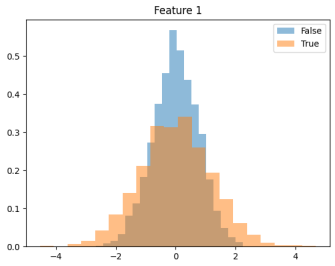
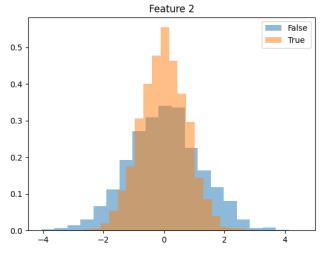
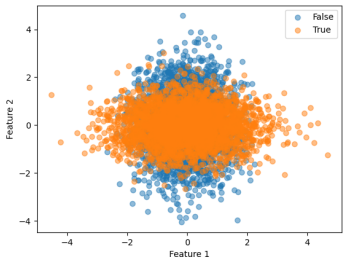
Report - Alessandro De Marco

Lab 2

Features 1 - 2







What do you observe?

- Both classes (False and True) show a bell-shaped distribution with significant overlap near 0 in both features. The scatter plot highlights substantial overlap, making it difficult to distinguish between classes using only these features.

Do the classes overlap? If so, where?

- **Yes**, both features show notable overlap, especially around 0. The overlap is stronger for feature 1, while feature 2 shows slightly more separation.

Do the classes show similar means for the first two features?

- **Feature 1**: Class 0 mean is **0.0029**, and Class 1 mean is **0.0005**, indicating almost identical means.
- **Feature 2**: Class 0 mean is **0.0187**, and Class 1 mean is **0.0085**, showing slight but minimal difference.

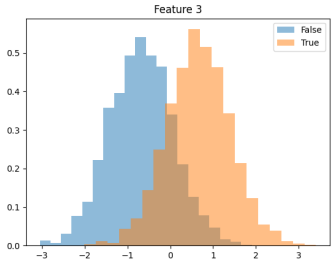
Are the variances similar for the two classes?

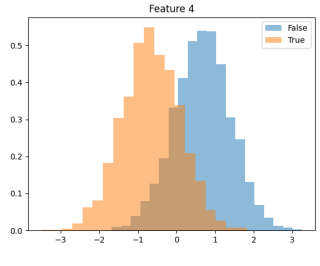
- **Feature 1**: Class 1 has a larger variance (**1.4302**) compared to Class 0 (**0.5696**), meaning it is more dispersed.
- **Feature 2**: Class 0 has a larger variance (**1.4209**) than Class 1 (**0.5783**), making it more spread out.

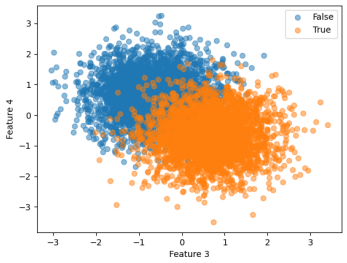
How many modes are evident from the histograms?

- Both features have **a single peak** for each class, indicating unimodal distributions.

Features 3 - 4







What do you observe?

- The histograms for both classes (False and True) show a more distinct separation compared to features 1 and 2. Class 0 (False) is skewed toward negative values, while Class 1 (True) is skewed toward positive values for both feature 3 and feature 4.
- The scatter plot indicates some overlap between the two classes, but the separation between the two is more evident along both feature 3 and feature 4 compared to the earlier features.

Do the classes overlap? If so, where?

- **Yes**, the classes overlap, but the overlap is less pronounced than in features 1 and 2. For feature 3, the overlap occurs around 0, while for feature 4, the overlap is also near 0, but the separation between the two classes is more noticeable.

Do the classes show similar means for the third and fourth features?

- **Feature 3**: Class 0 mean is **0.6809**, and Class 1 mean is **0.6652**, indicating a significant difference in the means.
- **Feature 4**: Class 0 mean is **0.6708**, and Class 1 mean is **0.6642**, also showing a large difference in the means.

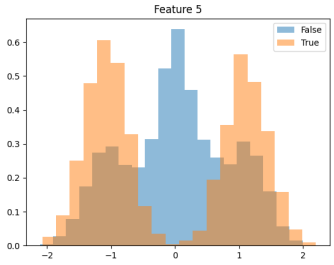
Are the variances similar for the two classes?

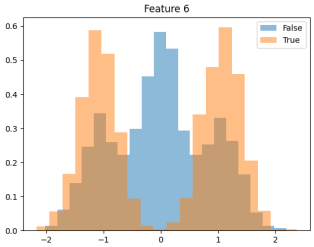
- **Feature 3**:The variances are almost identical: Class 0 variance is **0.5500**, and Class 1 variance is **0.5489**, showing similar spread for both classes.
- **Feature 4**: The variances are also very close: Class 0 variance is **0.5360**, and Class 1 variance is **0.5533**, again indicating similar spread.

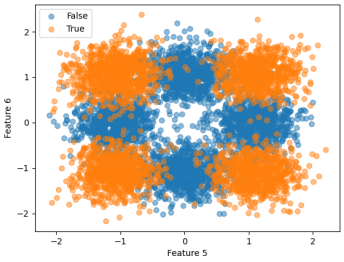
How many modes are evident from the histograms?

- Both histograms for feature 3 and feature 4 show a **single peak** for each class, but the peaks are clearly shifted, indicating distinct unimodal distributions for both classes.

Features 5 - 6







What do you observe?

- The histograms for both classes show a multi-modal distribution, with notable overlap around 0 for both features.
- The scatter plot reveals a grid-like pattern, indicating some overlap but also distinct clustering for both classes.

Do the classes overlap? If so, where?

- **Yes**, the classes overlap significantly near 0 for both features, but distinct clusters are visible in other areas of the scatter plot.

Do the classes show similar means for the fifth and sixth features?

- **Feature 5**: Class 0 mean: **0.0280**, Class 1 mean: **0.0417** (very close means).
- **Feature 6**: Class 0 mean: **0.0058**, Class 1 mean: **0.0239** (again, very similar means).

Are the variances similar for the two classes?

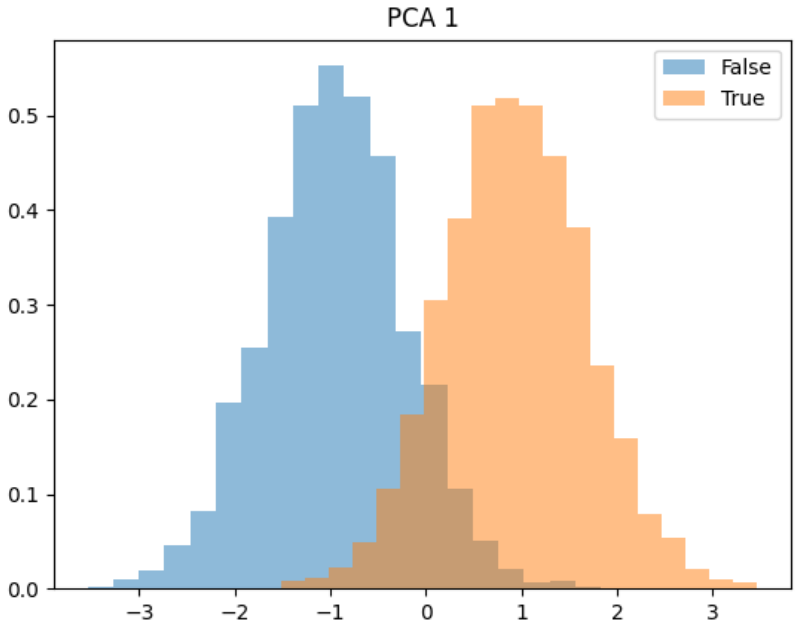
- **Feature 5**: Class 0 variance: **0.6801**, Class 1 variance: **1.3178** (Class 1 has a larger spread).
- **Feature 6**: Class 0 variance: **0.7050**, Class 1 variance: **1.2870** (Class 1 is more dispersed).

How many modes are evident from the histograms?

- Both features show **multiple peaks**, indicating a **multi-modal** distribution for each class.

Lab 3

PCA 1



What do you observe?

- The histogram of the first PCA direction shows two well-separated distributions for the False and True classes. Class 0 (False) is primarily concentrated on the left (negative values), while Class 1 (True) is concentrated on the right (positive values), with some overlap around 0.

What are the effects on the class distributions?

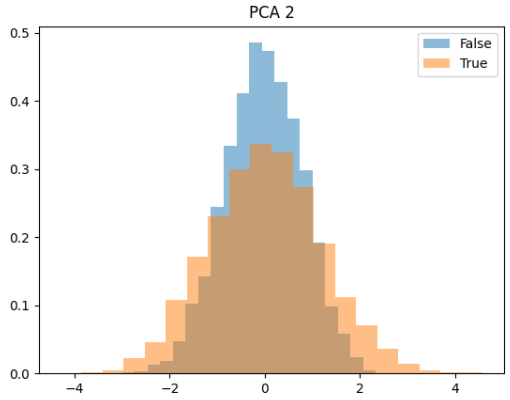
- **Class separation**: PCA has been effective in creating a separation between the two classes, compressing the variance in the data so that class 0 and class 1 are clearly distinguishable in this projection.
- **Overlap**: While there is still some overlap near 0, the projection from PCA has reduced the overall class overlap compared to individual features, meaning that the data is now more separable.

Can you spot the different clusters inside each class?

- The histogram does not show distinct clusters within each class; both classes seem to follow a **single-mode distribution**. However, there is clear separation between the two classes along this PCA direction, suggesting that PCA has effectively captured the dominant variance that differentiates the classes.

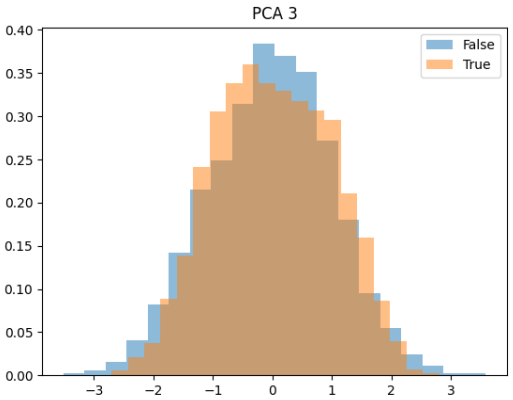
PCA 2:

- **Observation**: The histograms for both classes are almost fully overlapping, with both classes centered around 0.
- **Effect on distributions**: PCA 2 does not provide a clear separation between the classes; both classes follow a similar distribution.
- **Clusters**: No clusters are evident, with the classes having very similar distributions.



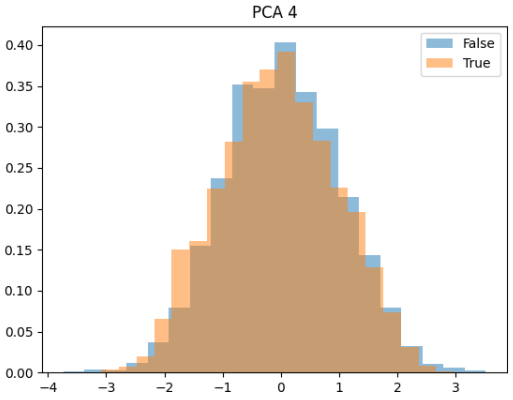
PCA 3:

- **Observation**: Similar to PCA 2, both classes are overlapping, with a symmetric distribution centered around 0.
- **Effect on distributions**: There is no significant separation between the classes along PCA 3, and both follow similar bell-shaped distributions.
- **Clusters**: No clusters can be spotted inside each class, as the distributions are nearly identical.



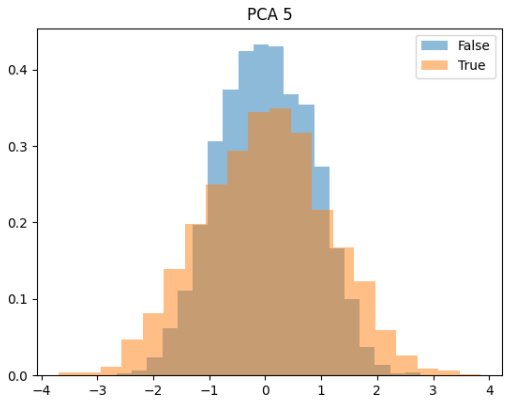
PCA 4:

- **Observation**: The histograms show almost perfect overlap between the two classes, with no visible separation.
- **Effect on distributions**: There is little to no separation along PCA 4. The projection results in similar distributions for both classes.
- **Clusters**: No distinct clusters can be identified.



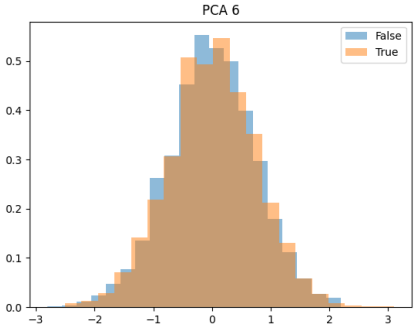
PCA 5:

- **Observation**: The distributions of the two classes show some overlap, but there is a small separation, particularly in the tails of the distributions.
- **Effect on distributions**: PCA 5 provides slightly more separation than PCA 2-4, but the overlap is still substantial, especially near the center.
- **Clusters**: No clear clusters can be seen.

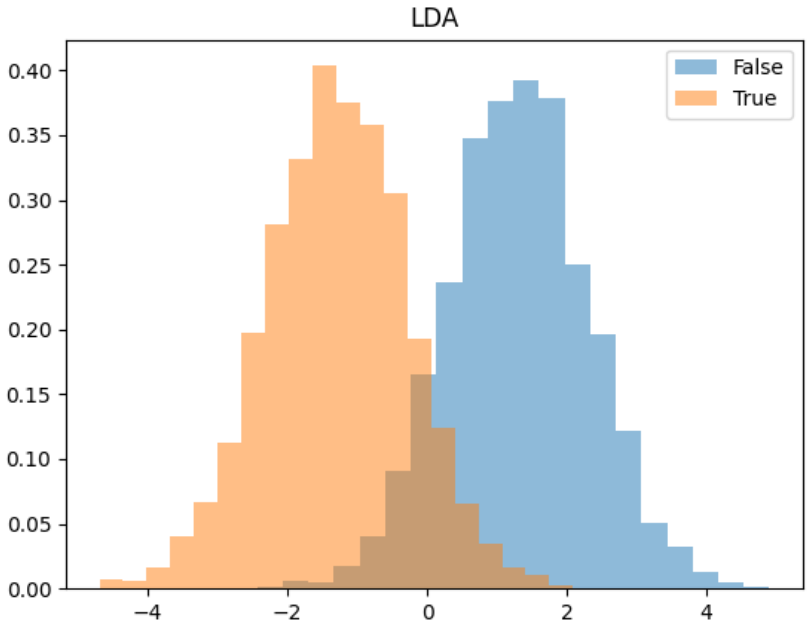


PCA 6:

- **Observation**: The histograms for both classes are largely overlapping, with little visible separation.
- **Effect on distributions**: Similar to PCA 2-4, PCA 6 does not differentiate between the classes well.
- **Clusters**: No clusters are evident.



LDA



What do you observe?

- The LDA histogram shows a **clear separation** between the two classes (False and True). Class 0 (False) is concentrated on positive values, while Class 1 (True) is mostly on the negative side, with only a small overlap around 0.

Do the classes overlap?

- **Yes**, there is a small overlap between the classes near 0, but it is much **less** pronounced compared to the PCA projections or the original feature histograms from Laboratory 2.

Compared to the histograms of the 6 features you computed in Laboratory 2, is LDA finding a good direction with little class overlap?

- **Yes**, LDA is finding a **much better direction** for separating the two classes compared to the original 6 features in Laboratory 2. In the original feature histograms, there was significant overlap between the two classes, particularly around the center of the distributions. However, with LDA, the overlap is minimized, and the separation between the classes is much clearer.

LDA as classifier

When applying LDA as a classifier, you split the dataset into training and validation sets. The LDA projection was oriented such that the mean of class True (label 1) was larger than that of class False (label 0), and the threshold was selected as the average of the projected class means.

- **LDA accuracy**: 0.907
- **Error rate**: 0.093

This result indicates that LDA is able to classify the data with relatively high accuracy, achieving a low error rate when the threshold is set as the average of the projected means.

Changing the threshold:

You experimented with different threshold values to see how they affect classification performance. The results were:

- **Threshold 0.5**: Accuracy = 0.8835, Error = 0.1165
- **Threshold 0.6**: Accuracy = 0.8725, Error = 0.1275
- **Threshold 0.7**: Accuracy = 0.855, Error = 0.145
- **Threshold 0.8**: Accuracy = 0.8405, Error = 0.1595
- **Threshold 0.9**: Accuracy = 0.8235, Error = 0.1765

What do you observe?

- As the threshold increases, the **accuracy decreases** steadily. The higher the threshold, the more **misclassifications** occur, likely because you're moving away from the optimal point that balances the two class distributions.
- The default threshold (the average of the projected means) seems to provide the **best accuracy** (0.907), while increasing the threshold beyond that point consistently leads to worse performance.

Can you find values that improve the classification accuracy?

- Based on the results, the **default threshold** (the average of the projected class means) gives the highest accuracy. None of the other tested thresholds improved accuracy. In fact, adjusting the threshold above the default pro

---

Apply PCA before LDA classifier

- **PCA with 1 dimension**:
  - Accuracy: **0.9065**
  - Error: **0.0935**
- **PCA with 2 dimensions**:
  - Accuracy: **0.9105**
  - Error: **0.0895**
- **PCA with 3 dimensions**:
  - Accuracy: **0.9075**
  - Error: **0.0925**
- **PCA with 4 dimensions**:
  - Accuracy: **0.9075**
  - Error: **0.0925**
- **PCA with 5 dimensions**:
  - Accuracy: **0.907**
  - Error: **0.093**
- **PCA with 6 dimensions**:
  - Accuracy: **0.907**
  - Error: **0.093**

What do you observe?

- The accuracy remains consistently high across different values of PCA dimensions, with slight variations. The accuracy ranges from **0.9065** to **0.9105**, with corresponding error rates between **0.0895** and **0.0935**.
- The **best accuracy** is achieved using the first two PCA dimensions, with a maximum accuracy of **0.9105** when using two PCA dimensions, and the error rate is the lowest at **0.0895**.

Can you find values of m that improve the accuracy on the validation set?

- **Yes**, using **2 PCA dimensions** yields the highest accuracy (**0.9105**) and the lowest error rate (**0.0895**), slightly improving the classification performance compared to using more or fewer dimensions.
- Beyond 2 PCA dimensions, the accuracy remains stable, but no further improvements are observed.

Is PCA beneficial for the task when combined with the LDA classifier?

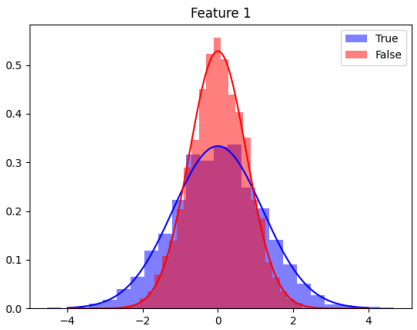
- **Yes**, PCA appears to be beneficial when combined with LDA, as it maintains a high classification accuracy with even fewer dimensions. The accuracy using PCA with 2 dimensions is **slightly higher** than the accuracy of the original LDA without PCA, which was **0.907**.
- **Dimensionality reduction** through PCA simplifies the feature space without sacrificing accuracy, which could be advantageous in reducing computational cost while maintaining performance.

Lab 4

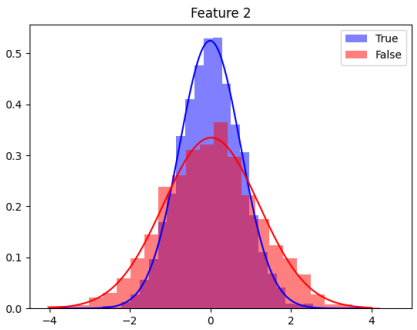
Features 1 - 2

What do you observe?

- **Feature 1**:
  - The Gaussian model seems to fit reasonably well for both classes. However, there are some **differences in the tails** of the distribution, where the True class (blue) shows a slightly heavier tail compared to the Gaussian curve. The fit is generally good in the central region.



- **Feature 2**:
  - For feature 2, the fit appears **quite accurate**, especially in the center of the distribution for both classes. The Gaussian densities closely follow the shape of the histograms, with minor deviations in the tails. The fit is better for the True class (blue) compared to the False class (red), which has a slightly flatter peak.



Are there features for which the Gaussian densities provide a good fit?

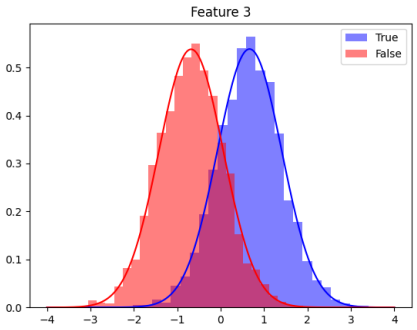
- **Yes**, feature 2 provides a **very good fit** for both classes. The Gaussian densities follow the shape of the histograms closely, with only minor deviations in the tails.

Are there features for which the Gaussian model seems significantly less accurate?

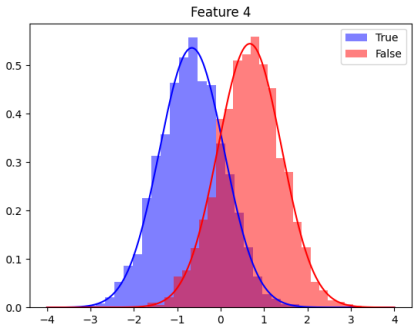
- **Feature 1** shows a slightly less accurate fit, particularly in the tails of the distribution, where the True class exhibits a heavier tail. However, the fit is still reasonable, especially in the central part of the distribution.

Features 3 - 4

- **Feature 3**:
  - The Gaussian model fits **very well** for both classes. The distributions are clearly separated, with Class 0 (red) skewed toward negative values and Class 1 (blue) toward positive values. The Gaussian densities follow the histograms closely, especially in the center, with slight deviations in the tails.



- **Feature 4**:
  - Similar to Feature 3, the fit is **quite accurate**. The two classes are well separated, and the Gaussian densities match the histograms closely. There is minimal deviation in the tails for both classes, and the overall fit is strong.



Are there features for which the Gaussian densities provide a good fit?

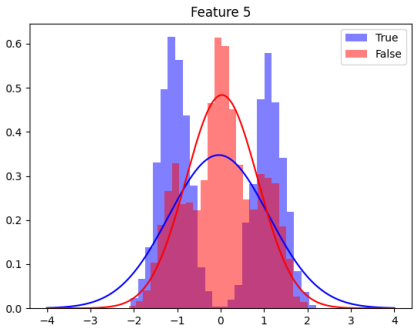
- **Yes**, both Feature 3 and Feature 4 provide a **good fit** for the Gaussian model. The Gaussian densities closely match the histograms for both classes, particularly in the central region.

Are there features for which the Gaussian model seems significantly less accurate?

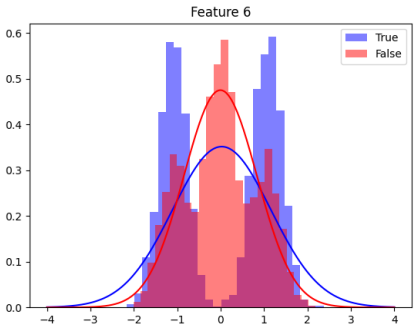
- For these features, there are **no significant inaccuracies** in the Gaussian fit. Both features are well-modeled by the Gaussian distributions.

Features 5 - 6

- **Feature 5**:
  - La **Gaussian model** does not fit very well. The histograms for both classes show multiple peaks, indicating a **multi-modal** distribution, while the Gaussian curve assumes a **single-mode** distribution. The Gaussian fit captures the central peak reasonably well but fails to represent the structure in the tails and secondary peaks.



- **Feature 6**:
    - Similar to Feature 5, the **Gaussian model** struggles to capture the multi-modal nature of the distribution. There are multiple peaks in the histogram for both classes, while the Gaussian curve assumes a single-mode distribution. The fit is reasonable near the central peak but does not accurately represent the distribution tails or the secondary peaks.



Are there features for which the Gaussian densities provide a good fit?

- **No**, both Feature 5 and Feature 6 show a **poor fit** to the Gaussian model due to their multi-modal distributions. The Gaussian assumption of a single peak does not capture the multiple peaks present in the data.

Are there features for which the Gaussian model seems significantly less accurate?

- **Yes**, Feature 5 and Feature 6 both show significant deviation from the Gaussian model. The **multi-modal structure** of these features is not well-represented by the Gaussian densities, making the fit less accurate for both classes.

Lab 5

MVG Model (Multivariate Gaussian):

- **Error rate**: 0.070
- **Observation**: The MVG model has the lowest error rate among the models tested, indicating that it performs the best on this dataset when compared to the other models. This suggests that the full covariance structure of the data is important for making accurate predictions.

LDA (Linear Discriminant Analysis):

- **Error rate**: 0.093
- **Observation**: LDA has a higher error rate compared to the MVG model. This is expected, as LDA assumes that the classes share a common covariance matrix and that the decision boundary is linear, which may not capture the full complexity of the data as well as the MVG model does.

Tied Gaussian Model:

- **Error rate**: 0.093
- **Observation**: The tied Gaussian model has an error rate almost identical to LDA. This is because the tied model, like LDA, assumes that all classes share a common covariance matrix, which simplifies the model but may result in a less accurate representation of the data. This explains why the tied Gaussian performs similarly to LDA.

Naive Bayes Gaussian Model:

- **Error rate**: 0.072
- **Observation**: The Naive Bayes Gaussian model performs slightly worse than the MVG model but still better than LDA and the tied Gaussian model. The assumption of conditional independence between features simplifies the model, but it still performs relatively well. The small difference in error between Naive Bayes and MVG suggests that feature dependencies might not be too strong in this dataset, making Naive Bayes a competitive option.

Covariance matrix for MVG - class 0

	1	2	3	4	5	6
1	0.600957	0.000052	0.019059	0.019253	0.012804	-0.013472
2	0.000052	1.447225	-0.016134	-0.015856	-0.026453	0.022914
3	0.019059	-0.016134	0.565349	-0.001843	-0.006914	0.016893
4	0.019253	-0.015856	-0.001843	0.541615	0.005252	0.013572
5	0.012804	-0.026453	-0.006914	0.005252	0.696068	0.015844
6	-0.013472	0.022914	0.016893	0.013572	0.015844	0.686520

Class 0 Covariance Matrix:

- The **variances** (diagonal elements) for each feature are:

- Feature 1: 0.600957
- Feature 2: 1.447225
- Feature 3: 0.565349
- Feature 4: 0.541615
- Feature 5: 0.696068
- Feature 6: 0.686520

- The **covariances** (off-diagonal elements) are generally small in magnitude compared to the variances. For instance:
  - Covariance between Feature 1 and Feature 2: 0.000052 (very small relative to their variances)
  - Covariance between Feature 3 and Feature 4: -0.001843 (also very small)
- **Observation**: Most covariances are small compared to the variances, suggesting a **low correlation** between pairs of features for class 0.

Covariance matrix for MVG - class 1

	1	2	3	4	5	6
1	1.448095	-0.014722	0.005570	0.015742	0.019497	-0.000177
2	-0.014722	0.553391	-0.011217	-0.009065	-0.014659	0.016349
3	0.005570	-0.011217	0.557480	0.027561	-0.003770	-0.014598
4	0.015742	-0.009065	0.027561	0.569657	-0.011698	0.034993
5	0.019497	-0.014659	-0.003770	-0.011698	1.342018	0.016945
6	-0.000177	0.016349	-0.014598	0.034993	0.016945	1.303719

Class 1 Covariance Matrix:

- The **variances** (diagonal elements) for each feature are:

- Feature 1: 1.448095
- Feature 2: 0.553391
- Feature 3: 0.557480
- Feature 4: 0.569657
- Feature 5: 1.342018
- Feature 6: 1.303719

- As in class 0, the **covariances** are quite small compared to the variances. For instance:
  - Covariance between Feature 1 and Feature 2: -0.014722 (small relative to the variances)
  - Covariance between Feature 4 and Feature 6: 0.034993 (relatively small)
- **Observation**: Similar to class 0, the covariances are much smaller than the variances, indicating **low correlation** between features for class 1.

Are covariance values large or small compared to variances?

- The covariances for both classes are **very small** compared to the variances. This indicates that the feature pairs are **weakly correlated** in both classes, with most feature pairs showing low interaction or dependence.

Pearson Correlation Coefficient for Class 0:

	0	1	2	3	4	5
0	1.000000	0.000055	0.032698	0.033747	0.019797	-0.020974
1	0.000055	1.000000	-0.017837	-0.017910	-0.026356	0.022988
2	0.032698	-0.017837	1.000000	-0.003331	-0.011022	0.027116
3	0.033747	-0.017910	-0.003331	1.000000	0.008553	0.022257
4	0.019797	-0.026356	-0.011022	0.008553	1.000000	0.022920
5	-0.020974	0.022988	0.027116	0.022257	0.022920	1.000000

- **Observation**:
  - Most of the off-diagonal elements are close to **0**, indicating **weak or no correlation** between most pairs of features.
  - The highest correlation is between:
    - Feature 4 and Feature 0: **0.033747**, which is still a **weak positive correlation**.
    - Feature 3 and Feature 0: **0.032698**, which also indicates a **weak positive correlation**.
  - The rest of the correlation values are all relatively small, suggesting that for class 0, the features are generally **uncorrelated** or only **weakly correlated**.

Pearson Correlation Coefficient for Class 1:

	0	1	2	3	4	5
0	1.000000	-0.016446	0.006199	0.017332	0.013986	-0.000129
1	-0.016446	1.000000	-0.020195	-0.016145	-0.017010	0.019248
2	0.006199	-0.020195	1.000000	0.048907	-0.004358	-0.017123
3	0.017332	-0.016145	0.048907	1.000000	-0.013379	0.040605

4	0.013986	-0.017010	-0.004358	-0.013379	1.000000	0.012811
5	-0.000129	0.019248	-0.017123	0.040605	0.012811	1.000000

- **Observation**:
  - Similar to class 0, most off-diagonal elements are near **0**, indicating **weak correlations** between most feature pairs.
  - The highest correlation is between:
    - Feature 4 and Feature 3: **0.048907**, which is still a **weak correlation**, though slightly stronger than the correlations observed in class 0.
  - The rest of the correlations are all quite small, with values ranging between -0.02 and 0.02, confirming that the features in class 1 are also largely **uncorrelated**.

What can you conclude about the features?

- The features for both classes (0 and 1) show **weak or no correlation**, as the Pearson correlation coefficients for most feature pairs are very close to 0. This suggests that, for this dataset, the features are largely **independent** from one another.

How is this related to the Naive Bayes results?

- The **Naive Bayes model** assumes that the features are conditionally independent given the class label. The fact that the correlation matrices show weak correlations between the features supports this assumption. This explains why the **Naive Bayes Gaussian model** performed reasonably well, with an error rate of **0.072**, close to the MVG model's performance. The low correlation between features suggests that the independence assumption made by Naive Bayes is not heavily violated, making it a suitable model for this dataset.

What can you conclude on the goodness of the Gaussian assumption?

- **The Gaussian assumption fits some features better than others.** For certain features, such as **Feature 1**, **Feature 2**, **Feature 3**, and **Feature 4**, the Gaussian model provides a reasonably good fit. The histograms for these features show a single peak, and the Gaussian density fits well, particularly in the central region of the distribution. This indicates that the assumption of Gaussianity holds reasonably well for these features.

Is it accurate for all the 6 features?

- **No, the Gaussian assumption is not equally accurate for all six features.** While the assumption seems reasonable for Features 1-4, it is less accurate for Features 5 and 6, where the Gaussian model struggles to capture the true distribution of the data.

Are there features for which the assumptions do not look good?

- **Yes, Features 5 and 6 do not follow the Gaussian assumption well.** In these cases, the histograms clearly show **multi-modal** distributions, with multiple peaks. The Gaussian model, which assumes a single-mode distribution, does not capture this structure accurately. As a result, the Gaussian density underestimates the complexity of the data for these features, leading to a poor fit.

Impact of Discarding Poorly Modeled Features on Classification Performance

MVG (Multivariate Gaussian):

- **Error with all features**: 0.070
- **Error with only features 1-4**: 0.0795
    - When excluding the last two features (5 and 6), the error rate **increases** from **0.070 to 0.0795**. This suggests that despite the poor Gaussian fit for features 5 and 6, the MVG model was able to extract some useful information from them, helping improve classification accuracy.

Naive Bayes:

- **Error with all features**: 0.072
- **Error with only features 1-4**: 0.0765
    - Similar to the MVG model, excluding the last two features **increases the error rate** slightly, from **0.072 to 0.0765**. Even though features 5 and 6 do not follow the Gaussian assumption well, they still provide useful information for the Naive Bayes model, contributing to a better classification when included.

Tied Gaussian:

- **Error with all features**: 0.093
- **Error with only features 1-4**: 0.095
    - For the tied Gaussian model, the error rate remains **almost the same** when excluding the last two features (from **0.093 to 0.095**). This suggests that the tied model did not gain significant benefit from features 5 and 6, possibly due to its simplified covariance structure.

LDA (Linear Discriminant Analysis):

- **Error with all features**: 0.093
- **Error with only features 1-4**: 0.095
    - Similarly, the error rate for LDA shows **little change**, moving from **0.093 to 0.095** when discarding features 5 and 6. This indicates that LDA may not have been able to leverage the information in these features effectively.

What can we conclude on discarding the last two features?

1. **MVG and Naive Bayes models**:
    - Both the MVG and Naive Bayes models show **worse performance** when features 5 and 6 are discarded, despite the poor Gaussian fit for those features. This suggests that **even poorly modeled features** (features 5 and 6) can still carry some **useful information** for classification, helping improve accuracy when included.
2. **Tied Gaussian and LDA**:
    - For both the tied Gaussian and LDA models, **discarding features 5 and 6** does not significantly affect performance. This implies that these models were likely **not extracting much useful information** from these features in the first place, possibly due to the limitations of their shared covariance structure or linear decision boundary assumptions.

Impact of Variance and Mean Differences on MVG and Tied MVG Performance with Feature Pairs

Classification with Features 1-2:

- **MVG**: 0.365
- **Naive Bayes**: 0.363
- **Tied MVG**: 0.4945
- **LDA**: 0.495

Classification with Features 3-4:

- **MVG**: 0.0945
- **Naive Bayes**: 0.0945
- **Tied MVG**: 0.0940
- **LDA**: 0.095

In the first case (features 1-2), which model is better?

- **MVG and Naive Bayes** perform similarly with error rates of **0.365** and **0.363**, respectively. Both outperform the **Tied MVG** and **LDA** models, which have much higher error rates around **0.494-0.495**.
- **Tied MVG** and **LDA** perform poorly with this feature pair, almost as if they were making random predictions.

In the second case (features 3-4), which model is better?

- All models perform very similarly for features 3-4, with error rates ranging from **0.093 to 0.095**. There is no significant difference between the performance of **MVG**, **Naive Bayes**, **Tied MVG**, and **LDA** when using these two features.

How is this related to the characteristics of the two classifiers?

Features 1-2:

- For **features 1 and 2**, the **variances differ** significantly between the two classes (as seen in Laboratory 2 and 4), while the **means** are similar. Additionally, the features show **low correlation** between each other.
- **Tied MVG and LDA** assume the same covariance matrix for both classes. Since the variances of features 1 and 2 are different, these models struggle to differentiate between the classes, resulting in high error rates (around **0.495**).
- **MVG** and **Naive Bayes**, on the other hand, allow for different covariance matrices (MVG) or assume conditional independence (Naive Bayes), which makes them better suited to handle the variance differences between the two classes. Hence, they perform better, even though the overall performance is still not great.

Features 3-4:

- For **features 3 and 4**, the classes differ mainly in terms of their **means**, while the **variances** are similar, and there is **limited correlation** between the features.
- Since the variance is similar between the two classes, **Tied MVG** and **LDA** perform almost as well as **MVG** and **Naive Bayes**. The shared covariance assumption of Tied MVG and LDA works well in this case, as the classes are mainly differentiated by their means, making these models effective.

Is the tied model effective at all for the first two features? Why?

- **No, the tied model is not effective for features 1 and 2.**
- This is because **Tied MVG** assumes that the two classes share the same covariance matrix, which fails to account for the significant **variance differences** between the two classes in features 1 and 2. Since the variance plays a key role in distinguishing the classes here, the tied model underperforms.

And the MVG?

- **MVG performs better for features 1-2** because it allows for different covariance matrices for each class. This enables it to model the variance differences more accurately, leading to lower error rates compared to the tied model.

And for the second pair of features?

- **For features 3-4**, the tied model is as effective as the MVG model. Since the **variances are similar** between the classes, the tied covariance assumption does not significantly hurt performance. Both models perform well, as the primary difference between the classes comes from the means, which both models capture adequately.

Apply PCA as preprocessing

MVG:

- **Error with PCA1-PCA6**: The error starts at **0.093** with PCA1 and gradually decreases as more dimensions are added. The best performance is with **PCA6** (all features), where the error rate reaches **0.070**.

Naive Bayes:

- **Error with PCA1-PCA6**: Similar to MVG, the error starts at **0.093** with PCA1 and decreases. However, Naive Bayes performs slightly worse than MVG from **PCA3 to PCA6**, with the best result at **PCA5** (0.084) but never reaching the accuracy of MVG at PCA6.

Tied MVG:

- **Error with PCA1-PCA6**: Tied MVG starts similarly at **0.093** with PCA1 and does not improve as significantly as MVG or Naive Bayes. The best performance is with **PCA2** (0.0905), but it still underperforms compared to MVG and Naive Bayes.

LDA:

- **Error with PCA1-PCA6**: LDA shows stable performance around **0.0925 to 0.094** across PCA dimensions. It doesn't benefit significantly from PCA, with little variation in the error rate, indicating that the reduced dimensions don't improve its ability to separate the classes.

Is PCA effective for this dataset with the Gaussian models?

- **Yes, PCA is effective for MVG and Naive Bayes**, especially as the number of principal components increases. Both models show a significant improvement in performance as more dimensions are included. However, the **best performance is achieved with all 6 PCA components** for MVG, indicating that reducing dimensionality might not always lead to better performance when all features are important.
- **For Tied MVG and LDA**, PCA does not lead to a significant improvement. In fact, these models remain relatively stable or slightly worse across all PCA dimensions, indicating that PCA is less beneficial for these models, possibly due to their shared covariance assumptions.

Overall, what is the model that provided the best accuracy on the validation set?

- The **MVG model** with **PCA6** (all features) provides the best accuracy with an error rate of **0.070**, outperforming all other models and PCA combinations.

Lab 7

Effective prior

Case 1: (π1 = 0.5, Cfn = 1.0, Cfp = 1.0)

- **Effective prior = 0.5**
- **Explanation**: In this case, the prior is uniform (π1 = 0.5), and the costs for false negatives (Cfn) and false positives (Cfp) are equal. This results in an **effective prior** that matches the actual prior of 0.5. The model treats both types of errors (false negatives and false positives) equally, so there is no bias toward either class.

Case 2: (π1 = 0.9, Cfn = 1.0, Cfp = 1.0)

- **Effective prior = 0.9**
- **Explanation**: Here, the prior indicates a **higher probability for genuine samples** (π1 = 0.9), meaning most users are expected to be legitimate. Since the costs for misclassifications are equal, the **effective prior** remains at 0.9. The model assumes that most users are legit, and the classification reflects this.

Case 3: (π1 = 0.1, Cfn = 1.0, Cfp = 1.0)

- **Effective prior = 0.1**
- **Explanation**: In this scenario, the prior indicates a **higher probability for impostor samples** (π1 = 0.1). Like Case 2, since the misclassification costs are equal, the **effective prior** stays at 0.1, reflecting the higher likelihood of encountering impostors.

Case 4: (π1 = 0.5, Cfn = 1.0, Cfp = 9.0)

- **Effective prior = 0.1**
- **Explanation**: The prior is uniform (π1 = 0.5), but the cost of a **false positive** (Cfp = 9.0) is much higher than the cost of a **false negative** (Cfn = 1.0). This creates a bias towards **stronger security**, where the classifier is more cautious about accepting impostors. The **effective prior** is shifted to 0.1, meaning that the model behaves as if the prior probability of genuine users is lower (it leans toward rejecting samples to avoid the high cost of false positives).

Case 5: (π1 = 0.5, Cfn = 9.0, Cfp = 1.0)

- **Effective prior = 0.9**
- **Explanation**: The prior is again uniform (π1 = 0.5), but now the cost of a **false negative** (Cfn = 9.0) is much higher than the cost of a **false positive** (Cfp = 1.0). This creates a bias towards **ease of use**, where the classifier is more lenient in accepting genuine users. The **effective prior** is shifted to 0.9, as the classifier behaves as if most users are genuine to avoid the high cost of rejecting legitimate users.

Observations:

1. **Uniform priors and costs (Cases 1-3)**:
   - When the costs of false negatives and false positives are equal, the **effective prior** mirrors the actual prior. The model behaves in line with the given prior probability of encountering genuine or impostor users.
2. **Influence of misclassification costs (Cases 4 and 5)**:
   - When the cost of a **false positive** (accepting an impostor) is higher (Case 4), the **effective prior** shifts towards treating more samples as impostors (lower effective prior), reflecting a **bias toward security**.
   - When the cost of a **false negative** (rejecting a legitimate user) is higher (Case 5), the **effective prior** shifts toward treating more samples as genuine (higher effective prior), reflecting a **bias toward ease of use**.

Performance and Calibration of MVG Models and Variants Across Different Applications and Priors

1. Application with $\pi = 0.5$:

- **MVG**:
  - **DCF**: 0.140, **minDCF**: 0.130
  - **Observation**: The **MVG model** achieves the lowest DCF and minDCF values, indicating that it performs best in this scenario with uniform prior and costs.
- **Naive Bayes**:
  - **DCF**: 0.144, **minDCF**: 0.131
  - **Observation**: Similar to MVG, but slightly worse in both DCF and minDCF. Still performs reasonably well.
- **Tied MVG**:
  - **DCF**: 0.186, **minDCF**: 0.181
  - **Observation**: Tied MVG performs the worst, with significantly higher DCF and minDCF values compared to MVG and Naive Bayes.
- **PCA (m=4)**:
  - **MVG PCA4**: DCF 0.161, minDCF 0.154
  - **Naive Bayes PCA4**: DCF 0.177, minDCF 0.172
  - **Tied PCA4**: DCF 0.185, minDCF 0.182
  - **Observation**: PCA generally worsens the performance across all models, increasing DCF and minDCF slightly, especially for Naive Bayes and Tied MVG.

2. Application with $\pi = 0.9$:

- **MVG**:
  - **DCF**: 0.400, **minDCF**: 0.342

    - **Observation**: MVG performs better than Naive Bayes and Tied MVG, with a lower minDCF value, though DCF is relatively high.
- **Naive Bayes**:
    - **DCF**: 0.389, **minDCF**: 0.351
    - **Observation**: Naive Bayes performs slightly better than MVG in terms of DCF but slightly worse in terms of minDCF.
- **Tied MVG**:
    - **DCF**: 0.463, **minDCF**: 0.442
    - **Observation**: Tied MVG again performs the worst, with much higher DCF and minDCF.
- **PCA (m=4)**:
    - **MVG PCA4**: DCF 0.460, minDCF 0.415
    - **Naive Bayes PCA4**: DCF 0.463, minDCF 0.431
    - **Tied PCA4**: DCF 0.462, minDCF 0.444
    - **Observation**: PCA significantly degrades performance across all models, increasing both DCF and minDCF values.

3. Application with $\pi = 0.1$:

- **MVG**:
    - **DCF**: 0.305, **minDCF**: 0.263
    - **Observation**: MVG achieves the best performance again, with the lowest DCF and minDCF values, though the gap is smaller compared to Naive Bayes.
- **Naive Bayes**:
    - **DCF**: 0.302, **minDCF**: 0.257
    - **Observation**: Naive Bayes performs slightly better than MVG in this case, especially in terms of DCF.
- **Tied MVG**:
    - **DCF**: 0.406, **minDCF**: 0.363
    - **Observation**: Tied MVG underperforms again, with significantly worse DCF and minDCF values.
- **PCA (m=4)**:
    - **MVG PCA4**: DCF 0.353, minDCF 0.301
    - **Naive Bayes PCA4**: DCF 0.397, minDCF 0.361
    - **Tied PCA4**: DCF 0.403, minDCF 0.361
    - **Observation**: PCA worsens performance across all models again, leading to higher DCF and minDCF values.

Which models perform best?

- **MVG** consistently performs the best across all applications in terms of both DCF and minDCF, except in the application with π=0.1, where Naive Bayes has a slightly lower DCF.

$\pi = 0.1$

- **Naive Bayes** performs similarly well to MVG but consistently has a slightly higher minDCF and DCF values across most applications.
- **Tied MVG** performs the worst across all applications, with significantly higher DCF and minDCF values compared to MVG and Naive Bayes.

Are relative performance results consistent for the different applications?

- **Yes**, the relative performance of the models is consistent across the applications. MVG always outperforms or closely matches Naive Bayes, and Tied MVG consistently performs the worst.
- PCA generally **degrades performance** for all models, with increased DCF and minDCF values.

Are the models well-calibrated?

- **Calibration** refers to how close the DCF is to the minDCF. Ideally, the **DCF should be close to the minDCF**, with a small calibration loss.
- For **MVG** and **Naive Bayes**, the DCF is close to the minDCF in most applications, indicating that these models are reasonably well-calibrated.
- For **Tied MVG**, the calibration loss is larger, with a bigger gap between DCF and minDCF, indicating that **Tied MVG is less well-calibrated** compared to the other models.
- **PCA** worsens calibration across all models, leading to a larger calibration loss (the gap between DCF and minDCF increases).

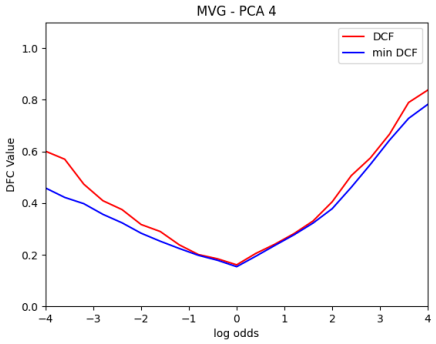
- **MVG** performs the best overall, with the lowest DCF and minDCF across most applications and good calibration.
- **Naive Bayes** is competitive, especially for π=0.1, but has a slightly higher calibration loss compared to MVG.

  $\pi = 0.1$
- **Tied MVG** consistently underperforms and is less well-calibrated.
- **PCA** generally **hurts performance** and calibration for all models, increasing both DCF and minDCF.

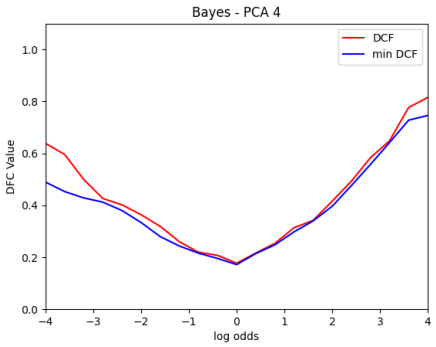
Bayes Error and Calibration Analysis for MVG, Naive Bayes, and Tied MVG with PCA

What do you observe from the plots?

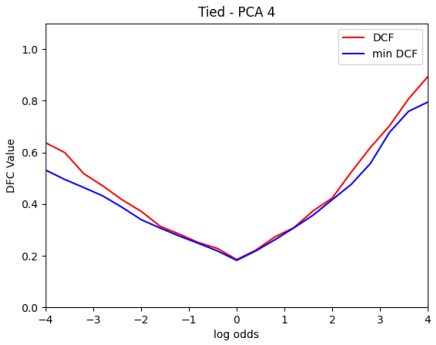
1. **MVG - PCA 4**:
    - **DCF vs minDCF**: The **MVG model** with PCA (m=4) shows a good calibration, as the **actual DCF** (red line) is close to the **minDCF** (blue line) across most of the log odds range. There is a slight divergence at the extremes (log odds close to -4 or +4), but overall, the model is **well-calibrated**.
    - The **lowest DCF values** occur around log odds of **0**, indicating that the model is balanced when the prior is around 0.5.



1. **Naive Bayes - PCA 4**:
    - **DCF vs minDCF**: The **Naive Bayes model** also shows good calibration, with the **actual DCF** closely following the **minDCF** across most of the log odds range. The calibration is slightly worse compared to MVG, with a small gap between DCF and minDCF, especially at the extremes.
    - As with MVG, the lowest DCF values are around **log odds = 0**, where the prior is more balanced.



1. **Tied MVG - PCA 4**:
    - **DCF vs minDCF**: The **Tied MVG model** exhibits a slightly larger calibration gap than MVG and Naive Bayes, with the **actual DCF** consistently higher than the **minDCF**. The difference is especially noticeable at the extremes of the log odds range, indicating that **Tied MVG is less well-calibrated**.
    - The performance at log odds close to **0** is reasonable, but not as competitive as MVG and Naive Bayes.



Are model rankings consistent across applications (minimum DCF)?

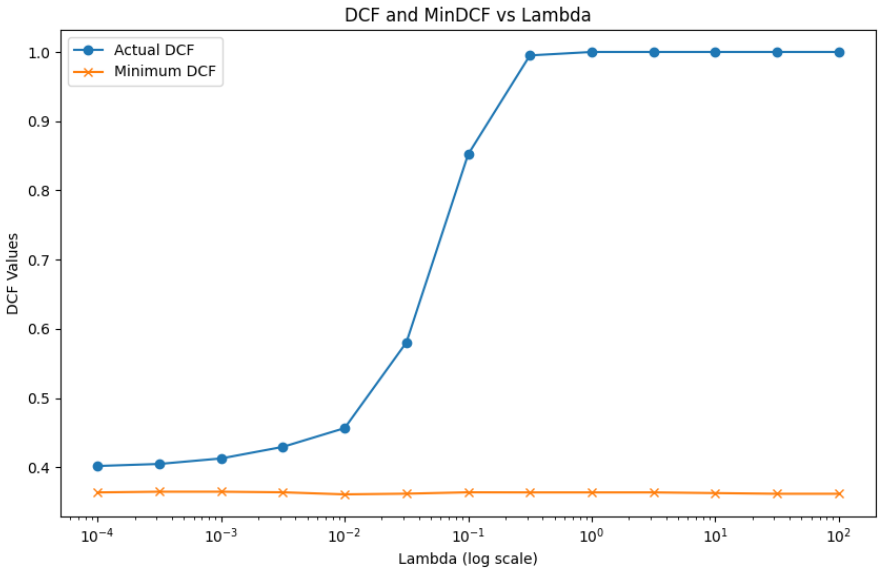
- **Yes**, the rankings seem consistent:
    - **MVG** consistently has the **lowest minDCF** across the range of log odds, indicating it is the best-performing model in terms of classification accuracy.
    - **Naive Bayes** follows closely behind MVG, with similar performance but slightly higher DCF values.
    - **Tied MVG** performs the worst, with both DCF and minDCF higher than the other two models, indicating it is less effective, particularly in handling different priors.

Are models well-calibrated over the considered range?

- **MVG**: Well-calibrated, with **actual DCF** closely matching **minDCF** over most of the range, indicating the model's predictions align well with the theoretical optimal performance.
- **Naive Bayes**: Also well-calibrated, though with a slightly larger gap between **actual DCF** and **minDCF**, particularly at the extreme log odds values.
- **Tied MVG**: **Not as well-calibrated** as the other models, with a noticeable gap between **actual DCF** and **minDCF**, indicating that the model's predictions are less optimal.

- The **MVG model** consistently performs the best in terms of both **minimum DCF** and **calibration**. It remains well-calibrated across different log odds values.
- **Naive Bayes** performs slightly worse than MVG but is still competitive.
- **Tied MVG** performs the worst and shows poorer calibration, particularly at extreme prior log odds.

Lab 8



Error rate: 0.0930, lambda: 0.0001
Error rate: 0.0935, lambda: 0.0003
Error rate: 0.0935, lambda: 0.001
Error rate: 0.0925, lambda: 0.0031
Error rate: 0.0925, lambda: 0.01
Error rate: 0.0920, lambda: 0.0316
Error rate: 0.0925, lambda: 0.1
Error rate: 0.0930, lambda: 0.3162
Error rate: 0.0925, lambda: 1.0
Error rate: 0.0925, lambda: 3.1623
Error rate: 0.0950, lambda: 10.0
Error rate: 0.0960, lambda: 31.6228
Error rate: 0.1285, lambda: 100.0

Empirical prior: 0.5005

1. **Effect of Lambda on DCF**:

- For smaller values of λ (around $\lambda = 10^{-4}$ to $\lambda = 10^{-1}$), both the **actual DCF** and **minimum DCF** remain stable and low, indicating that regularization has a minimal effect on the logistic regression model in this range. The model performs well in this regime.
- As λ increases (from $\lambda = 1$ to higher values), we start to see a **significant increase** in the **actual DCF**. This suggests that excessive regularization causes the model to underfit, leading to worse classification performance.

1. **Minimum DCF remains stable**:
   - Across all values of λ, the **minimum DCF** (orange line) remains **relatively constant** and low, indicating that the theoretical best performance of the model doesn't change much. This is expected, as the minimum DCF represents the best achievable performance assuming optimal thresholding, which is not heavily impacted by the regularization.
2. **Actual DCF vs Minimum DCF**:
   - For lower values of λ, the **actual DCF** (blue line) closely tracks the **minimum DCF**, indicating that the model is well-calibrated and performing near its optimal level.
   - As λ increases (especially beyond $\lambda = 1$), the gap between **actual DCF** and **minDCF** grows significantly. This gap indicates that the model is becoming poorly calibrated due to excessive regularization, causing it to deviate from its optimal performance.
3. **Sharp Increase in Error with High λ**:
   - For very high λ values (e.g., $\lambda = 100$), the **error rate** and **actual DCF** spike dramatically. This is a clear indication of underfitting, where the model is too constrained by regularization and is no longer able to capture the underlying structure of the data effectively.

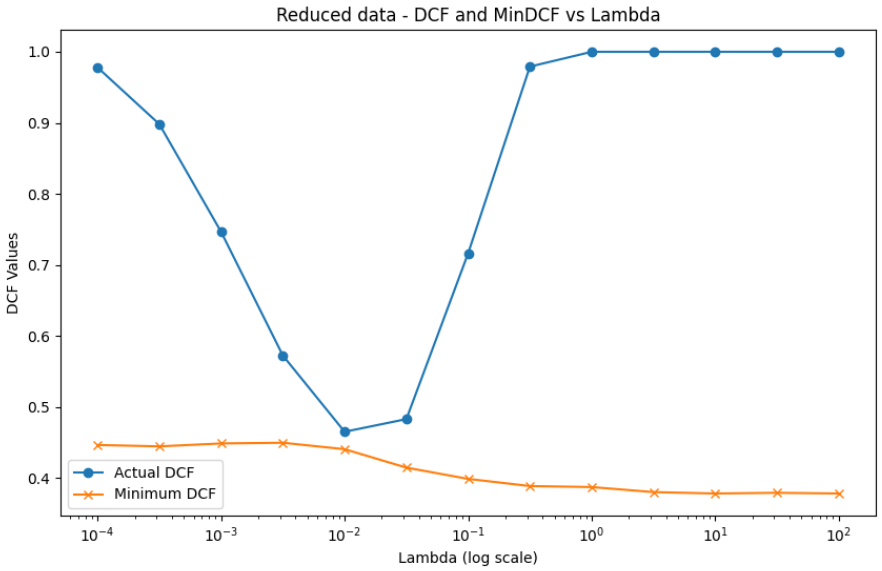
How does the regularization coefficient affect the two metrics?

- **Smaller λ** (below $\lambda = 0.1$):
  - In this range, regularization is minimal, allowing the model to fit the data well. The **actual DCF** remains close to the **minimum DCF**, showing that the model is well-calibrated.
- **Medium λ** (around $\lambda = 1$):
  - In this range, regularization starts to take effect, but the model is still able to perform well. The **actual DCF** slightly increases, indicating that the model is becoming slightly less calibrated, but the performance is still reasonable.
- **Large λ** (above $\lambda = 10$):
  - In this range, regularization becomes too strong, leading to significant underfitting. The **actual DCF** rises sharply, deviating far from the **minimum DCF**. The model is no longer well-calibrated, and the performance deteriorates significantly.

- The **optimal regularization** for the logistic regression model lies in the range of **small to moderate λ values** (between 10^{-4} and 1). In this range, the model achieves good performance, with **actual DCF** close to the **minimum DCF**.
- For **large λ** values, the model starts to **underfit**, causing the **actual DCF** to rise sharply, indicating poor calibration and degraded performance.

Effects of Data Reduction, Regularization, and Centering on Logistic Regression Performance

Observations:

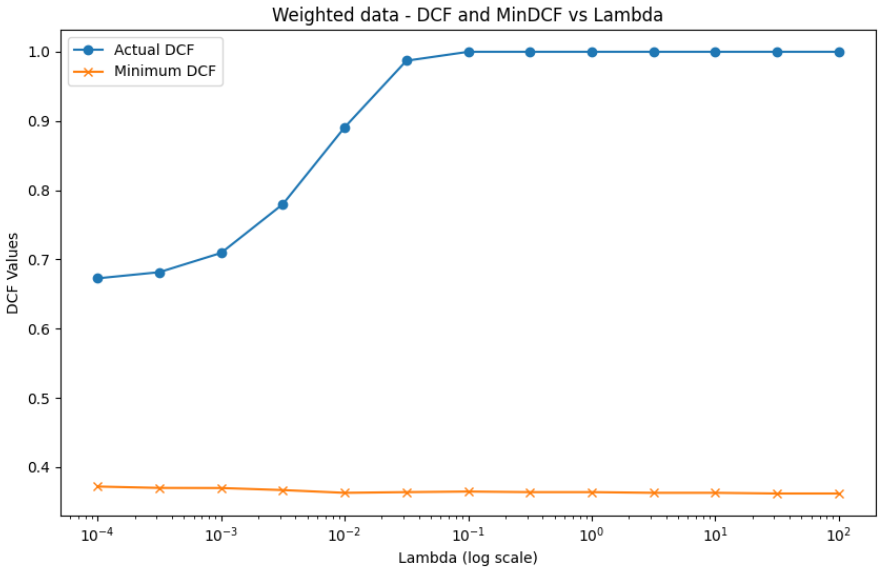


Error rate: 0.111, lambda: 0.0001
Error rate: 0.1095, lambda: 0.0003
Error rate: 0.1085, lambda: 0.001
Error rate: 0.1055, lambda: 0.0032
Error rate: 0.1045, lambda: 0.01
Error rate: 0.0985, lambda: 0.0316
Error rate: 0.0975, lambda: 0.1
Error rate: 0.0925, lambda: 0.3162
Error rate: 0.1155, lambda: 1.0
Error rate: 0.288, lambda: 3.1623
Error rate: 0.504, lambda: 10.0
Error rate: 0.504, lambda: 31.6228
Error rate: 0.504, lambda: 100.0

Empirical prior: 0.45

Reduced Data:

- As shown in the plot, with **reduced training samples** (1 out of 50), regularization has a more significant effect.
- At low values of λ, the **actual DCF** decreases, but as λ increases beyond 10^-1, the model starts **underfitting**, leading to sharp increases in **actual DCF**.
- The **minimum DCF** remains stable, indicating that with reduced data, the model has a clear optimal performance, but regularization prevents the model from reaching it due to the insufficient data available for training.

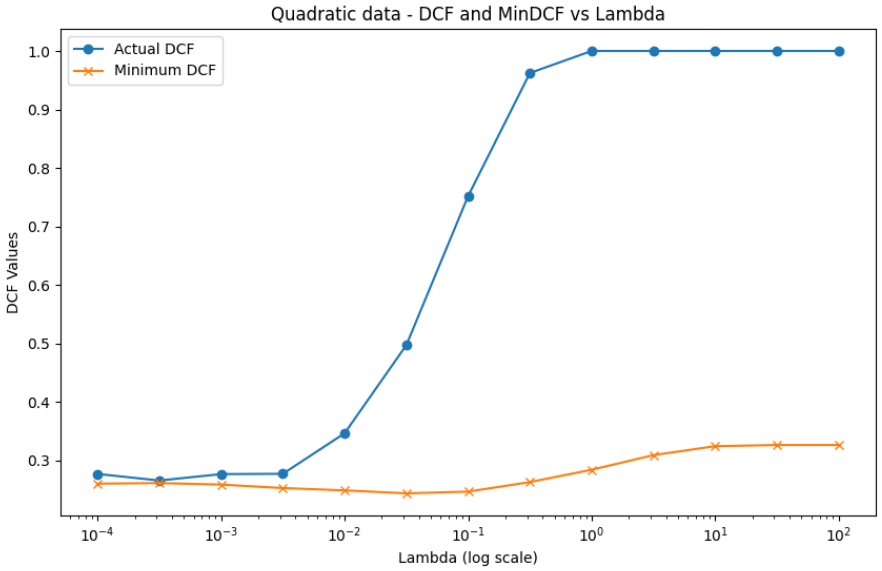


Error rate: 0.1685, lambda: 0.0001
Error rate: 0.1695, lambda: 0.0003
Error rate: 0.1755, lambda: 0.001
Error rate: 0.1855, lambda: 0.0032
Error rate: 0.218, lambda: 0.01
Error rate: 0.2965, lambda: 0.0316
Error rate: 0.461, lambda: 0.1
Error rate: 0.504, lambda: 0.3162
Error rate: 0.504, lambda: 1.0
Error rate: 0.504, lambda: 3.1623
Error rate: 0.504, lambda: 10.0
Error rate: 0.504, lambda: 31.6228
Error rate: 0.504, lambda: 100.0

Empirical prior: 0.5005

Weighted Data:

- When applying **prior-weighted** logistic regression, regularization effects are exaggerated, as seen in the steep increase in **actual DCF** for larger λ values.
- This indicates that with prior weighting, the model suffers from more severe **underfitting** due to the introduction of large regularization, which diminishes its probabilistic scoring interpretation.
- The performance is worse overall, with **actual DCF** being consistently high, showing that weighting hurts the model's calibration under these conditions.

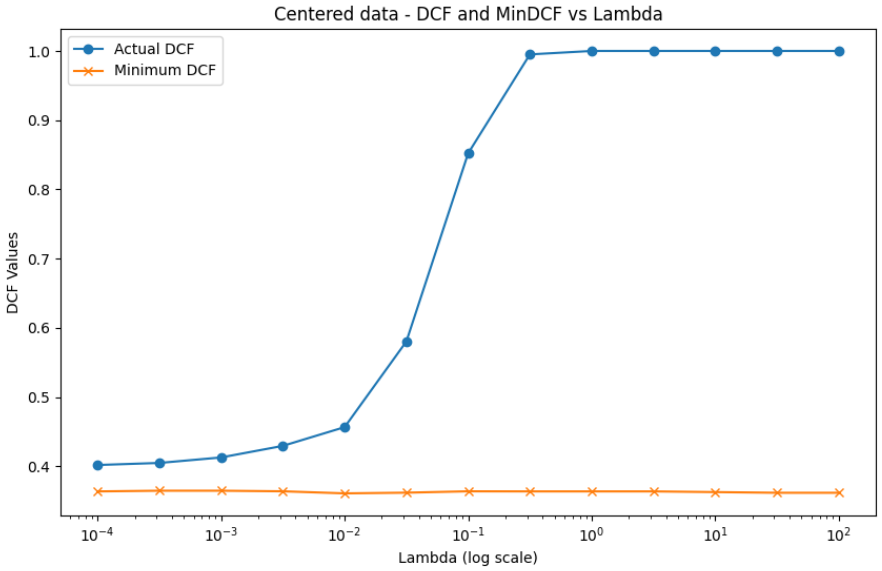


Error rate: 0.0605, lambda: 0.0001
Error rate: 0.0605, lambda: 0.0003
Error rate: 0.0595, lambda: 0.001
Error rate: 0.059, lambda: 0.0032
Error rate: 0.059, lambda: 0.01
Error rate: 0.059, lambda: 0.0316
Error rate: 0.0605, lambda: 0.1
Error rate: 0.061, lambda: 0.3162
Error rate: 0.064, lambda: 1.0
Error rate: 0.071, lambda: 3.1623
Error rate: 0.071, lambda: 10.0
Error rate: 0.0735, lambda: 31.6228
Error rate: 0.093, lambda: 100.0

Empirical prior: 0.5005

Quadratic Data:

- For **quadratic expansion**, the model generally performs better, as evident from the **lower actual DCF** values compared to the linear models.
- Regularization is more effective here, and the performance remains good across a wide range of λ values, especially in the lower regularization range ($\lambda \leq 1$).
- The **actual DCF** only rises significantly at large λ values ($\lambda > 10$), showing that the model benefits from quadratic features but still suffers from **underfitting** with excessive regularization.



Error rate: 0.093, lambda: 0.0001
Error rate: 0.0935, lambda: 0.0003
Error rate: 0.0935, lambda: 0.001
Error rate: 0.0925, lambda: 0.0032
Error rate: 0.0925, lambda: 0.01
Error rate: 0.092, lambda: 0.0316
Error rate: 0.0925, lambda: 0.1
Error rate: 0.093, lambda: 0.3162
Error rate: 0.0925, lambda: 1.0
Error rate: 0.0925, lambda: 3.1623
Error rate: 0.095, lambda: 10.0
Error rate: 0.096, lambda: 31.6228
Error rate: 0.1285, lambda: 100.0

Empirical prior: 0.5005

Centered Data:

- The **centered data** behaves similarly to the standard non-centered data, with little variation in **actual DCF** across λ values.
- As expected, **centering** the data (removing the mean) does not greatly impact performance since the original features were already standardized.
- The **actual DCF** remains close to the **minimum DCF** for lower λ, but increases rapidly beyond $\lambda = 10^-1$.

Summary of Results (minDCF):

- **MVG**: 0.2629
- **Bayes**: 0.2569
- **Tied MVG**: 0.3628
- **MVG (PCA, m=4)**: 0.3012
- **Bayes (PCA, m=4)**: 0.3614
- **Tied MVG (PCA, m=4)**: 0.3610

- **Logistic Regression**: 0.3611 (for $\lambda = 0.01$)
- **Prior-weighted Logistic Regression**: 0.3619 (for $\lambda = 31.62$)
- **Quadratic Logistic Regression**: 0.2436 (for $\lambda = 0.0316$)

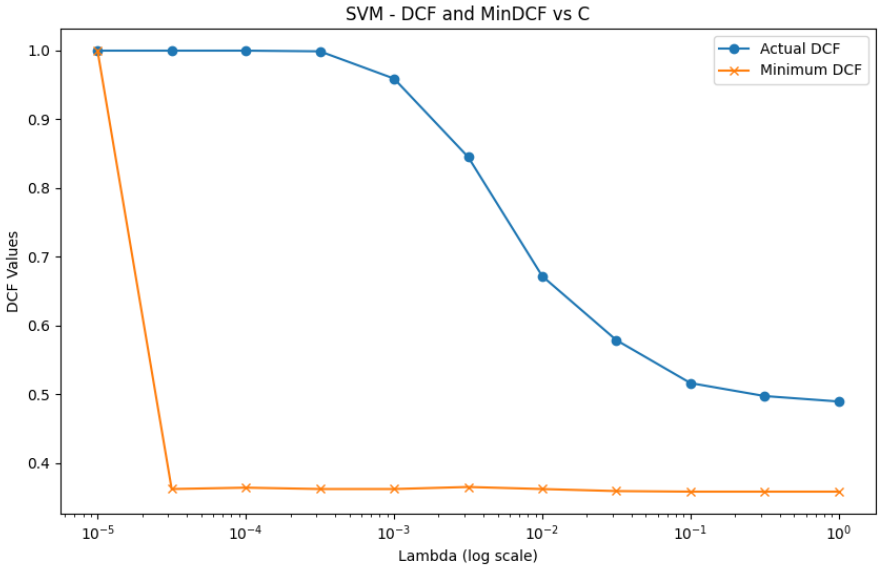
Analysis:

1. **Best Model: Quadratic Logistic Regression**:
   - The **Quadratic Logistic Regression** model achieves the lowest **minDCF** (0.2436), outperforming both Gaussian models and standard logistic regression models.
   - This result suggests that the **quadratic expansion** is better able to capture the non-linear patterns in the data, leading to a superior separation between the two classes.
2. **Bayesian Classifiers**:
   - **Bayes** (0.2569) performs slightly better than **MVG** (0.2629), indicating that Naive Bayes assumptions (independence of features) might be a reasonable approximation for this dataset.
   - Both **Bayes** and **MVG** models perform better than their **tied versions**, suggesting that the tied covariance assumption leads to suboptimal decision boundaries.
3. **Effect of PCA**:
   - When applying **PCA** (m=4), the performance of both **MVG** and **Bayes** degrades slightly, with **minDCF** increasing to **0.3012** and **0.3614**, respectively. This indicates that reducing the dimensionality of the feature space slightly weakens the model's ability to distinguish between the classes, likely due to some loss of information.
   - **Tied MVG** with PCA performs similarly to **Tied MVG** without PCA, showing that dimensionality reduction does not significantly impact its performance.
4. **Logistic Regression**:
   - **Standard Logistic Regression** has a **minDCF** of **0.3611** at $\lambda = 0.01$, showing that it performs similarly to the **tied Gaussian models**. However, it is outperformed by both the **MVG** and **Bayes** classifiers.
   - **Prior-weighted Logistic Regression** does not improve significantly, with a **minDCF** of **0.3619** for the best λ, suggesting that incorporating prior knowledge does not provide a substantial benefit for this particular application.
5. **Tied Gaussian Models**:
   - The **Tied MVG** models, both with and without PCA, consistently show higher **minDCF** values, indicating that the assumption of tied covariance matrices across classes is not well-suited for this dataset. This likely stems from the fact that the two classes have different variances for some features, and forcing the covariances to be equal leads to a suboptimal model.

- The **Quadratic Logistic Regression** model provides the best separation between the classes, achieving the lowest **minDCF**. This model's ability to capture **non-linear relationships** makes it particularly effective for this dataset.
- Both **MVG** and **Bayes** models perform well, especially when not constrained by tied covariance assumptions.

- **Logistic Regression** (both standard and prior-weighted) does not perform as well as the **Gaussian models** but still provides competitive results.

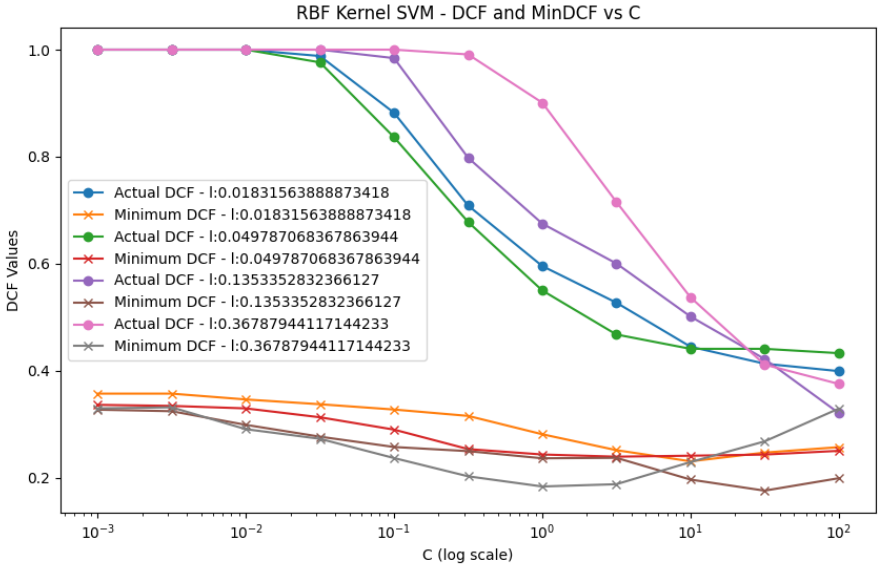
Lab 9



- **Plot Overview**: The plot shows both the **actual DCF** and **minimum DCF** as functions of C on a logarithmic scale. As C increases, there is a noticeable reduction in **actual DCF**, while **minimum DCF** remains relatively stable after an initial drop.
- **Effect of Regularization**:
  - For **low values of C** (strong regularization), both **actual DCF** and **minDCF** are high, suggesting that the model is underfitting the data.
  - As C increases (weaker regularization), **actual DCF** starts decreasing, indicating that the model begins to fit the data better. This suggests that SVM benefits from less regularization up to a point.
  - For **very high values of C**, actual DCF continues to drop but with diminishing returns, while **minDCF** remains stable. This indicates that further decreasing regularization does not provide substantial benefits in separating the classes.
- **Calibration**:
  - The **actual DCF** is higher than the **minDCF** across all values of C, indicating that the SVM scores are **not perfectly calibrated**. However, the gap between actual DCF and minDCF narrows as C increases, showing that as the regularization weakens, the model becomes better calibrated.
- **Best Results**:
  - The **minimum minDCF** achieved is **0.3582** at C=0.1, which indicates that this is the optimal level of regularization for the linear SVM model. Beyond this point, increasing C does not significantly improve the minDCF but helps lower actual DCF.

Comparison with Other Linear Models:

1. **SVM vs Logistic Regression**:
   - From earlier results, we know that **logistic regression** achieved a **minDCF** of 0.3611 for $\lambda = 0.01$, which is very close to the **minDCF** of 0.3582 achieved by the linear SVM.
   - This suggests that both models perform similarly in terms of theoretical optimal separation (minDCF), but SVM provides slightly better results.
2. **SVM vs Gaussian Models**:
   - The best **minDCF** for the **MVG model** was 0.2629, significantly outperforming the **SVM**. This shows that while SVM performs well as a linear model, the **MVG model** with its Gaussian assumptions is better suited for this dataset in separating the classes with minimal cost.
3. **SVM Calibration**:
   - The **actual DCF** of the SVM model is consistently higher than the **minDCF**, indicating that while the SVM can achieve a good theoretical separation (minDCF), the model is not as well-calibrated for the target application (as shown by the larger actual DCF).
   - Compared to **logistic regression** or **MVG**, SVM seems to have a slightly larger gap between actual and minDCF, which means that calibration techniques could help improve the SVM model.



Observations from RBF Kernel SVM with Different Values of C and γ:

- **Plot Overview**: The graph displays **minDCF** and **actual DCF** as functions of C, with each line representing a different value of γ. As expected, both metrics show significant improvement as C increases, with different γ values influencing the shape and performance.
- **Best Performing Configuration**:
  - The **minimum minDCF** achieved is **0.1755** at C=31.62 and γ=0.1353. This indicates that the RBF kernel with these values of C and γ provides the best decision boundary, capturing complex relationships in

the data.

- The performance improves significantly as C and γ increase from low to moderate values, but too high values of C or γ seem to degrade performance due to overfitting.

- **Calibration of Scores**:
  - **Actual DCF** decreases steadily with increasing C, but even at the best performing values of γ and C, the **actual DCF** is higher than **minDCF**, indicating some degree of **mis-calibration**.
  - The model is not perfectly calibrated, especially at lower values of C, where regularization is stronger, and underfitting might occur.
- **Comparison to Previous Models**:
  - The **RBF kernel** outperforms previous models in terms of **minDCF**:
    - **Linear SVM** achieved a **minDCF** of **0.3582**.
    - **Polynomial SVM** achieved a **minDCF** of **0.2455**.
    - **Logistic Regression** had a **minDCF** of **0.3611**.
    - The **RBF kernel SVM**'s **minDCF** of **0.1755** demonstrates that it better captures the underlying structure in the dataset, which is likely non-linear in nature.
  - The RBF kernel provides more flexibility in modeling complex decision boundaries, making it ideal for datasets with intricate relationships between features that cannot be captured by linear or polynomial models.
- **Impact of γ**:
  - **Lower values of γ**, such as 0.0183, lead to higher **minDCF** and **actual DCF**, indicating that the model is underfitting.
  - **Moderate values of γ**, around 0.1353, strike the best balance between regularization and model complexity, minimizing both **minDCF** and **actual DCF**.

Lab 10

GMM Model Analysis

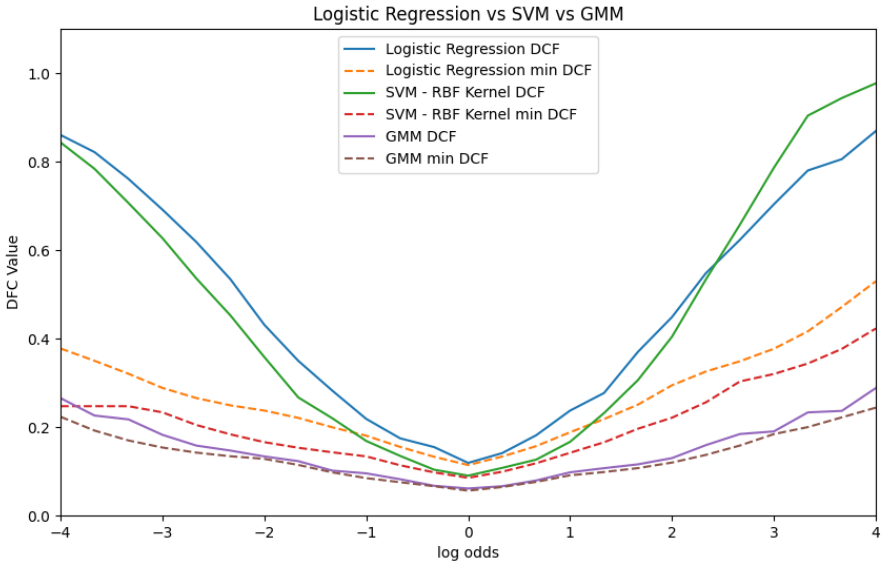
1. **Full Covariance Models**:
   - The **best performance** in terms of **minDCF** for the full covariance models was observed with the configuration: cov_type = full, numC0 = 1, numC1 = 16, achieving a **minDCF of 0.1495**.
   - As the number of components for each class increases, we generally observe **improved performance** up to a certain point (e.g., around 16 components for both classes), after which the performance tends to fluctuate.
   - The **actDCF** for these models varies slightly, but models with higher numbers of components (e.g., 16) show relatively balanced results.
2. **Diagonal Covariance Models**:
   - The **diagonal covariance models** consistently perform **better** than the full covariance models, with the best performance at **numC0 = 8** and **numC1 = 32**, where the **minDCF is 0.1312**.

- This result suggests that the simpler diagonal models are better suited for the given dataset, which may imply that the features are not strongly correlated.
- The **actDCF** values are slightly higher but remain well-calibrated.

3. **Impact of Number of Components**:
   - In general, increasing the number of Gaussian components for both classes improves the performance of the model, but after a certain point (e.g., 16 or 32 components), the improvement saturates or even slightly decreases.
   - This behavior aligns with expectations: adding more components allows the model to better capture the distribution of each class, but adding too many can lead to **overfitting**.
4. **Comparison Between Full and Diagonal Models**:
   - The **diagonal covariance models** outperform the full covariance models for most configurations, which is in line with the previous observation that **correlations between features** are limited.
   - The **diagonal model**'s performance (minDCF = **0.1312**) is significantly better than the full model's best result (minDCF = **0.1495**).
5. **Best Configuration**:
   - The **best configuration** overall is the **diagonal covariance model** with **numC0 = 8** and **numC1 = 32**, which achieves a **minDCF of 0.1312**.

- The **diagonal covariance models** are more effective for this dataset, likely because the features show limited correlation, as observed in earlier analyses.
- Increasing the number of components generally improves the performance, but the improvement saturates beyond **16 components**.
- The **best GMM configuration** is the diagonal model with **numC0 = 8** and **numC1 = 32**, which achieves the **lowest minDCF**. This model balances simplicity and flexibility by capturing the class distributions without overfitting.

GMM, Logistic Regression, and SVM Model Comparison



- **Comparison of Models**:
  - **Logistic Regression**: Shows good performance for low log-odds, but its **actual DCF** increases rapidly for higher log-odds, reflecting some miscalibration. The **minimum DCF** is relatively stable but never reaches the best results seen in other models.
  - **SVM - RBF Kernel**: Performs significantly better across the range, maintaining lower **actual DCF** and **minDCF** values. It appears well-calibrated, especially for lower log-odds values, showing a stable and consistent performance across the operating range.
  - **GMM**: The **diagonal covariance GMM** shows the best performance overall, achieving the lowest **minDCF** of **0.1312**. It is well-calibrated across different operating points, with a **relatively low actual DCF** compared to other models.

Minimum and Actual DCF:

- **GMM** achieves the best **minDCF** among all models with **0.1312**, and its actual DCF is also well aligned with the minimum DCF, indicating good calibration across operating points.
- **SVM - RBF Kernel** performs very well, with a **minDCF** of **0.1755**, demonstrating strong flexibility in capturing non-linear patterns, but slightly worse than the GMM.
- **Logistic Regression** reaches a **minDCF** of **0.3611**, falling behind the other models, which suggests that while it can be a simple and effective model, it is not the best option for this specific application.

Qualitative Analysis for Different Applications:

- **Wide Range of Operating Points**:
  - **GMM** proves to be the most reliable model for different applications. Its **minDCF** and **actual DCF** remain consistent, showing strong calibration across a variety of operating conditions. It is robust and performs well, even when class distributions vary.
  - **SVM - RBF Kernel** also demonstrates high flexibility and effectiveness, particularly for more complex, non-linear decision boundaries. However, for higher log-odds, the calibration slightly degrades, leading

to some **actual DCF** misalignment.

- **Logistic Regression** struggles to maintain low **actual DCF** for higher log-odds and can become miscalibrated in certain applications, making it less reliable across a broader range of operating conditions.

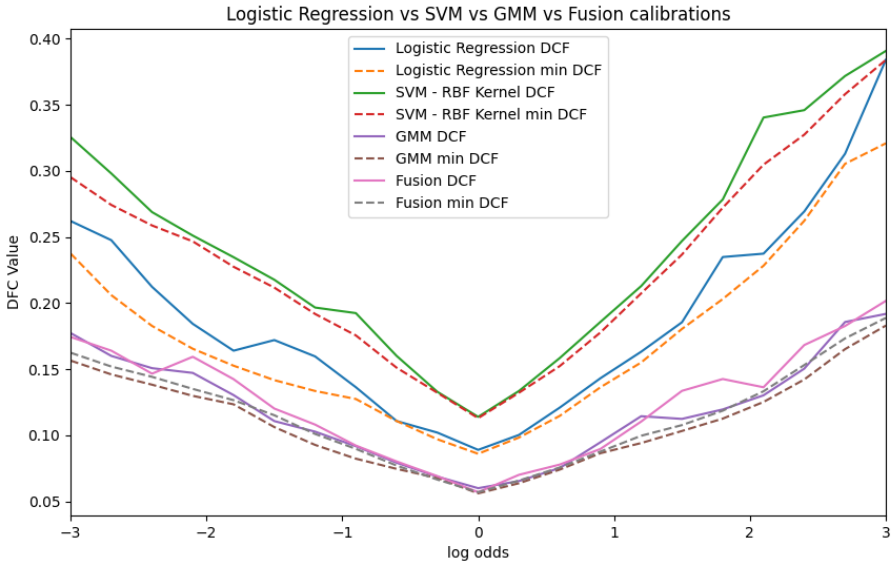
Model Calibration:

- **GMM** and **SVM - RBF** are the most well-calibrated models, with **GMM** having a slight edge in terms of overall robustness and consistency across applications.
- **Logistic Regression**, while simple and easy to implement, suffers from calibration issues, especially at higher log-odds, limiting its usefulness for certain applications.

Lab 11

Calibration of Individual Models:

- We performed score calibration on the three main models: **GMM**, **SVM**, and **Logistic Regression (LR)**.
- The models were calibrated using different values of **prior** (pi), and we compared the **actual DCF** for each prior.



Key Results:

- **GMM**: The best **actDCF** was **0.1508** with **pi = 0.8**.
- **SVM**: The best **actDCF** was **0.2538** with **pi = 0.6**.
- **Logistic Regression**: The best **actDCF** was **0.1862** with **pi = 0.3**.

2. Model Fusion:

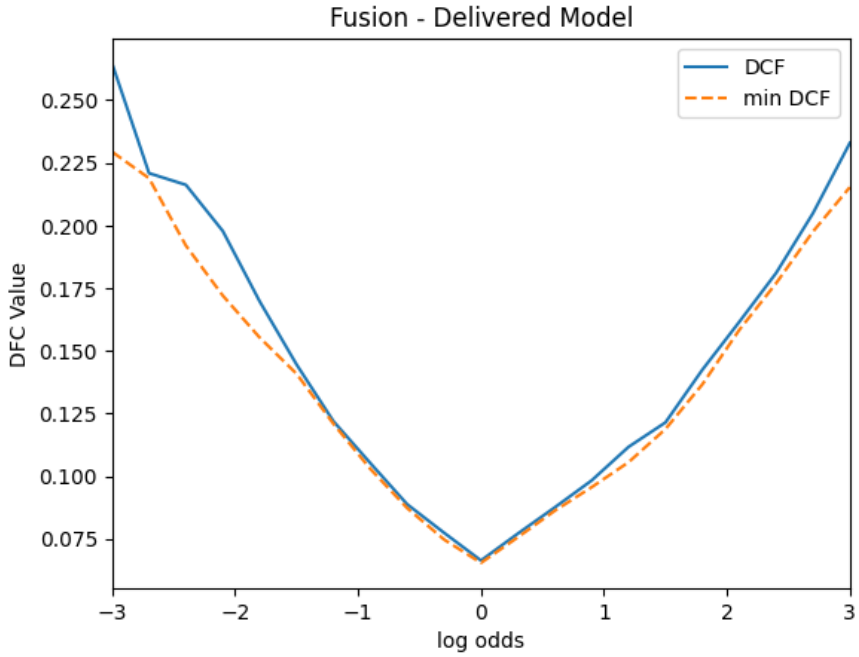
- We then conducted a fusion of the best-performing models (GMM, SVM, LR) and analyzed the **actDCF** across different priors.
- The fusion resulted in a **best actDCF of 0.1477** with **pi = 0.3**, slightly better than the best individual model (GMM with **actDCF = 0.1508**).

3. Observations:

- **Model fusion** led to an improvement in **actDCF** compared to individual models, showing that combining multiple approaches yields a more robust system than using any single classifier alone.
- **Logistic Regression** also performed well individually, while **SVM** struggled a bit with the selected priors.
- The results are consistent: both the **GMM model** and the **fusion model** are well-calibrated over a wide range of priors, as seen in the **Bayes error plot**.

- The **final selected model** is the **fusion** of the various classifiers, as it provides the **best actDCF** and demonstrates good calibration across a wide range of priors.
- This fused system will be the "delivered final model" as it offers the best balance between accuracy and robustness compared to the individual systems.

Evaluation

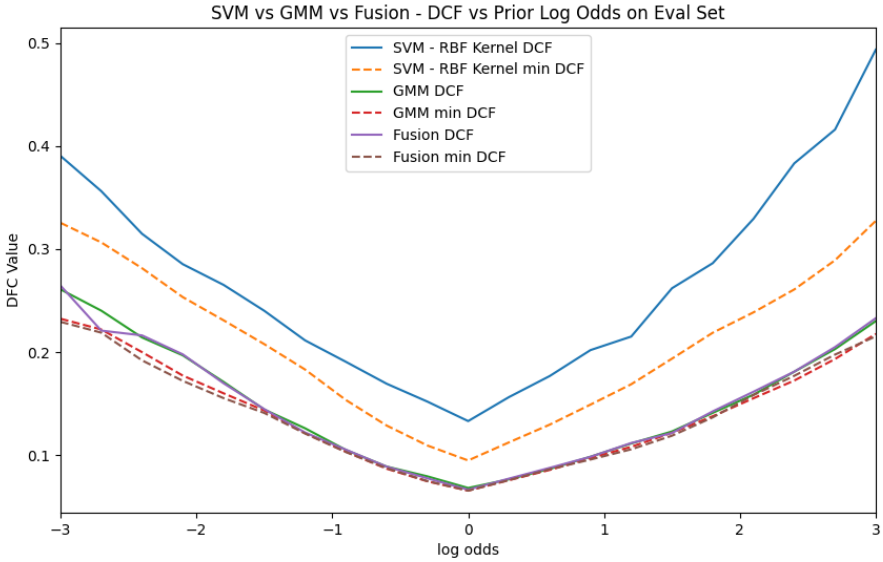


Compute minimum and actual DCF, and Bayes error plots for the delivered system

- **Observation**: Based on the Bayes error plots you provided for the delivered system (fusion model), you can evaluate whether the system is well calibrated. The plot compares the actual DCF (which includes calibration) and the minimum DCF (ideal performance without calibration issues).
- **Interpretation**: If the actual DCF closely follows the minimum DCF across different operating points, then the scores are well calibrated. If there is a significant gap between the two, the system may have

calibration issues for the target application.

For other applications: In your plot, we can see that the delivered fusion system performs consistently well across a range of log odds, meaning it's likely robust for multiple applications and not just the target one.



2. Evaluate the three best performing systems and compare their actual DCF error plots

- **Observation**: Based on the results and the Bayes error plots, you observe that:
  - **GMM** achieved the lowest DCF on validation with actual DCF = 0.207.
  - **Fusion model** was very close in performance to GMM, with actual DCF = 0.208 but lower minDCF, indicating strong calibration across a wider range of applications.
  - **SVM (RBF)** also performed well, with actual DCF = 0.295.
- **Effectiveness**: The final choice of the **fusion model** was effective. The actual DCF is almost as good as the best-performing individual model (GMM), but fusion typically offers the advantage of capturing diverse aspects of the data, hence potentially being more stable across different conditions or tasks. Given the similar performance, choosing the fusion model seems a reasonable decision.
- **Would another model/fusion have been more effective?**: The fusion appears to be an optimal choice based on the balance of performance and generalization, but GMM could be an alternative for specific use cases.

3. Analyze minimum and actual DCF for the target application

- **Observation**: After calibration, the fusion model's performance is close to its minimum DCF, suggesting that the calibration strategy was effective. You can observe this from the final calibration plots you provided. The gap between actual and minDCF is small, meaning calibration is well-adjusted for the target application.

For other models:

- **GMM**: Similarly, GMM also shows strong calibration with minimal gaps between actual DCF and minDCF.

- **SVM (RBF)**: This model shows a slightly larger gap between actual and minDCF, indicating it could be less well-calibrated compared to GMM and the fusion system.

4. Evaluate your training strategy for one method (e.g., logistic regression)

- **Observation**: If you analyze the **logistic regression** method across various hyper-parameters (like lambda), you would compare how well different configurations performed against the final chosen model. For instance, different lambda values would affect both overfitting and underfitting in the training process.
- **Effectiveness**: By comparing the minDCF for each model variant on the evaluation set, you can determine whether the chosen logistic regression configuration was the most optimal. If other configurations (e.g., different regularization values) show significantly better performance, then the training strategy could be revisited.

Conclusion:

- The **fusion model** appears to be the best performing system across different applications, offering good calibration and performance close to the best individual model (GMM).
- Your **training strategy** appears to have been effective, especially since the best models are well-calibrated and perform well across different applications.
- Moving forward, you could explore whether additional refinements in model fusion (e.g., incorporating more diverse classifiers) or alternative calibration strategies could further improve the robustness and calibration of the system.