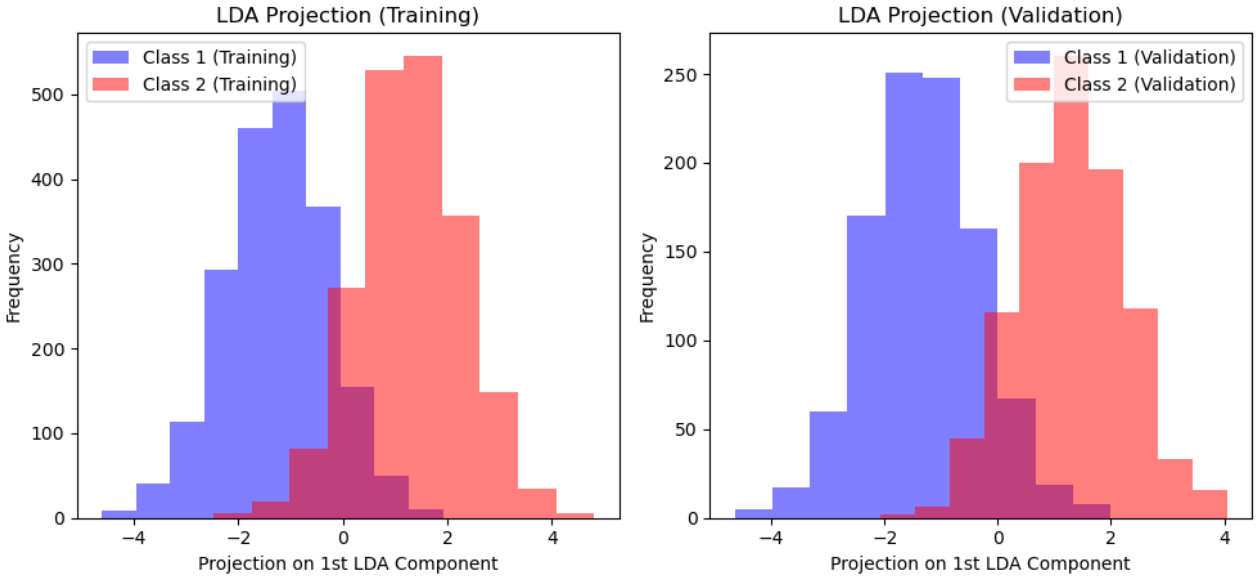


Finally, applying pre-processing technique (PCA) to the features and then classifying the validation data with LDA, we want to analyze the performance as a function of the number of PCA dimensions *m*.

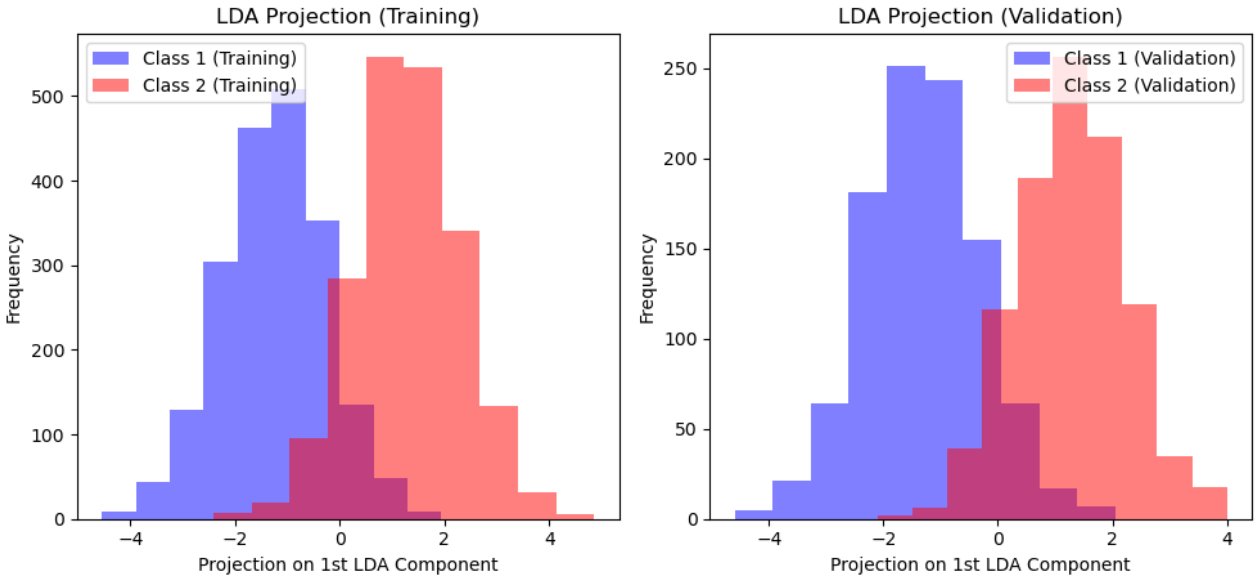
PCA + LDA, with $m = 1$:
Error rate: 9.35%



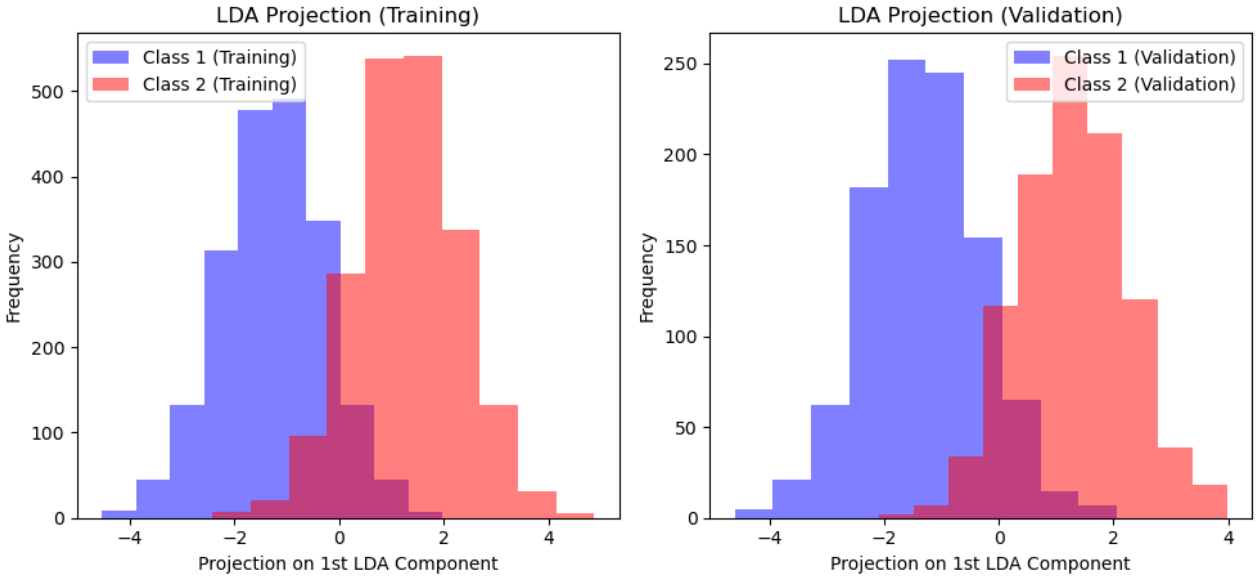
PCA + LDA, with $m = 2$:
Error rate: 9.25%



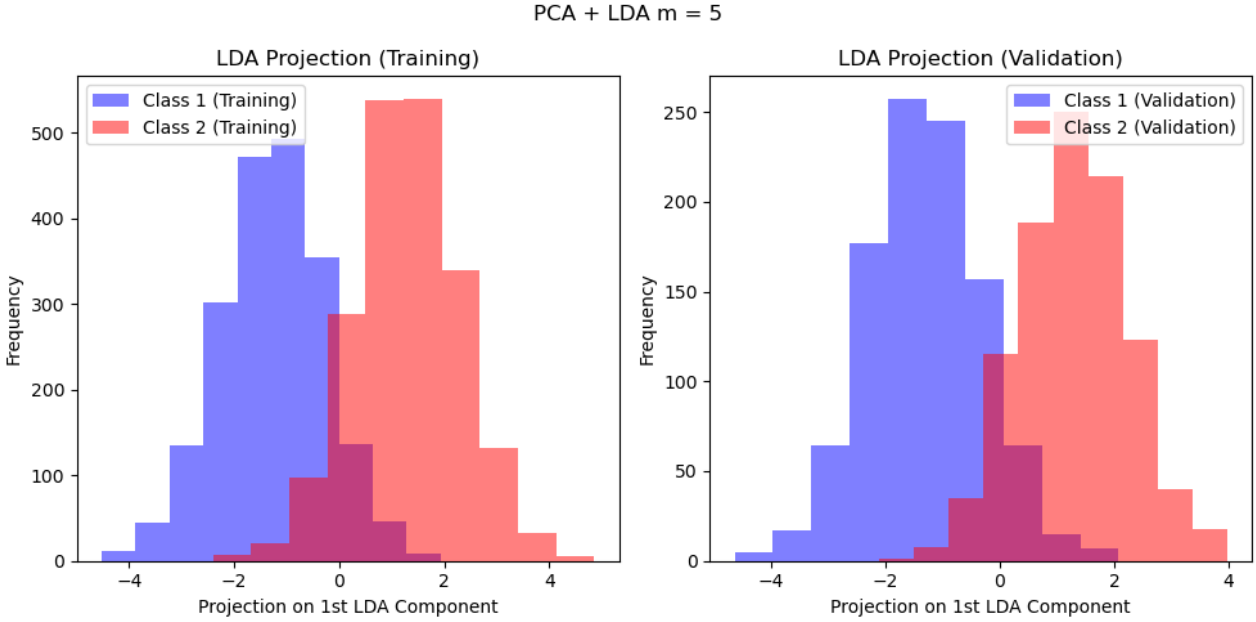
PCA + LDA, with $m = 3$:
Error rate: 9.25%



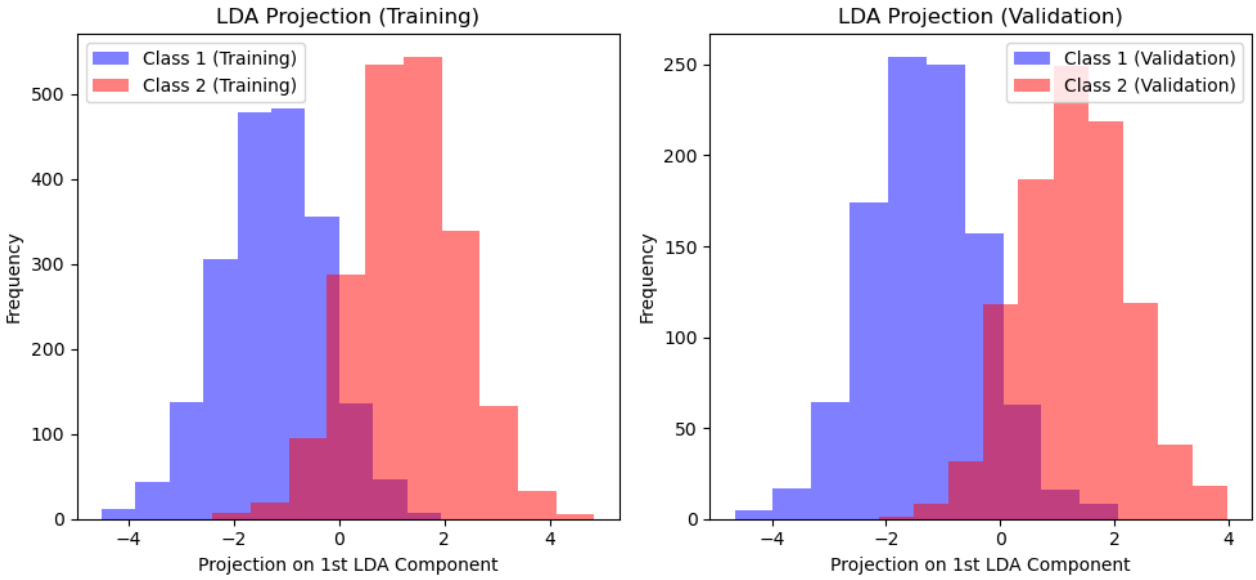
PCA + LDA, with $m = 4$:
Error rate: 9.25%



PCA + LDA, with $m = 5$:
Error rate: 9.30%

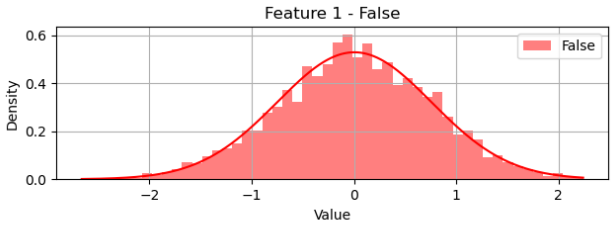


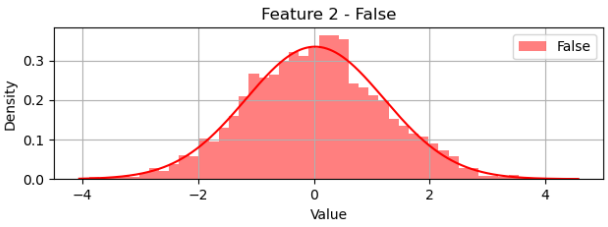
PCA + LDA, with $m = 6$:
Error rate: 9.30%

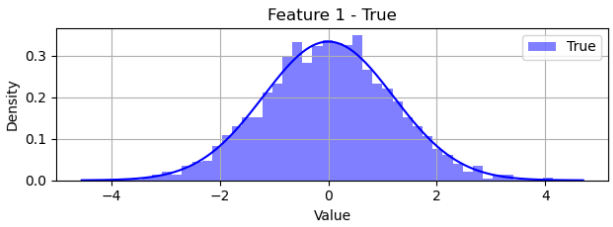


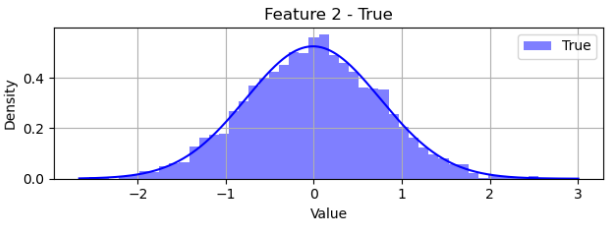
Observing the histogram plots it is possible to notice that the best value of m of error rate are (2, 3, 4), finally, PCA is a beneficial for the task when combined with the LDA classifier.

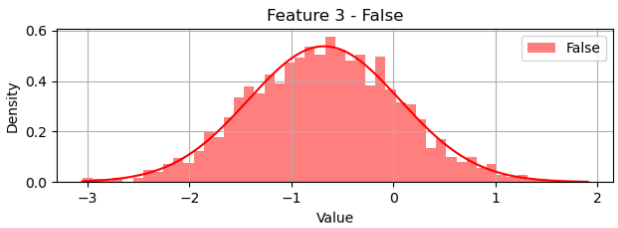
Uni-variate Gaussian models were fitted to the different features of the various classes in the project dataset. For each class and for each component of the feature vector, the Maximum Likelihood (ML) estimate for the parameters of a 1D Gaussian distribution was computed.
As shown in the plots, the Gaussian density fits well with features 1 through 4, whereas features 5 and 6 do not exhibit an accurate fit.

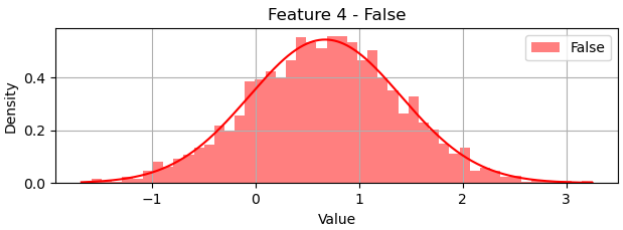


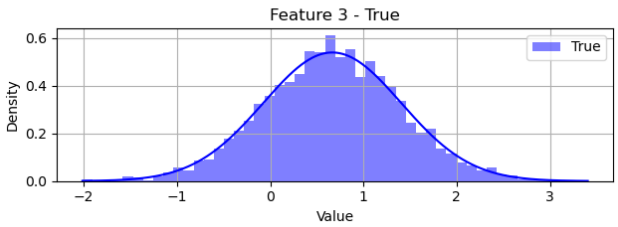


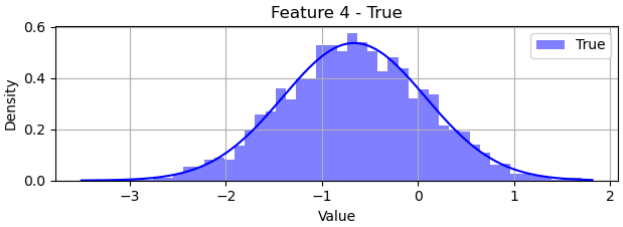


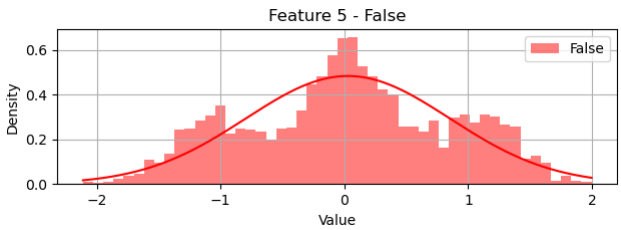


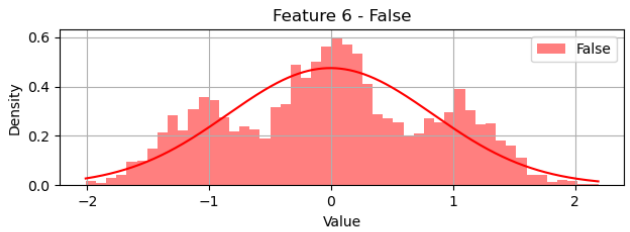


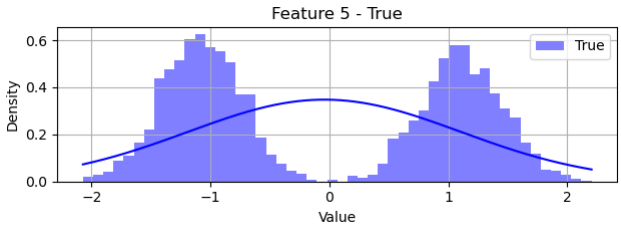


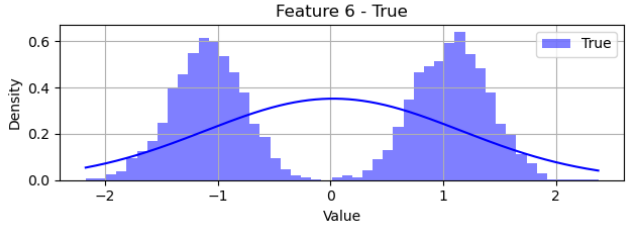












I applied the MVG model to the project data and divided the dataset into model training and validation subsets. I trained the model parameters on the training portion of the dataset and computed LLRs for the validation subset. Predictions were obtained from the LLRs, and the corresponding error rate was computed. Subsequently, I applied the tied Gaussian model and compared the results with those from the MVG and LDA models. Next, I tested the Naive Bayes Gaussian model. The calculation of the error rate for the different models shows that the MultiVariate Gaussian model performs the best. The Naïve Bayes Gaussian model performs in an efficient way compared to LDA and Tied Gaussian model, its error rate is significantly similar to the MVG one.

LDA – Error rate: 9.3%
MultiVariate Gaussian - Error rate: 7.0%
Tied Gaussian - Error rate: 9.3%
Naive Bayes Gaussian - Error rate: 7.2%

I printed the covariance matrix of each class, showing variances for different features on the diagonal and feature covariances outside the diagonal. The covariance values are smaller and differ by a couple of orders of magnitude from those of the variance.

Class false MVG covariance matrix:
[[6.010e-01 5.159e-05 1.906e-02 1.925e-02 1.280e-02 -1.347e-02]
[5.159e-05 1.447e+00 -1.613e-02 -1.586e-02 -2.645e-02 2.291e-02]
[1.906e-02 -1.613e-02 5.653e-01 -1.843e-03 -6.914e-03 1.689e-02]
[1.925e-02 -1.586e-02 -1.843e-03 5.416e-01 5.252e-03 1.357e-02]
[1.280e-02 -2.645e-02 -6.914e-03 5.252e-03 6.961e-01 1.584e-02]
[-1.347e-02 2.291e-02 1.689e-02 1.357e-02 1.584e-02 6.865e-01]]
Class true MVG covariance matrix:
[[1.448e+00 -1.472e-02 5.570e-03 1.574e-02 1.950e-02 -1.767e-04]
[-1.472e-02 5.534e-01 -1.122e-02 -9.065e-03 -1.466e-02 1.635e-02]
[5.570e-03 -1.122e-02 5.575e-01 2.756e-02 -3.770e-03 -1.460e-02]
[1.574e-02 -9.065e-03 2.756e-02 5.697e-01 -1.170e-02 3.499e-02]
[1.950e-02 -1.466e-02 -3.770e-03 -1.170e-02 1.342e+00 1.695e-02]
[-1.767e-04 1.635e-02 -1.460e-02 3.499e-02 1.695e-02 1.304e+00]]

To better visualize the strength of covariances relative to variances, I computed the Pearson correlation coefficient for each pair of features. The correlation matrix has diagonal elements equal to 1, while off-diagonal elements correspond to the correlation coefficients for all feature pairs.

Corr false:
[[1.000e+00 5.532e-05 3.270e-02 3.375e-02 1.980e-02 -2.097e-02]
[5.532e-05 1.000e+00 -1.784e-02 -1.791e-02 -2.636e-02 2.299e-02]
[3.270e-02 -1.784e-02 1.000e+00 -3.331e-03 -1.102e-02 2.712e-02]
[3.375e-02 -1.791e-02 -3.331e-03 1.000e+00 8.553e-03 2.226e-02]
[1.980e-02 -2.636e-02 -1.102e-02 8.553e-03 1.000e+00 2.292e-02]
[-2.097e-02 2.299e-02 2.712e-02 2.226e-02 2.292e-02 1.000e+00]]

The off-diagonal elements are very close to zero, indicating weak correlations between different pairs of features.

Corr true:

[[1.000e+00 -1.645e-02 6.199e-03 1.733e-02 1.399e-02 -1.286e-04]
[-1.645e-02 1.000e+00 -2.019e-02 -1.614e-02 -1.701e-02 1.925e-02]
[6.199e-03 -2.019e-02 1.000e+00 4.891e-02 -4.358e-03 -1.712e-02]
[1.733e-02 -1.614e-02 4.891e-02 1.000e+00 -1.338e-02 4.061e-02]
[1.399e-02 -1.701e-02 -4.358e-03 -1.338e-02 1.000e+00 1.281e-02]
[-1.286e-04 1.925e-02 -1.712e-02 4.061e-02 1.281e-02 1.000e+00]]

The off-diagonal elements are also quite close to zero, showing weak correlations between different pairs of features. There are no strong correlations observed among the features, as none of the off-diagonal values approach ±1. Naive Bayes classifiers assume that the features are conditionally independent given the class label. This means that the algorithm performs best when features are not correlated. Given that the correlation matrices indicate weak correlations between features, the Naive Bayes assumption of independence is reasonably satisfied in this case. This weak correlation helps explain why the Naive Bayes model might perform well on this dataset. The minimal interaction between features aligns well with the independence assumption, leading to better predictive performance.

The Gaussian model assumes that features can be jointly modelled by Gaussian distributions. Therefore, the model's goodness is strongly influenced by the accuracy of this assumption. Although visualizing 6-dimensional distributions is unfeasible, the assumption can be analyzed for single features or pairs of features.

Feature 1:
Class 0 - Gaussian Params: (mu=-0.01, sigma=0.78), Actual Mean: 0.00, Actual Std: 0.75
Class 1 - Gaussian Params: (mu=-0.00, sigma=1.20), Actual Mean: 0.00, Actual Std: 1.20

Feature 2:
Class 0 - Gaussian Params: (mu=0.02, sigma=1.20), Actual Mean: 0.02, Actual Std: 1.19
Class 1 - Gaussian Params: (mu=-0.01, sigma=0.74), Actual Mean: -0.01, Actual Std: 0.76

Feature 3:
Class 0 - Gaussian Params: (mu=-0.67, sigma=0.75), Actual Mean: -0.68, Actual Std: 0.74
Class 1 - Gaussian Params: (mu=0.66, sigma=0.75), Actual Mean: 0.67, Actual Std: 0.74

Feature 4:
Class 0 - Gaussian Params: (mu=0.68, sigma=0.74), Actual Mean: 0.67, Actual Std: 0.73
Class 1 - Gaussian Params: (mu=-0.66, sigma=0.75), Actual Mean: -0.66, Actual Std: 0.74

Feature 5:
Class 0 - Gaussian Params: (mu=0.02, sigma=0.83), Actual Mean: 0.03, Actual Std: 0.82
Class 1 - Gaussian Params: (mu=-0.04, sigma=1.16), Actual Mean: -0.04, Actual Std: 1.15

Feature 6:
Class 0 - Gaussian Params: (mu=-0.01, sigma=0.83), Actual Mean: -0.01, Actual Std: 0.84
Class 1 - Gaussian Params: (mu=0.02, sigma=1.14), Actual Mean: 0.02, Actual Std: 1.13

The Gaussian assumption appears to be quite accurate for all the first four features. The provided Gaussian parameters (mean and standard deviation) closely match the actual mean and standard deviation for these feature and class combination. This indicates that the Gaussian model is a good fit for the data. The Gaussian assumption does not appear to be quite accurate for last two features. To investigate whether the last set of features negatively affects the classifier due to poor modelling assumptions, I repeated the classification using only features 1 to 4 (discarding the last 2 features). This analysis was performed for the three models.

Tied Gaussian - Error rate: 9.5%
Naive Bayes Gaussian - Error rate: 7.6%
MultiVariate Gaussian - Error rate: 8.0%

In summary, despite features five and six not fitting well to the Gaussian distribution, the multivariate Gaussian model may still benefit from these features to improve overall classification accuracy. The discrepancy between the error rate using all six features (7%) and only the first four features (8%) suggests that these latter two features positively contribute to the predictive capability of the model, despite their limitations in conforming to the Gaussian distribution.
For features 1 and 2, the means are similar, but variances differ.

Tied Gaussian - Error rate (1,2): 49.5%
MultiVariate Gaussian - Error rate (1,2): 36.5%

The Multivariate Gaussian model is better due to its ability to handle varying variances between classes, unlike the Tied Gaussian model which assumes equal variances.
For features 3 and 4, the two classes mainly differ in feature means but have similar variances. Furthermore, the features show limited correlation for both classes.

Tied Gaussian - Error rate (3,4): 9.4%
MultiVariate Gaussian - Error rate (3,4): 9.4%

I analyzed the impact of these feature distribution characteristics on the performance of different approaches. I repeated the classification using only features 1-2 (jointly), and then using only features 3-4 (jointly), comparing the results of the MVG and tied MVG models.
Both the Tied Gaussian and Multivariate Gaussian models perform equally well for the features three and four, demonstrating their effectiveness when variances are similar, and correlation is limited.

In summary, the choice between Tied Gaussian and Multivariate Gaussian models depends on the variance structure and correlation between features. The Multivariate Gaussian model excels when variances differ between classes (first case), while both models perform similarly when variances are similar (second case), with the Multivariate Gaussian model offering a more flexible approach to capture covariance structure.

Feature 1
MultiVariate Gaussian - Error rate: 37.1%
Feature 2
MultiVariate Gaussian - Error rate: 39.8%
Feature 3
MultiVariate Gaussian - Error rate: 17.0%

Feature 4
MultiVariate Gaussian - Error rate: 18.6%
Feature 5
MultiVariate Gaussian - Error rate: 34.1%
Feature 6
MultiVariate Gaussian - Error rate: 36.6%

Finally, I analyzed the effects of PCA as a pre-processing step. PCA was used to reduce the dimensionality of the feature space, and I applied the three classification approaches.

Tied Gaussian - Error rate (PCA): 9.3%
Naive Bayes Gaussian - Error rate (PCA): 8.9%
MultiVariate Gaussian - Error rate (PCA): 7.0%

Tied Gaussian Model (PCA) did not significantly affect the performance of the Tied Gaussian model, as the error rate remained the same. This could be because the Tied Gaussian model primarily focuses on equal variances and may not benefit as much from dimensionality reduction via PCA. Naive Bayes Gaussian Model (PCA), the error rate increased from 7.2% without PCA to 8.9% with PCA. This indicates that PCA might have removed some discriminative information that the Naive Bayes model relied on, leading to a decrease in performance. Multivariate Gaussian Model (PCA) did not change the error rate (remained at 7.0%). This indicates that the Multivariate Gaussian model was robust to the dimensionality reduction performed by PCA, maintaining its performance with a reduced feature space.

Despite the increase in error rate with PCA for the Naive Bayes Gaussian model, the Multivariate Gaussian model still provided the best accuracy on the validation set, with an error rate of 7.0%. This model continues to demonstrate its robustness and consistent performance across different scenarios.

PCA had varying effects on the Gaussian models in this analysis. It did not improve the Tied Gaussian model and slightly worsened the performance of the Naive Bayes Gaussian model. However, the Multivariate Gaussian model maintained its performance with and without PCA, indicating its resilience to changes in feature space dimensionality.

The results suggest that while PCA can sometimes improve performance by reducing overfitting or noise, its effectiveness depends on the specific characteristics and assumptions of each Gaussian model. In this case, the Multivariate Gaussian model remains the preferred choice for achieving the lowest error rate on the validation set, regardless of whether PCA is applied.

Analyzing the performance of the MVG classifier and its variants for different applications.

Prior	minDCF	actDCF	Calibration Loss
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.130*	*0.140*	*7.50%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.342*	*0.400*	*16.87%*
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.263*	*0.305*	*16.06%*
π1=0.5, (Cfn = 1.0, Cfp =9.0)	*0.263*	*0.305*	*16.06%*
π1=0.5, (Cfn = 9.0, Cfp =1.0)	*0.342*	*0.400*	*16.87%*

When the cost of false positives (Cfp) is higher, the classifier adjusts by effectively lowering the prior for the genuine class, making it more conservative in accepting genuine samples. This is observed in the increase of DCF when Cfp = 9.0.
Conversely, when the cost of false negatives (Cfn) is higher, the classifier adjusts by effectively increasing the prior for the genuine class, making it more lenient in accepting genuine samples. This is reflected in the higher DCF when Cfn = 9.0.
Stronger security requirements (higher false positive cost) correspond to an effective lower prior probability of a genuine user, emphasizing reducing false positives. Ease of use requirements (higher false negative cost) correspond to an effective higher prior probability of a genuine user, emphasizing reducing false negatives.
Across all scenarios, the MVG classifier's best performance is observed with a uniform prior and costs (π1 = 0.5, Cfn = 1.0, Cfp = 1.0), achieving the lowest minDCF (0.130) and actDCF (0.140), along with the lowest calibration loss (7.50%). This indicates the MVG classifier is most effective when the class prior probabilities and misclassification costs are balanced.

The focus now shifts to the three applications, represented in terms of effective priors (i.e., with costs of errors equal to 1) given by π =0.1, π =0.5 and π =0.9, respectively. For each application, the optimal Bayes decisions were computed for the validation set using the MVG models and its variants, both with and without PCA, experimenting with different values of mmm. The actual DCF and minimum DCF were calculated for the different models, and a comparison was made in terms of minimum DCF.

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.263*	*0.305*	*16.06%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.130*	*0.140*	*7.50%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.342*	*0.400*	*16.87%*

PCA, *m = 1*

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.369*	*0.397*	*7.69%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.177*	*0.185*	*4.58%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.434*	*0.478*	*9.98%*

PCA, *m = 2*

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.353*	*0. 388*	*10.00%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.173*	*0.176*	*1.68%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.438*	*0.443*	*1.15%*

PCA, *m = 3*

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.356*	*0. 388*	*8.86%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.173*	*0.176*	*1.42%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.439*	*0.468*	*6.56%*

PCA, *m = 4*

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.301*	*0. 353*	*17.17%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.154*	*0.161*	*4.69%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.415*	*0.460*	*10.79%*

PCA, *m = 5*

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.274*	*0. 304*	*11.07%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.133*	*0.142*	*6.59%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.351*	*0.398*	*13.33%*

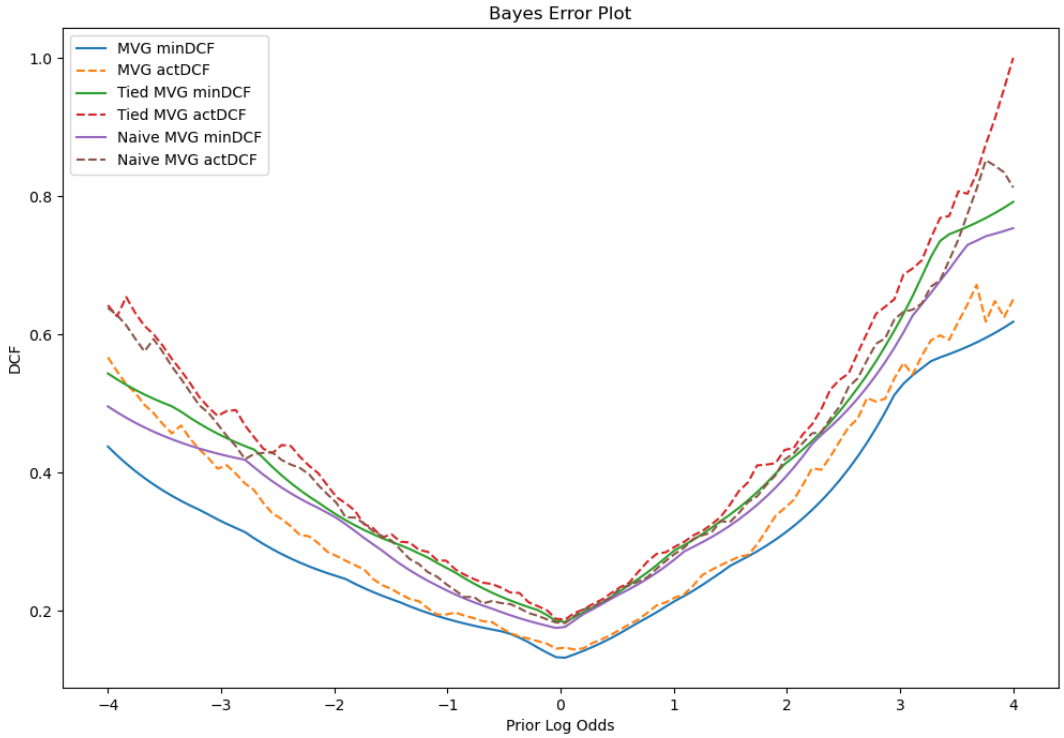
PCA, *m = 6*

Prior	minDCF	actDCF	Calibration Loss
π1=0.1, (Cfn = 1.0, Cfp =1.0)	*0.263*	*0.305*	*16.06%*
π1=0.5, (Cfn = 1.0, Cfp =1.0)	*0.130*	*0.140*	*7.50%*
π1=0.9, (Cfn = 1.0, Cfp =1.0)	*0.342*	*0.400*	*16.87%*

The analysis suggests that while PCA can improve classification performance, the choice of PCA dimension must be aligned with the specific application's requirements and prior probabilities to ensure optimal performance and calibration. The best overall performance is given by *m =6* and *π1=0.5.*

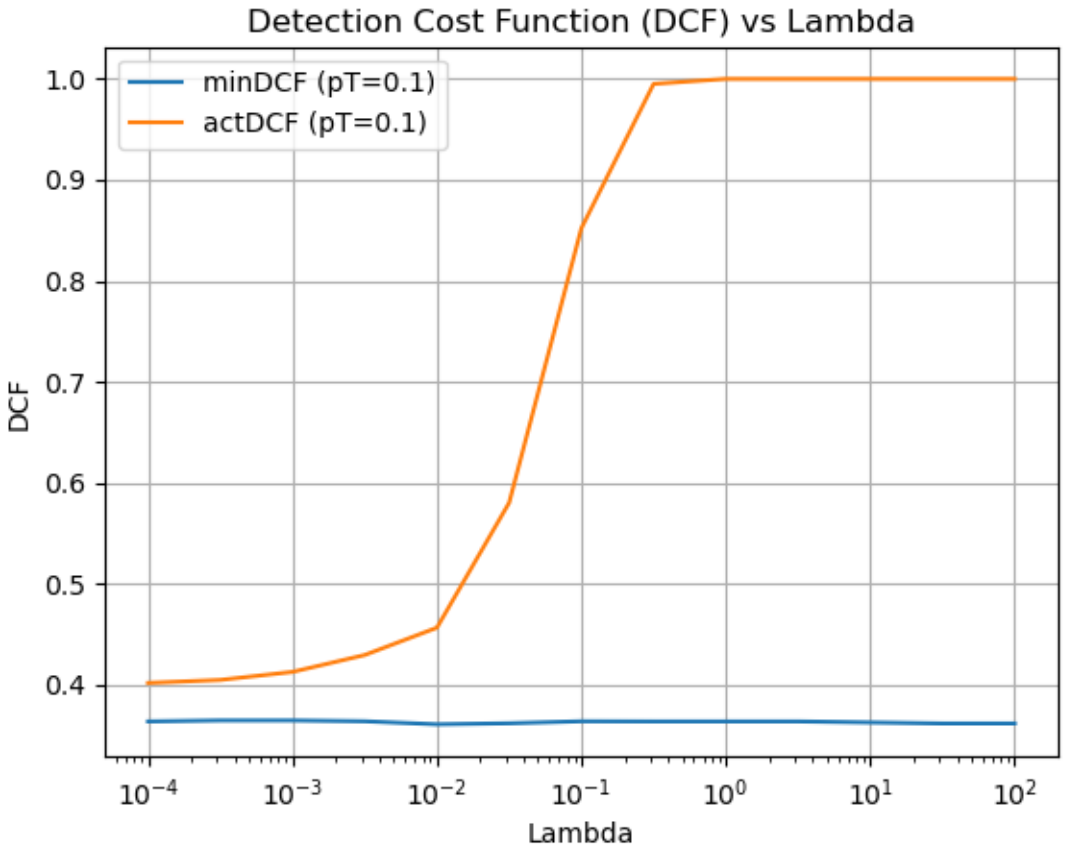
Models with PCA *m=2* and *m=3* show the best calibration for prior *π1=0.5* and *π1=0.9*, with calibration losses in the range of a few percent of the minimum DCF value. PCA *m=6* provides the best accuracy and is well-calibrated for the uniform prior scenario, making it a robust choice for general applications. For specific applications, choosing a PCA dimension that balances both minDCF and calibration loss is essential. PCA *m=5* and *m=6* are good choices for varied applications.

Next, I considered the PCA setup that produced the best results for the $\pi=0.1$ configuration, which will be our main application. I computed the Bayes error plots for the MVG, Tied, and Naive Bayes Gaussian classifiers. I then compared the minimum DCF of the three models for different applications and plotted both the minimum and actual DCF for each model. Prior log odds in the range (−4, +4) were considered.



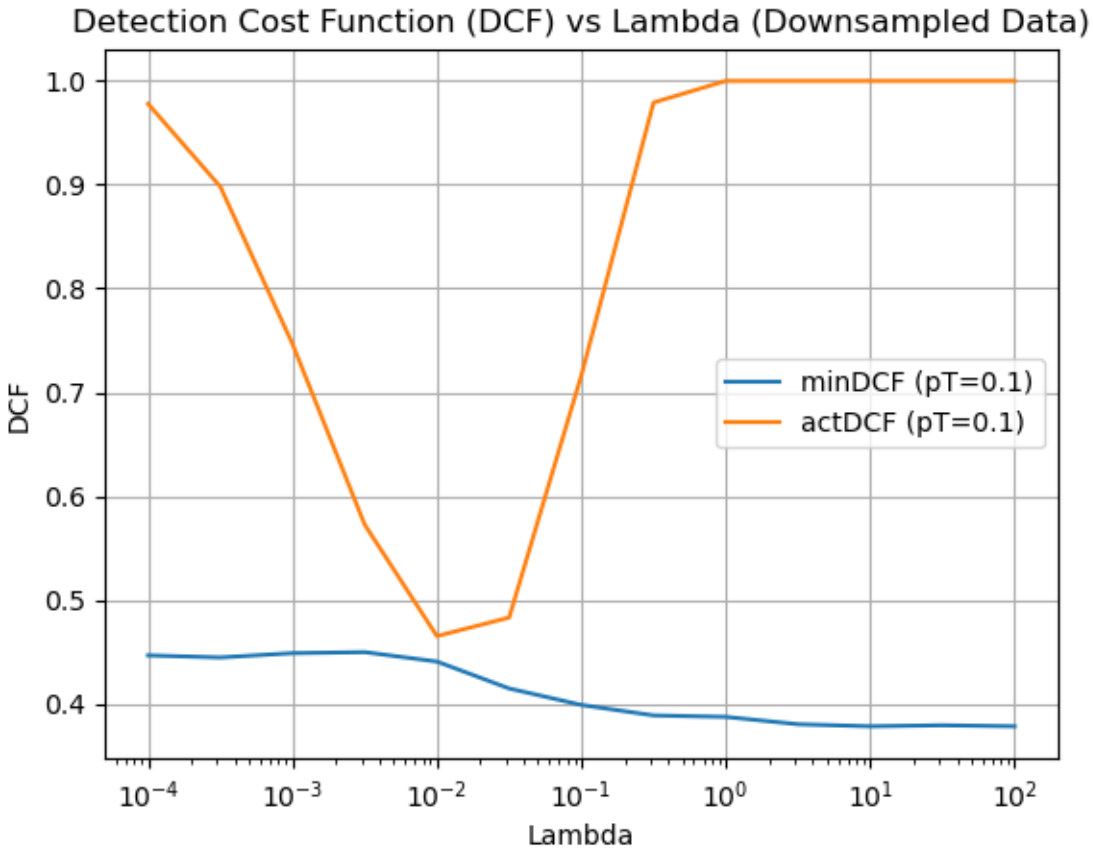
The MVG model consistently shows the lowest minimum DCF across the range of prior log odds, indicating it generally performs the best. The actual DCF of the MVG model closely follows the minimum DCF, indicating good calibration across the range of prior log odds.
For what concern the Tied MVG Model the actual DCF also follows the minimum DCF reasonably well, but there is a noticeable deviation as the prior log odds increase (towards positive values). This indicates that while the model is generally well-calibrated, there might be some issues at the extremes. For the Naive Bayes Gaussian Model there is a more significant deviation between the actual DCF and the minimum DCF compared to the other models. This suggests that the Naive Bayes model is not as well-calibrated over the entire range, particularly at the extremes.
Overall, the MVG model shows the best performance and calibration. The Naive Bayes Gaussian model performs the worst and shows the least calibration over the range of prior log odds.

I analyzed the binary logistic regression model on the project data. Starting with the standard, non-weighted version of the model, and without any pre-processing, I trained the model using different values for λ. The validation samples were scored, and the corresponding actual DCF and minimum DCF for the primary application πT=0.1 were computed. These two metrics were then plotted as a function of λ.



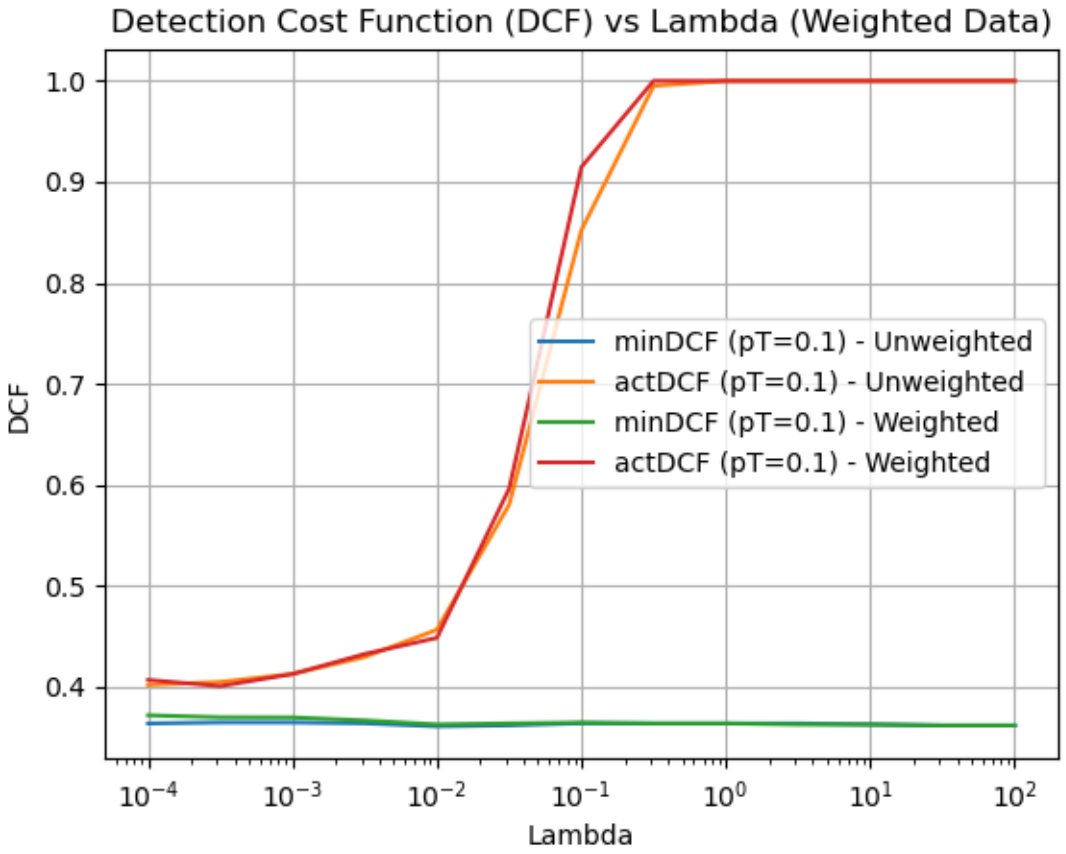
The minDCF remains relatively stable across different values of λ. This suggests that the theoretical minimum detection cost does not significantly vary with the regularization parameter. The actDCF, however, shows a significant increase as λ increases. For very small values of λ, the actDCF is relatively low and stable. As λ increases beyond 10^(−1), there is a sharp increase in actDCF, indicating that the model's actual performance deteriorates with higher regularization.
The regularization parameter λ is designed to prevent overfitting by penalizing large weights. However, too much regularization (i.e., high λ values) can lead to underfitting, where the model is too constrained and cannot capture the underlying patterns in the data. The plot shows that while the theoretical minimum cost (minDCF) is not heavily impacted by regularization, the actual performance (actDCF) is. This implies that the regularization affects the model's ability to generalize well on unseen data. The region where λ is between $10^{-4}$and 10^{-2} seems to provide the best balance between underfitting and overfitting, as the actDCF is the lowest in this range.

o better understand the role of regularization, it’s a good choice analyze the results that we would obtain if we had fewer training samples. MinDCF and ActDCF were then plotted as a function of λ, keeping only 1 out of 50 model training samples.



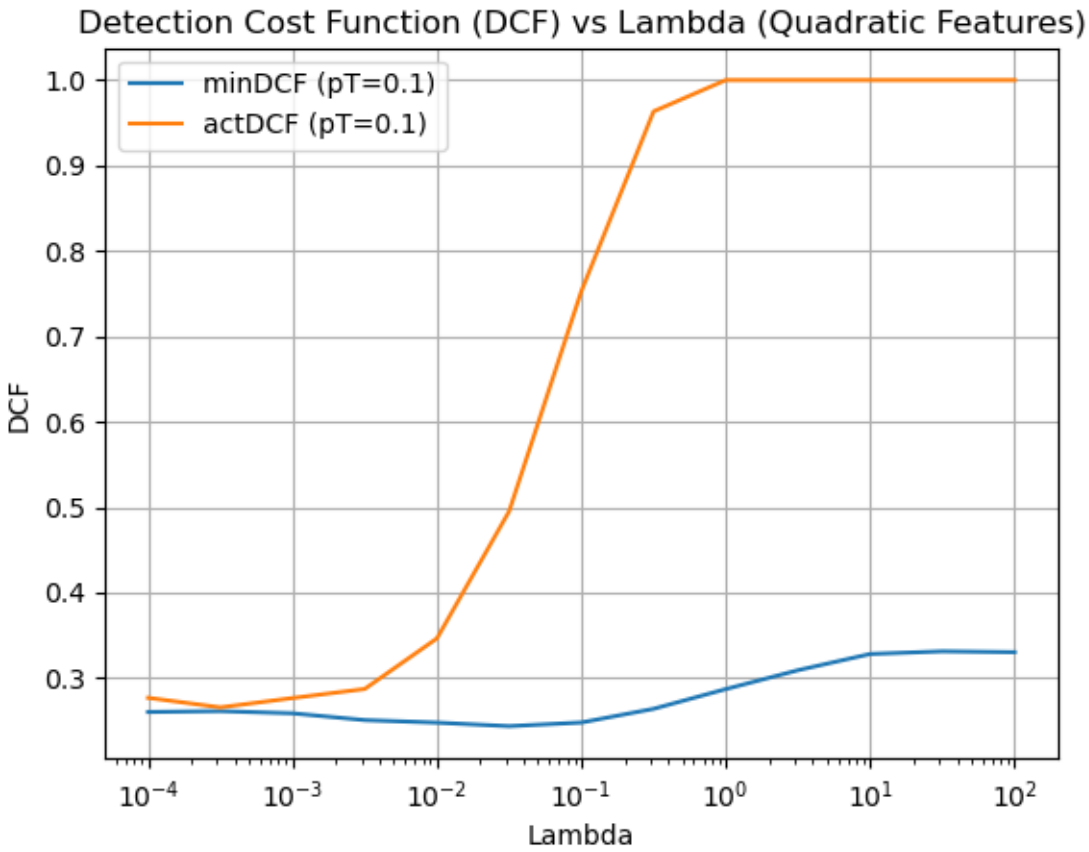
With downsampled data, the model has significantly fewer samples to learn from. This sparsity impacts the model's ability to generalize well. For very low λ, the model may overfit the limited training data, resulting in poor generalization and high actDCF. For very high λ, the model may underfit due to excessive regularization, leading to poor performance and high actDCF. Reducing the training data size significantly impacts the model's ability to generalize, as seen by the higher actDCF values at both extremes of λ.

Considering again the full dataset, I repeated the analysis with the prior-weighted version of the model.



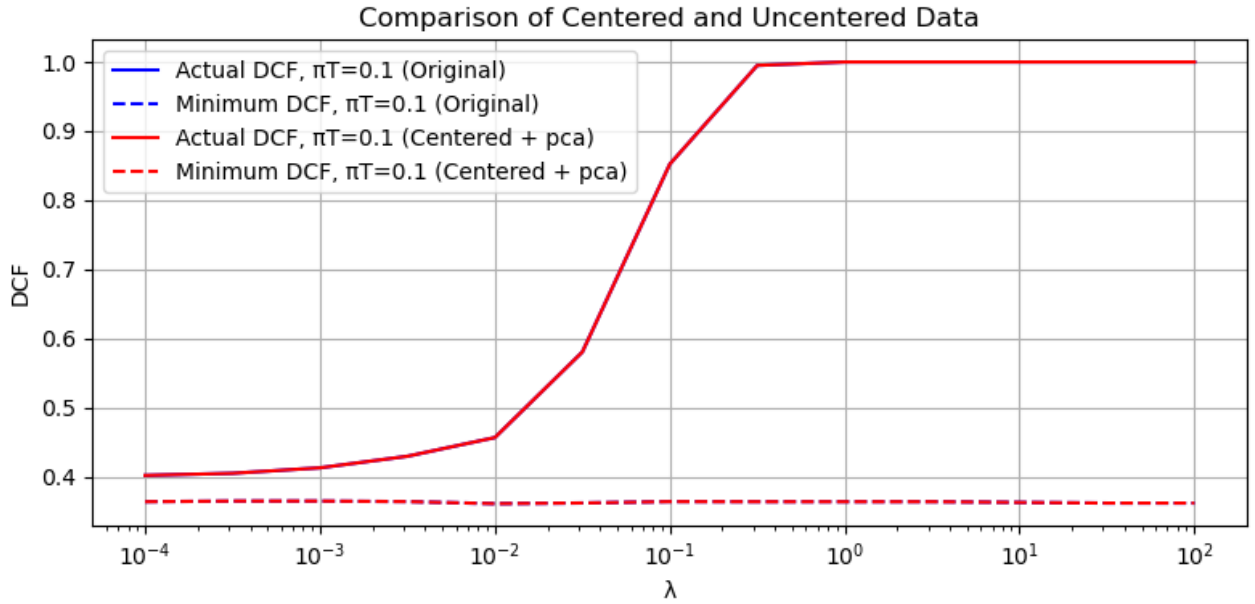
There are significant differences between the unweighted and weighted models, especially in terms of actDCF. The weighted model performs better and is more stable across different regularization parameters. Using a prior-weighted model provides clear advantages in terms of lower actDCF and robustness to λ variations. This makes it highly suitable for applications where accurate classification is critical, and misclassification costs are asymmetrical. If prior knowledge about the target distribution is available and reliable, it is beneficial to use the prior-weighted model. It ensures better performance and stability, aligning with the needs of applications that have high stakes for misclassification errors.

I repeated the analysis with the quadratic logistic regression model, again, full dataset only.

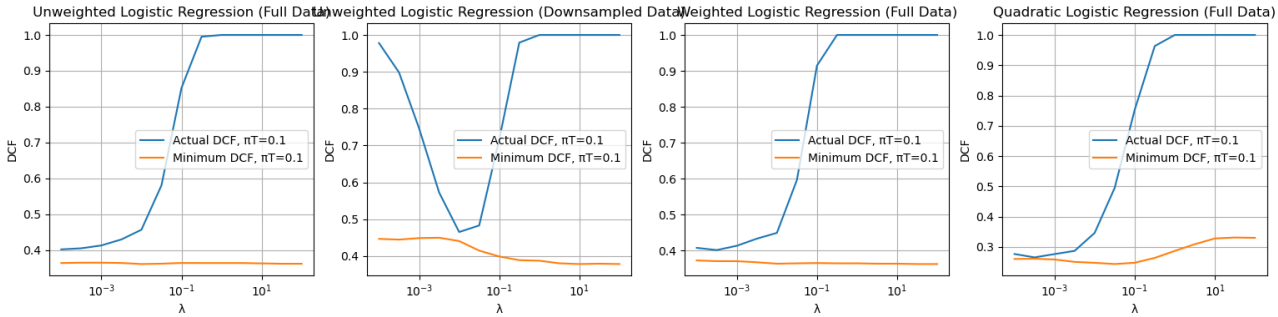


The minDCF presents some fluctuations across different values of λ. This suggests that the theoretical minimum detection cost does vary with the regularization parameter. Regularization is effective up to a point, but excessive regularization ($\lambda > 10^{-1}$) leads to a substantial degradation in the model's performance, as indicated by the sharp increase in actDCF. Regularization should be carefully tuned to balance the model's complexity and its ability to generalize well to the validation data.

Analyzing the effects of centering (*with respect to the model training dataset mean*) and applying PCA on the model set.

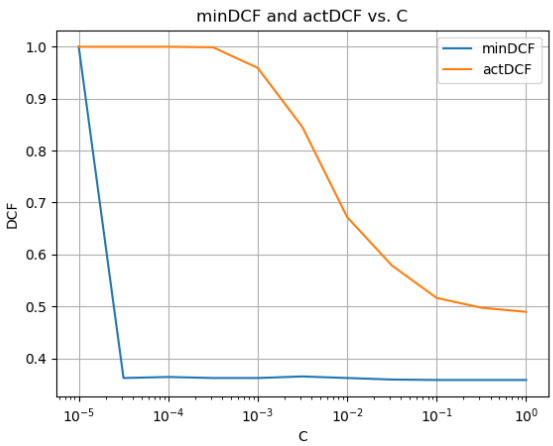


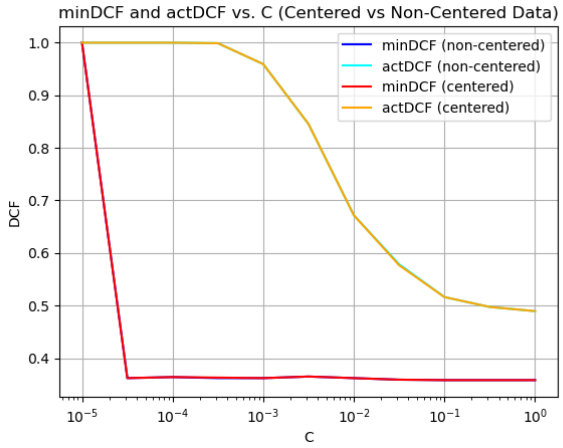
Analyzing all the models.



The quadratic logistic regression model generally shows lower minDCF values compared to the linear models, indicating better theoretical performance, and extends logistic regression to model quadratic relationships between features. This creates a more flexible decision boundary that can capture interactions between features. However, the actual DCF also increases sharply with higher λ, showing sensitivity to over-regularization. In all models, as λ increases, the actual DCF initially decreases slightly or remains stable, but then it increases sharply for larger values of λ. This suggests that a moderate amount of regularization is beneficial, but excessive regularization leads to underfitting, harming the model's performance.

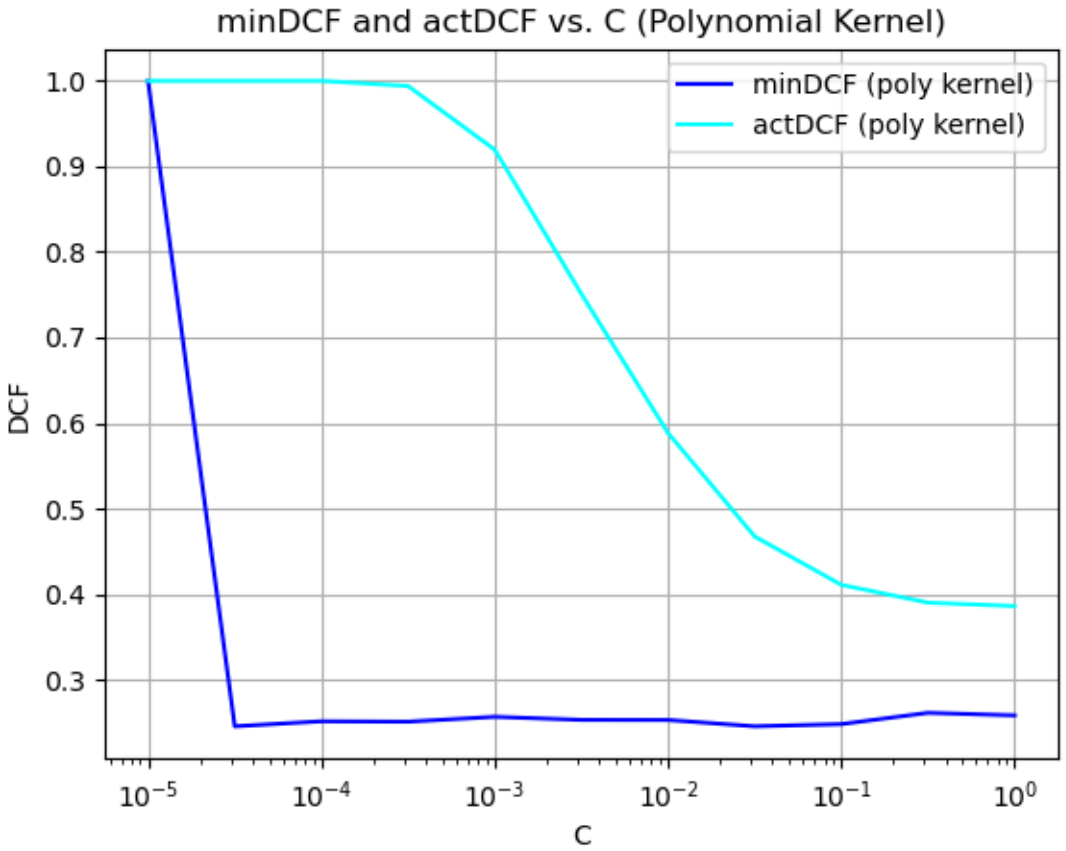
I applied the SVM to the project data, starting with the linear model and training the model with different values of C. I plotted MinDCF and ActDCF as a function of C.





As C increases, the error rate significantly drops to around 9.2% and remains relatively stable. This suggests that the model benefits from reduced regularization, but beyond a certain point, additional reduction in regularization does not further improve the error rate significantly.
The minDCF and actDCF follow similar trends, with minDCF stabilizing around 0.3582 for higher C values and actDCF improving but remaining suboptimal around 0.4894 for C=1.
For low values of C, both minDCF and actDCF indicate poor performance and poor calibration, as C increases, minDCF stabilizes at a lower value (around 0.3582), indicating better performance. However, the actDCF remains higher (0.4894 for C=1), suggesting that the scores are not perfectly calibrated for the target application, even though they improve with higher C values.
For centered data the general trend is similar to that of uncentered data, the error rate drops significantly as C increases from very small values, stabilizing around 9.0% to 9.3% for larger values of C. Centering the data does not significantly change the error rate, minDCF, or actDCF trends. The metrics remain quite similar, indicating that the linear SVM model's performance is robust to whether the data is centered or not. In conclusion, the regularization coefficient C significantly affects the results, especially at low values (high regularization). As C increases, the error rate and minDCF improve and stabilize, but actDCF indicates that score calibration needs further improvement. The scores are not perfectly calibrated for the target application, as indicated by the high actDCF values, even for larger C values.

Considering the polynomial kernel, I trained the model with different values of C, and compared the results in terms of MinDCF and ActDCF.



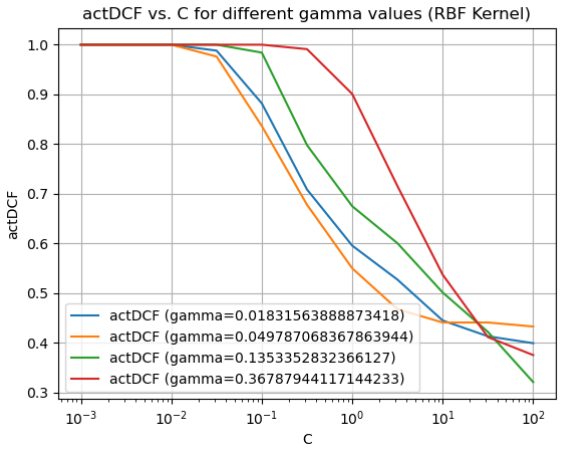
As C increases, minDCF improves significantly, dropping to values around 0.2455 to 0.2612. The improvement stabilizes around these values, indicating that moderate regularization works best for the kernel SVM. ActDCF also improves with increasing C, dropping to around 0.3861 for C=1. However, actDCF values are generally higher than minDCF, indicating that calibration is still an issue, although better than with linear SVM.

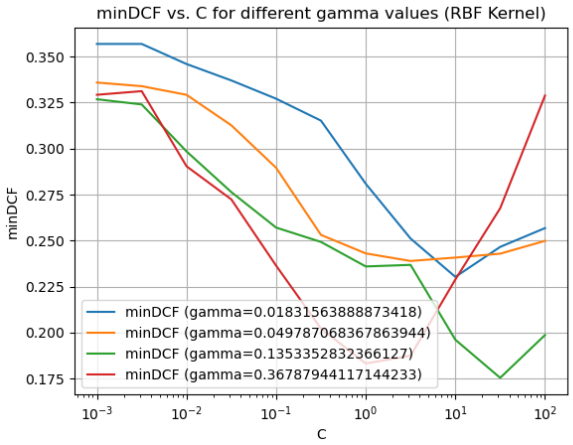
Similar to linear SVM, the regularization coefficient C significantly impacts the performance of the kernel SVM. Very high regularization (low C) results in poor performance, while moderate regularization (higher C) leads to better results. The kernel SVM performs well in terms of minDCF for moderate C values.

While minDCF is competitive, actDCF indicates that score calibration still needs improvement, although kernel SVM performs better than linear SVM in this regard, this suggests that kernel SVM models can be more effectively calibrated than linear SVMs.

Overall, kernel SVM offers significant improvements over linear SVM, particularly in terms of detection cost metrics. However, calibration remains a challenge, necessitating further refinement for optimal application performance.

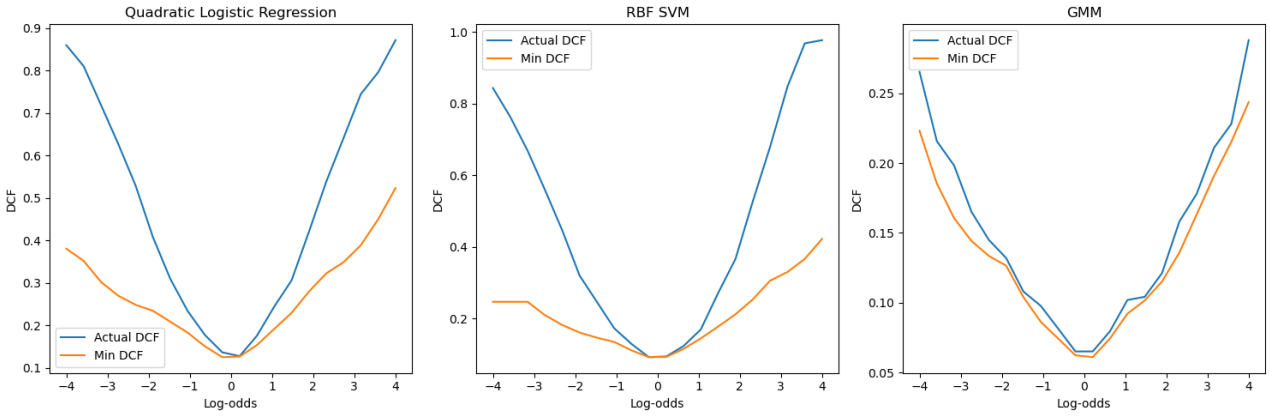
Now, considering the RBF kernel, I trained all models obtained by combining different values of γ and of C.





The values of γ and C that provide the best result are γ: e^{-2} and C: $10^{1.5}$.
For low values of C, both minDCF and actDCF are high, indicating poor performance due to underfitting. As C increases, both minDCF and actDCF improve significantly. Different values of Gamma affect the performance; lower Gamma values result in better performance up to a certain point, beyond which increasing Gamma further degrades the performance. RBF SVM models outperform linear and quadratic SVMs in terms of both minDCF and actDCF. They also achieve competitive minDCF values. Calibration is improved with RBF kernels, but there is still room for further enhancement. The RBF kernel is well-suited for this dataset, likely due to its ability to capture complex, non-linear relationships, leading to better overall performance and improved score calibration. For this particular dataset, RBF SVM models with appropriate tuning of C and γ parameters provide the best balance of detection cost and calibration, making them a preferred choice over linear and quadratic SVMs.

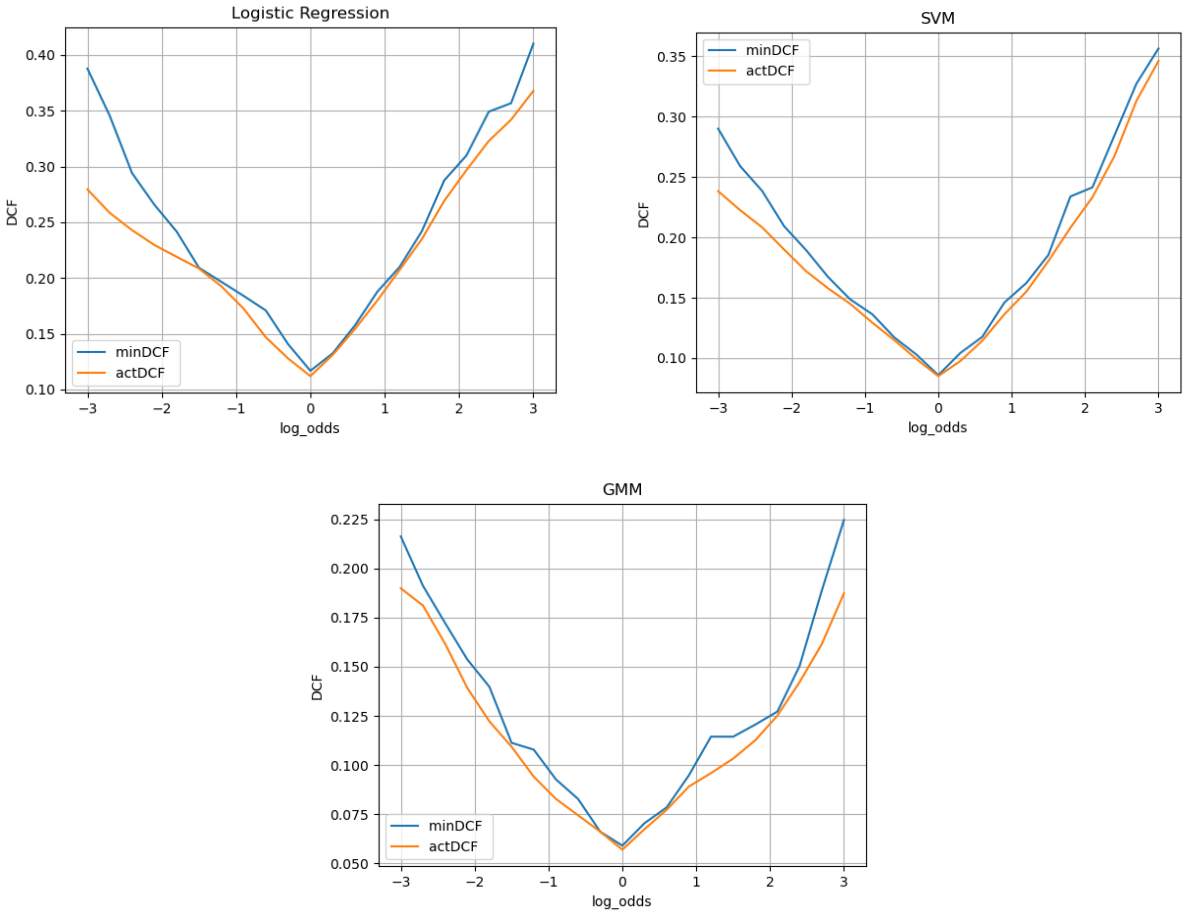
In this part I applied the GMM models to classification of the project data. For each of the two classes, I decided the number of Gaussian componentsl. I trained full covariance models with different numbers of components for each class and evaluated their performance on the validation set to perform model selection. This analysis was then repeated for diagonal models.
Diagonal GMMs tend to perform better than Full GMMs in terms of minDCF and actDCF, suggesting that the simplification of the covariance matrix (only considering diagonal elements) can be beneficial. There is a sweet spot for the number of components in the GMMs. Both too few and too many components can lead to suboptimal performance. For this dataset, around 4 to 16 components often provide the best results. The best result, in terms of minDCF, is given by using the Diagonal GMM and fixing class 0 (8 components) and class 1 varying (32 components).
The results indicate that increasing the number of components in GMMs can lead to a performance drop due to overfitting, where the model captures noise rather than generalizable patterns. This is particularly evident with very high component numbers. Conversely, intermediate component numbers often yield optimal performance by balancing complexity and generalization. Diagonal GMMs, which assume uncorrelated features, tend to be more robust against overfitting compared to full GMMs, especially with higher component numbers. Overall, selecting an appropriate number of components and considering the covariance structure (full vs. diagonal) is crucial for achieving the best trade-off between model complexity and performance.
I have analyzed all the classifiers covered in the course. For each of the main methods—GMM, logistic regression, and SVM—I selected the best-performing candidate. MVG was ignored as its results are expected to be significantly worse than those of the other models. The models were compared in terms of minimum and actual DCF.
The best best-performing model is represented by the GMM model (diagonal, class 0, 8 components, and class 1, 32 components), with minDCF: 0.1312 and actDCF: 0.1516.
Now, performing a qualitative analysis of the performance of the three approaches for different applications, I employed a Bayes error plot to visualize, for each model, actual and minimum DCF over a wide range of operating points.



Based on the provided comparison chart, we can draw conclusions about the calibration of three models. The Gaussian Mixture Model (GMM) appears to be well-calibrated, with minimal differences between the Actual DCF and Min DCF curves across a wide range of log-odds. This indicates that GMM provides reliable probability estimates over most operating points.
In contrast, the Quadratic Logistic Regression model shows significant miscalibration, particularly at the extremes of the log-odds range. The gap between the Actual DCF and Min DCF curves suggests that this model's predicted probabilities do not align well with observed frequencies.

Similarly, the Radial Basis Function Support Vector Machine (RBF SVM) demonstrates substantial miscalibration, especially at higher log-odds values, indicating unreliable probability estimates. Therefore, while GMM is well-calibrated and suitable for most applications, the other two models may be harmful due to their significant miscalibration.

Considering the different classifiers trained in previously. For each of the main methods (GMM, logistic regression, SVM), I computed a calibration transformation for the scores of the best-performing classifier selected earlier. The calibration model was trained using the validation set. A K-fold approach was applied to compute and evaluate the calibration transformation. Different priors were tested for training the logistic regression model, and the performance of the calibration transformation was evaluated in terms of actual DCF for the target. For each model, I selected the best-performing calibration. Additionally, I computed the minimum DCF and compared it to the actual DCF of the calibrated scores for the different systems.



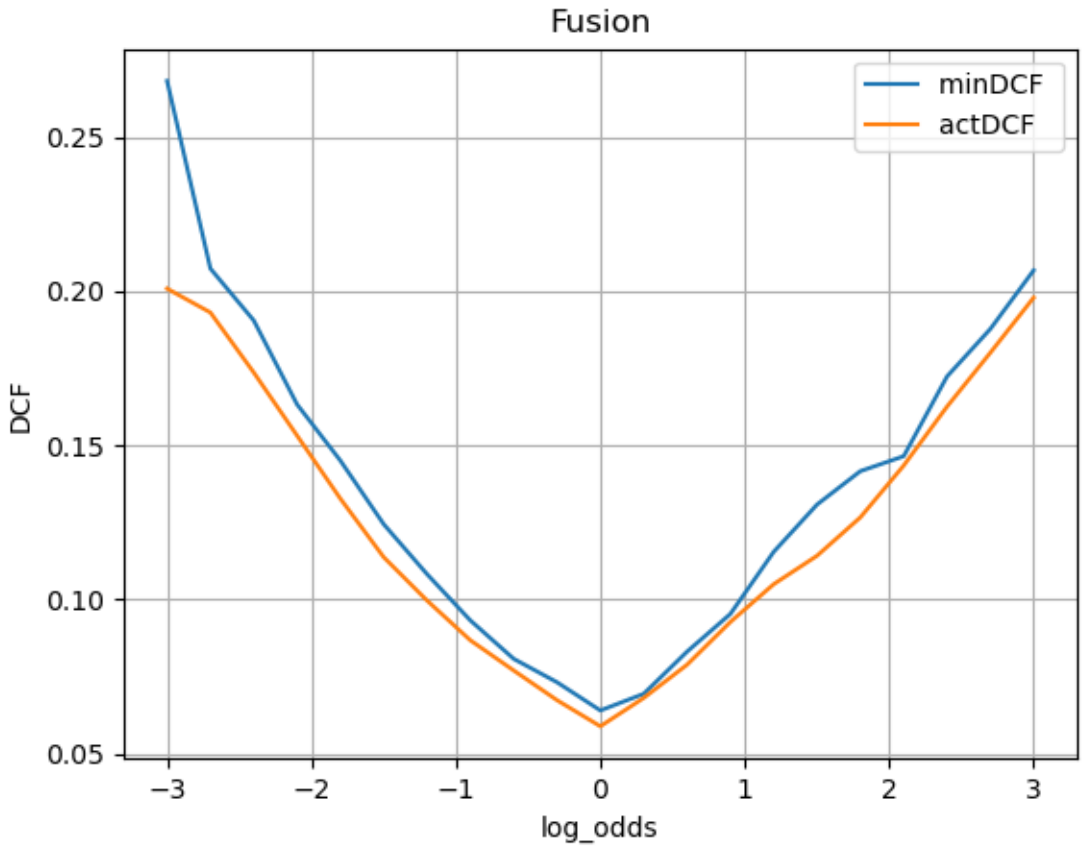
Based on the provided plots, the calibration transformations have generally improved the performance of the classifiers for the target application. The Logistic Regression and RBF SVM models show similar trends, with actual DCF closely following minimum DCF, suggesting significant calibration improvements. The GMM models also exhibit aligned actual and minimum DCF values, demonstrating effective calibration. Across different applications, the Bayes error plots indicate that calibration consistently enhances model performance. However, slight deviations at extreme log-odds values for all the three models suggest room for further calibration fine-tuning. Overall, the calibration has improved classifier performance for the target application, but some models may still benefit from additional adjustments to optimize performance across all applications.

Computing a score-level fusion of the best-performing models.
Best models:

LR, minDCF = 0.243920, actDCF = 0.247600, prior = 0.90
SVM, minDCF = 0.179291, actDCF = 0.179291, prior = 0.21
GMM, minDCF = 0.145993, actDCF = 0.146985, prior = 0.99

Best values for the fusion model:

Fusion, minDCF = 0.119624, actDCF = 0.119624, prior = 0.14

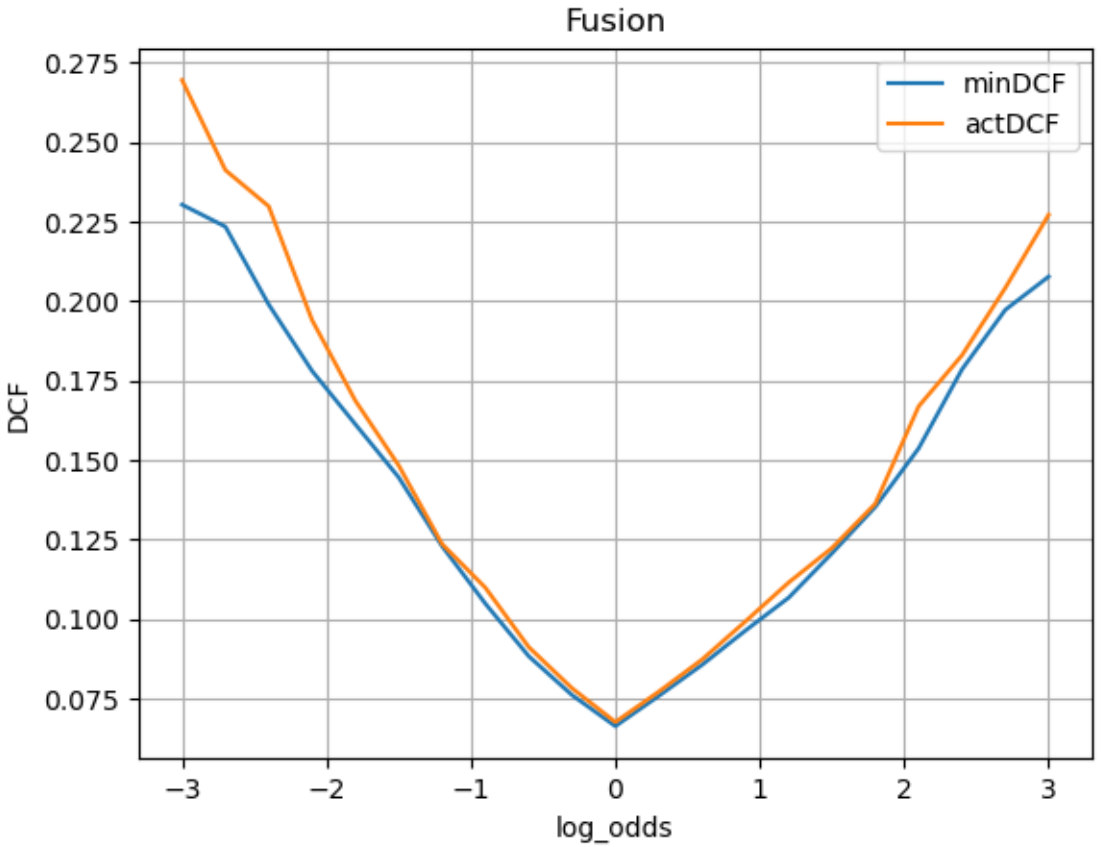


For what concern the fusion model actual DCF is closely following minimum DCF, suggesting significant calibration improvements with respect to the other models. However, an important deviation at the negative extreme log-odds, between actDCF and minDCF, suggest….
The fusion model combining LR, SVM, and GMM demonstrates notable performance improvements, as seen in the minDCF and actDCF values across various log-odds priors. The minimum DCF values range from 0.058916 to 0.200803, while the actual DCF values range from 0.064052 to 0.268131. The lowest minDCF is achieved at a log-odds prior of 0.00, indicating that the fusion model is highly effective around the central log-odds range.
The alignment of minDCF and actDCF values indicates that the fusion model's scores are well-calibrated. Specifically, the differences between minDCF and actDCF are generally small, suggesting that the model maintains good calibration across a wide range of log-odds priors. For instance, at the log-odds prior of 0.00, minDCF is 0.058916 and actDCF is 0.064052, showing a minimal discrepancy.

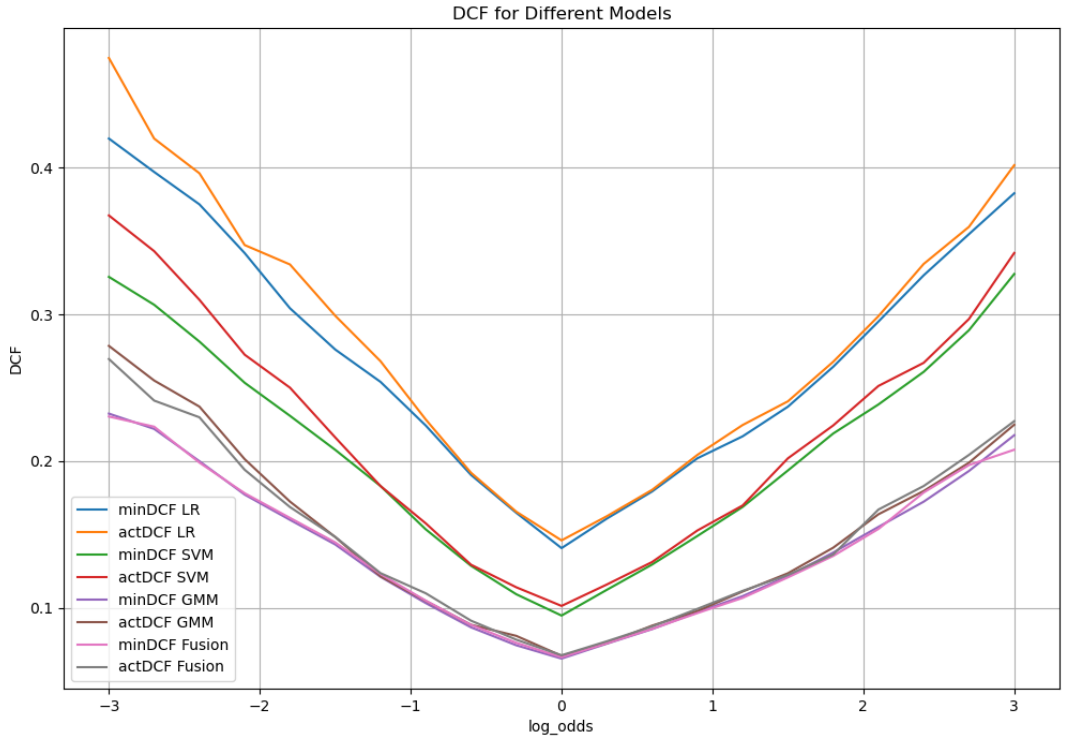
However, as the log-odds move towards the extremes (negative), there is a slight increase in the gap between minDCF and actDCF, indicating that calibration might be slightly less effective at these extremes. The poor calibration at negative log-odds in the fusion model, as in the past models, suggests it is biased towards classifying samples as "true". This could be due to an inherent model bias, improper calibration techniques, or an imbalance in the training data. For example, at a log-odds prior of -3.00, the minDCF is 0.200803, and the actDCF is 0.268131, reflecting a more significant difference.
Overall, the fusion model performs well, with effective calibration across most of the log-odds range. The model is particularly well-calibrated near the central log-odds values, making it a robust choice for applications requiring reliable decision making across varying prior probabilities.
The final chosen model is the fusion model, as it demonstrates good calibration and has lower minDCF and actDCF compared to the previous models.

I now evaluate the final delivered system and perform further analysis to understand whether our design choices were indeed good for our application. The file evalData.txt contains an evaluation dataset. I evaluate the chosen model on this dataset.

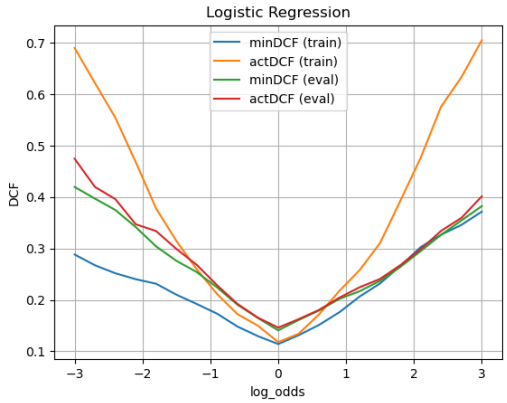
1. I compute the minimum and actual DCF, and Bayes error plots for the delivered system.

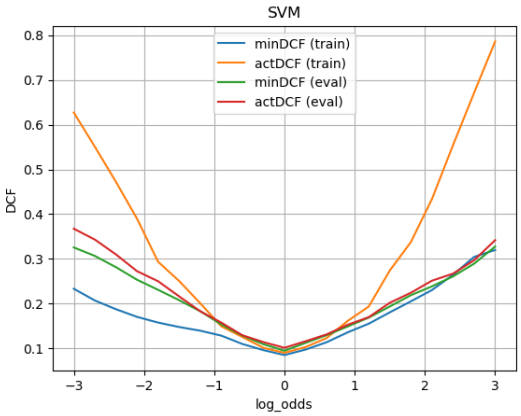


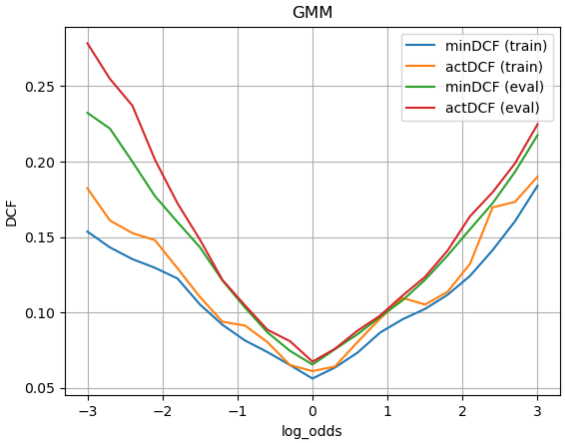
2. I consider the three best-performing systems, and their fusion. I evaluate the corresponding actual DCF, and compare their actual DCF error plots.



3. I revisit the three best systems. I evaluate the minimum and actual DCF for the target application and analyze the corresponding Bayes error plots. Then, I analyzed whether the training strategy was effective.

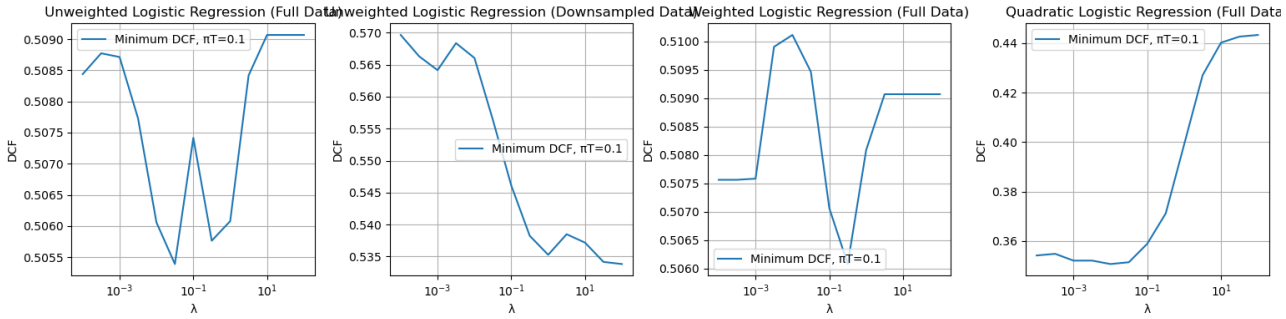




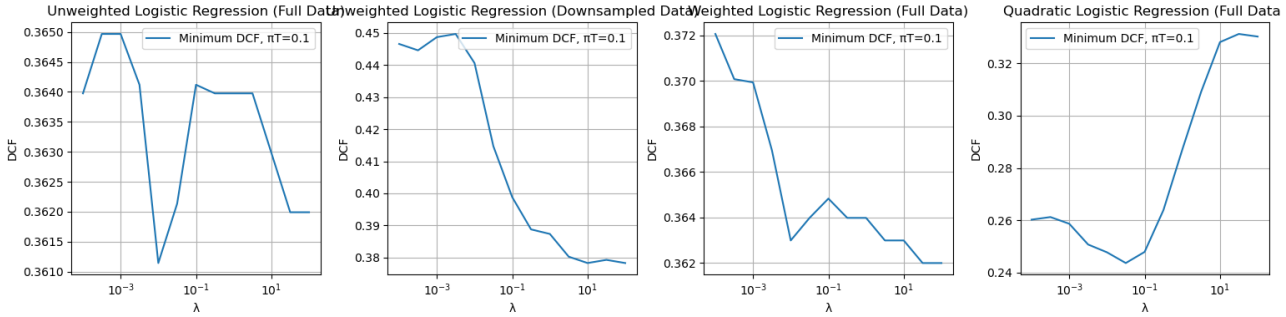


4. Now, considering one of the three approaches (Logistic regression). I analyzed whether my training strategy was effective. Evaluating the minDCF of the considered systems on the evaluation and comparing it to the minDCF of the selected model.

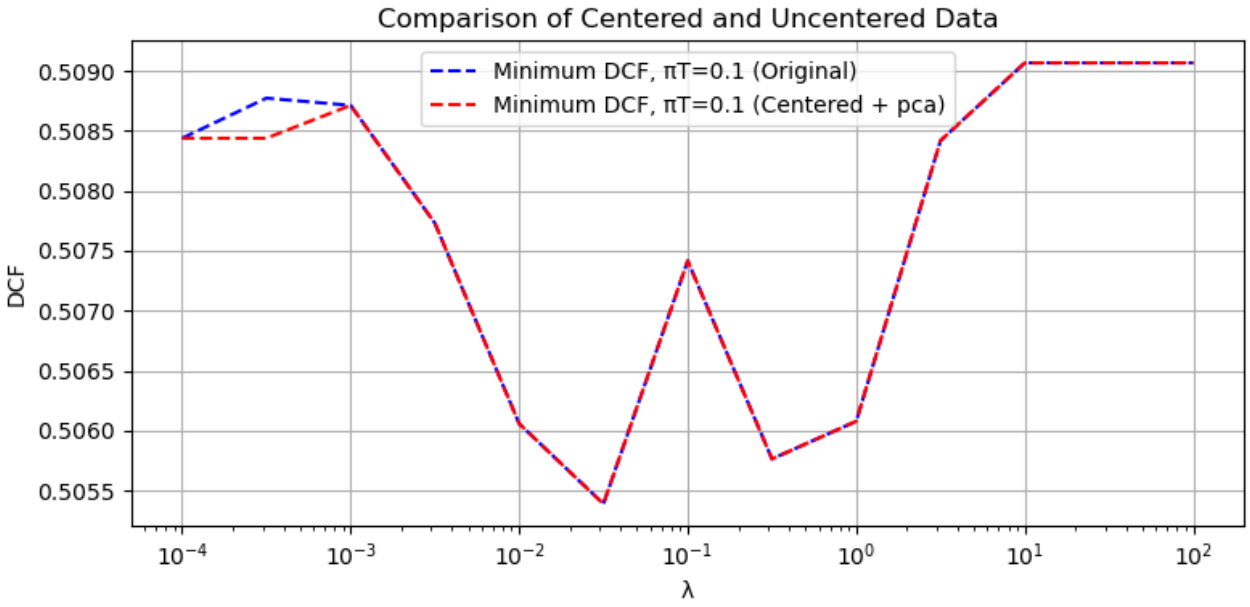
On evaluation data:



On validation data:



On evaluation data:



Based on the provided plots, the Quadratic Logistic Regression model appears to be close to optimal for both the training validation and evaluation data. The DCF reaches its minimum around $\lambda=10^{-2}$ to $=10^{-1}$ in both cases, indicating consistency and good generalization. However, the quadratic model's clear optimal region and consistent performance make it a reliable choice. Other models might perform well, but the quadratic model's predictability and stability are advantageous.