S-319899

04/07/2024

Fingerprint Spoofing detection

# Project :

The project task consists of a binary classification problem. The goal is to perform fingerprint spoofing detection, i.e. to identify genuine vs counterfeit fingerprint images. The dataset consists of labeled samples corresponding to the genuine (True, label 1) class and the fake (False, label 0) class. The samples are computed by a feature extractor that summarizes high-level characteristics of a fingerprint image. The data is 6-dimensional. The training files for the project are stored in file Project/trainData.txt. The format of the file is the same as for the Iris dataset, i.e. a csv file where each row represents a sample. The first 6 values of each row are the features, whereas the last value of each row represents the class (1 or 0). The samples are not ordered.

## Feature analysis

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

A graph of a pyramid

Description automatically generatedA graph of a pyramid

Description automatically generated

A blue and orange dots

Description automatically generated

 **Classes Overlap:** Both features show significant overlap between the True and False classes, making it difficult to distinguish between the classes based on these features alone.

 **Means:** The means of the True and False classes for both features are similar and centered around 0.

 **Variances:** The variances of both classes are similar for each feature.

 **Modes:** The histograms for both features are unimodal, each showing a single peak centered around 0.

There is no clear separation or clustering observed in the scatter plot, indicating that these features alone may not be sufficient for distinguishing between the two classes.

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

A graph of different colored shapes

Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceA blue and orange dots

Description automatically generated

 **Classes Overlap:** Both features show slight overlap at the center for classes spoofed , authentic.

 **Means:**

* Feature 3 : The means of the two classes are different with the spoofed class having a lower mean than the Authentic class.
* Feature 4: The means of two classes are different with the authentic class having a lower mean than the Spoofed class.

 **Variances:** The variances of both classes are similar for each feature.

 **Modes:** The histograms for both features and both classes are unimodal.

There is a separation or clustering observed in the scatter plot, indicating that these features alone can be useful for distinguishing between the two classes.

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

A graph with different colored columns

Description automatically generatedA graph of different colored squares

Description automatically generatedA blue and orange dots

Description automatically generated

 **Classes Overlap:** Both features show slight overlap at -1 ,1 for classes spoofed , authentic.

 **Modes:** For both features the class authentic is bimodal having two modes close to -1 and 1 .

The class spoofed in unimodal.

 **Scatter Plot Observations:** We can notice 4 different clusters for each class , there is some separation indicating that the feature 5 and 6 can be useful for classification.

## Dimensionality Reduction

2.1 Apply PCA and LDA to the project data. Start analyzing the effects of PCA on the features. Plot the histogram of the projected features for the 6 PCA directions, starting from the principal (largest variance). What do you observe? What are the effects on the class distributions? Can you spot the different clusters inside each class?  
  
A graph of different colored bars

Description automatically generatedA graph of a graph

Description automatically generated with medium confidenceA graph of a number of objects

Description automatically generated with medium confidenceA graph of a diagram

Description automatically generated with medium confidenceA graph of a pyramid

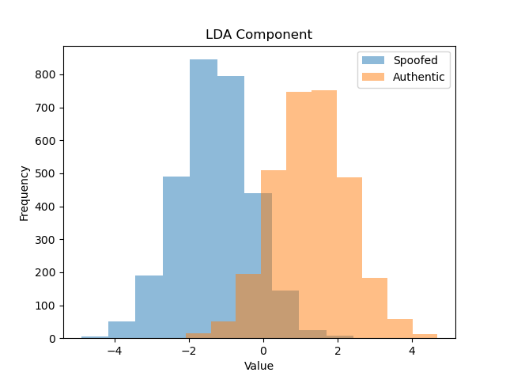
Description automatically generated with medium confidenceA graph of a graph

Description automatically generated with medium confidence

In the first principal component of PCA we can see it captures the variance and can be effective for separating the classes.

In all the other PCA components the classes overlap.  
I cannot spot different clusters inside either of the classes.

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



The classes overlap slightly around the center but still it has less overlap and better separation than the best feature class separation that was feature 3 and feature 4.

LDA found a good direction with less overlap.

2.3 Try applying LDA as classifier. Divide the dataset in model training and validation sets (you can reuse the previous function to split the dataset). Apply LDA, select the orientation that results in the projected mean of class True (label 1) being larger than the projected mean of class False (label 0), and select the threshold as in the previous sections, i.e., as the average of the projected class means. Compute the predictions on the validation data, and the corresponding error rate.

Now try changing the value of the threshold. What do you observe? Can you find values that improve the classification accuracy?

LDA without PCA:

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 186 (out of 2000 samples)  
Error rate: 9.3%

Changing the threshold to other values did not improve accuracy.

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

LDA with PCA (m=1):

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 187 (out of 2000 samples)

Error rate: 9.3%

LDA with PCA (m=2):

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 185 (out of 2000 samples)

Error rate: 9.2%

LDA with PCA (m=3):

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 185 (out of 2000 samples)

Error rate: 9.2%

LDA with PCA (m=4):

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 185 (out of 2000 samples)

Error rate: 9.2%

LDA with PCA (m=5):

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 186 (out of 2000 samples)

Error rate: 9.3%

LDA with PCA (m=6):

Labels: [0 0 1 ... 0 0 0]

Predictions: [0 0 1 ... 0 0 0]

Number of errors: 186 (out of 2000 samples)

Error rate: 9.3%

M=2 ,3,4 improved the classification by 0.1% so PCA it is not very beneficial when applied to the LDA classifier

## 3.Gaussian density estimation

Try fitting uni-variate Gaussian models to the different features of the different classes of the project dataset. For each class, for each component of the feature vector of that class, compute the ML estimate 3 for the parameters of a 1D Gaussian distribution. Plot the distribution density (remember that you have to exponentiate the log-density) on top of the normalized histogram (set density=True when creating the histogram, see Laboratory 2).

What do you observe? Are there features for which the Gaussian densities provide a good fit? Are there features for which the Gaussian model seems significantly less accurate?

A green and red graph

Description automatically generatedA green and red line graph

Description automatically generatedA green and red line graph

Description automatically generatedA green and red line graph

Description automatically generatedA green and red graph

Description automatically generatedA green and red line graph

Description automatically generatedA green and red line graph

Description automatically generatedA green and red graph

Description automatically generatedA green graph with a red line

Description automatically generatedA green and red line graph

Description automatically generatedA green and red graph

Description automatically generatedA green and red line graph

Description automatically generated

As we can see from the histograms features :1 ,2 ,3,4 are a good fit for the gaussian model and for features 5 and 6 the Gaussian model is less accurate for both class 0 (spoofed) and class 1 (Authentic)

## 4.Generative Classifiers

4.1 Apply the tied Gaussian model, and compare the results with MVG and LDA. Which model seems to perform better?

MVG - Error rate: 7.0%

LDA -Error rate : 9.3%

Tied Gaussian - Error rate: 9.3%

Naive Bayes - Error rate: 7.2%

The MVG performs best.

### 4.2 Covariance and correlation matrices for the two classes :

Class 0 covariance matrix:

[[ 6.00956506e-01 5.15866517e-05 1.90589145e-02 1.92529876e-02

1.28039402e-02 -1.34721598e-02]

[ 5.15866517e-05 1.44722543e+00 -1.61340110e-02 -1.58561474e-02

-2.64529141e-02 2.29139833e-02]

[ 1.90589145e-02 -1.61340110e-02 5.65348901e-01 -1.84344435e-03

-6.91446277e-03 1.68928322e-02]

[ 1.92529876e-02 -1.58561474e-02 -1.84344435e-03 5.41615202e-01

5.25171375e-03 1.35717775e-02]

[ 1.28039402e-02 -2.64529141e-02 -6.91446277e-03 5.25171375e-03

6.96067641e-01 1.58438399e-02]

[-1.34721598e-02 2.29139833e-02 1.68928322e-02 1.35717775e-02

1.58438399e-02 6.86519710e-01]]

Class 0 correlation matrix:

[[ 1.00000000e+00 5.53156127e-05 3.26977873e-02 3.37466904e-02

1.97968638e-02 -2.09743833e-02]

[ 5.53156127e-05 1.00000000e+00 -1.78367604e-02 -1.79095288e-02

-2.63560127e-02 2.29882544e-02]

[ 3.26977873e-02 -1.78367604e-02 1.00000000e+00 -3.33139656e-03

-1.10223563e-02 2.71155043e-02]

[ 3.37466904e-02 -1.79095288e-02 -3.33139656e-03 1.00000000e+00

8.55322509e-03 2.22569065e-02]

[ 1.97968638e-02 -2.63560127e-02 -1.10223563e-02 8.55322509e-03

1.00000000e+00 2.29196624e-02]

[-2.09743833e-02 2.29882544e-02 2.71155043e-02 2.22569065e-02

2.29196624e-02 1.00000000e+00]]

Class 1 covariance matrix:

[[ 1.44809527e+00 -1.47222433e-02 5.57010301e-03 1.57415883e-02

1.94971163e-02 -1.76682539e-04]

[-1.47222433e-02 5.53390796e-01 -1.12168681e-02 -9.06473359e-03

-1.46589901e-02 1.63492048e-02]

[ 5.57010301e-03 -1.12168681e-02 5.57480229e-01 2.75609663e-02

-3.76966451e-03 -1.45976943e-02]

[ 1.57415883e-02 -9.06473359e-03 2.75609663e-02 5.69657013e-01

-1.16983404e-02 3.49931863e-02]

[ 1.94971163e-02 -1.46589901e-02 -3.76966451e-03 -1.16983404e-02

1.34201767e+00 1.69454096e-02]

[-1.76682539e-04 1.63492048e-02 -1.45976943e-02 3.49931863e-02

1.69454096e-02 1.30371880e+00]]

Class 1 correlation matrix:

[[ 1.00000000e+00 -1.64459687e-02 6.19940380e-03 1.73317836e-02

1.39859734e-02 -1.28588787e-04]

[-1.64459687e-02 1.00000000e+00 -2.01948630e-02 -1.61447883e-02

-1.70101823e-02 1.92481371e-02]

[ 6.19940380e-03 -2.01948630e-02 1.00000000e+00 4.89072205e-02

-4.35821698e-03 -1.71229097e-02]

[ 1.73317836e-02 -1.61447883e-02 4.89072205e-02 1.00000000e+00

-1.33794547e-02 4.06054941e-02]

[ 1.39859734e-02 -1.70101823e-02 -4.35821698e-03 -1.33794547e-02

1.00000000e+00 1.28109397e-02]

[-1.28588787e-04 1.92481371e-02 -1.71229097e-02 4.06054941e-02

1.28109397e-02 1.00000000e+00]]

Class 0 :   
Covariances (off-diagonal values) are generally small compared to the variances, indicating weak relationships between the features.

Correlation Matrix:

Most off-diagonal elements are close to zero, indicating weak correlations between the features.

Class 1:

Covariances (off-diagonal values) are again small compared to the variances, indicating weak relationships between the features.

Correlation Matrix:

Similar to Class 0, most off-diagonal elements are close to zero, indicating weak correlations between the features.

**Naive Bayes Performance:** The weak correlations suggest that the Naive Bayes model, which assumes feature independence, might perform relatively well since the actual dependencies between features are minimal.

### 4.3

In Laboratory 4 we separately fitted a Gaussian density over each feature for each class. This corresponds to the Naive Bayes model. What can you conclude on the goodness of the Gaussian assumption? Is it accurate for all the 6 features? Are there features for which the assumptions do not look good?

The gaussian assumption is accurate for features 1, 2,3,4 and not good for features 5 and 6.

### 4.4

To analyze if indeed the last set of features negatively affects our classifier because of poor modeling assumptions, we can try repeating the classification using only feature 1 to 4 (i.e., discarding the last 2 features). Repeat the analysis for the three models. What do you obtain? What can we conclude on discarding the last two features? Despite the inaccuracy of the assumption for these two features, are the Gaussian models still able to extract some useful information to improve classification accuracy?

MVG (features 1-4) - Error rate: 8.0%

Tied Gaussian (features 1-4) - Error rate: 9.5%

Naive Bayes (features 1-4) - Error rate: 7.6%

We have an increase of the error rates as expected with mvg 1% , Tied .2% and Naïve Bayes .4%

Which can be a good tradeoff of volume and complexity to accuracy but the choice depends on the application.

The models still are able to extract useful information.

### 4.5

Repeat the classification using only features 1-2 (jointly), and then do the same using only features 3-4 (jointly), and compare the results of the MVG and tied MVG models. In the first case, which model is better? And in the second case? How is this related to the characteristics of the two classifiers? Is the tied model effective at all for the first two features? Why? And the MVG? And for the second pair of features?

MVG (features 1-2) - Error rate: 36.5%

Tied Gaussian (features 1-2) - Error rate: 49.5%

MVG (features 3-4) - Error rate: 9.4%

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

The models performed both worse for features 1-2 but MVG performed better.

The MVG model uses separate covariance matrices for each class, allowing it to capture differences in variances more effectively. In Laboratory 2 and 4, we observed that for features 1-2, the means are similar, but the variances differ significantly between the classes.

The tied model assumes a single shared covariance matrix for all classes, which fails to capture the differing variances between the classes.

For features 3-4 they both have an equal error rate.  
For features 3-4, the two classes mainly differ in their means, but their variances are similar.

The tied model's assumption of a shared covariance matrix is not a problem here because the variances are similar between classes.

### 4.6

We can analyze the effects of PCA as pre-processing. Use PCA to reduce the dimensionality of the feature space, and apply the three classification approaches. What do you observe? Is PCA effective for this dataset with the Gaussian models? Overall, what is the model that provided the best accuracy on the validation set?

MVG with PCA - Error rate: 8.1%

Tied Gaussian with PCA - Error rate: 9.2%

Naive Bayes with PCA - Error rate: 8.8%

The PCA can be considered effective since it reduces the dimensionality and increases the error rate of MVG by 1.1% , Tied -.1% and Naïve Bayes 1.6% which can be a good tradeoff.

Overall the best accuracy is reached by MVG 7% but Naïve Bayes has an error rate of 7.2% and is significantly lighter in terms of complexity as we are assuming independence between features.

## 5.Bayes optimal decision

Effective priors: [0.5, 0.9, 0.1, 0.1, 0.9]

**NO PCA**

**Application (pi=0.5, Cfn=1.0, Cfp=1.0)**

**MVG - Error rate: 7.0%**

**MVG - DCF: 0.140**

**MVG - minDCF: 0.130**

Application (pi=0.5, Cfn=1.0, Cfp=1.0)

Tied Gaussian - Error rate: 9.3%

Tied Gaussian - DCF: 0.186

Tied Gaussian - minDCF: 0.181

Application (pi=0.5, Cfn=1.0, Cfp=1.0)

Naive Bayes - Error rate: 7.2%

Naive Bayes - DCF: 0.144

Naive Bayes - minDCF: 0.131

**PCA with m=5**

**Application (pi=0.5, Cfn=1.0, Cfp=1.0)**

**MVG with PCA (m=5) - Error rate: 7.1%**

**MVG with PCA (m=5) - DCF: 0.142**

**MVG with PCA (m=5) - minDCF: 0.133**

Application (pi=0.5, Cfn=1.0, Cfp=1.0)

Tied Gaussian with PCA (m=5) - Error rate: 9.3%

Tied Gaussian with PCA (m=5) - DCF: 0.186

Tied Gaussian with PCA (m=5) - minDCF: 0.181

Application (pi=0.5, Cfn=1.0, Cfp=1.0)

Naive Bayes with PCA (m=5) - Error rate: 8.8%

Naive Bayes with PCA (m=5) - DCF: 0.175

Naive Bayes with PCA (m=5) - minDCF: 0.174

### Models' Performance Analysis:

**Minimum DCF Analysis:**

* **MVG Model:**
  + Without PCA: Consistently shows lower minDCF compared to Tied Gaussian and Naive Bayes models.
  + With PCA: Best performance with m=5 across all applications.
* **Tied Gaussian Model:**
  + Without PCA: Generally higher minDCF compared to MVG.
  + With PCA: Performs better with m=5 but still higher minDCF than MVG.
* **Naive Bayes Model:**
  + Without PCA: Comparable performance to MVG.
  + With PCA: Best performance with m=5.

**Actual DCF Analysis:**

* **MVG Model:**
  + Demonstrates small differences between DCF and minDCF, indicating good calibration.
* **Tied Gaussian Model:**
  + Shows larger differences between DCF and minDCF, indicating less effective calibration.
* **Naive Bayes Model:**
  + Similar to MVG with good calibration.

### Relative Performance Consistency:

* The MVG model, especially with PCA (m=5), consistently performs better across different applications.
* Tied Gaussian models generally show higher error rates and DCF, indicating they are less effective for this dataset.
* The Naive Bayes model performs comparably to MVG but slightly worse in some cases.

### Calibration Analysis:

* **Well-Calibrated Models:**
  + Both the MVG and Naive Bayes models demonstrate good calibration with actual DCFs close to minDCFs.
* **Less Well-Calibrated Models:**
  + The Tied Gaussian model shows a noticeable difference between actual DCF and minDCF, indicating less effective calibration.

### Best Models:

* The **MVG model with PCA (m=5)** shows the best performance in terms of both minimum DCF and actual DCF across different applications.
* The **Naive Bayes model** also performs well, but the MVG model with PCA (m=5) is slightly better.

#### Consider now the PCA setup that gave the best results for the π˜ = 0.1 configuration

**Application (pi=0.1, Cfn=1.0, Cfp=1.0)**

**MVG with PCA (m=5) - Error rate: 13.7%**

**MVG with PCA (m=5) - DCF: 0.304**

**MVG with PCA (m=5) - minDCF: 0.274**

**Application (pi=0.1, Cfn=1.0, Cfp=1.0)**

**Tied Gaussian with PCA (m=5) - Error rate: 16.8%**

**Tied Gaussian with PCA (m=5) - DCF: 0.405**

**Tied Gaussian with PCA (m=5) - minDCF: 0.365**

**Application (pi=0.1, Cfn=1.0, Cfp=1.0)**

**Naive Bayes with PCA (m=5) - Error rate: 16.6%**

**Naive Bayes with PCA (m=5) - DCF: 0.393**

**Naive Bayes with PCA (m=5) - minDCF: 0.354**

#### Consider now the PCA setup that gave the best results for the π˜ = 0.1 configuration (this will be our main application). Compute the Bayes error plots for the MVG, Tied and Naive Bayes Gaussian classifiers. Compare the minimum DCF of the three models for different applications, and, for each model, plot minimum and actual DCF. Consider prior log odds in the range (−4, +4). What do you observe? Are model rankings consistent across applications (minimum DCF)? Are models well-calibrated over the considered range? A graph with red and blue lines Description automatically generatedA graph with red and blue lines Description automatically generatedA graph with red and blue lines Description automatically generated

The model rankings are consistent and the models seem well calibrated as we can see small gaps.

## 6.Logistic Regression

#### 6.1 Train the model using different values for λ. You can build logarthimic-spaced values for λ using numpy.logspace. To obtain good coverage, you can use numpy.logspace(-4, 2, 13) (check the documentation). Train the model with each value of λ, score the validation samples and compute the corresponding actual DCF and minimum DCF for the primary application πT = 0.1. To compute actual DCF remember to remove the log-odds of the training set empirical prior. Plot the two metrics as a function of λ (suggestion: use a logartihmic scale for the x-axis of the plot - to change the scale of the x-axis you can use matplotlib.pyplot.xscale(’log’, base=10)). What do you observe? Can you see significant differences for the different values of λ? How does the regularization coefficient affects the two metrics?

A graph with a red line

Description automatically generated

 For small values of lambda both the Actual DCF and Minimum DCF are relatively low and stable. This suggests that the model performs well with minimal regularization.

 As lambda increases there is a sharp increase in the Actual DCF, which indicates that the model's performance degrades significantly with higher regularization.

 The Minimum DCF shows little change across all lambda

#### 6.2 Since we have a large number of samples, regularization seems ineffective, and actually degrades actual DCF since the regularized models tend to lose the probabilistic interpretation of the scores. To better understand the role of regularization, we analyze the results that we would obtain if we had fewer training samples. Repeat the prebvious analysis, but keep only 1 out of 50 model training samples, e.g. using data matrices DTR[:, ::50], LTR[::50] (apply the filter only on the model training samples, not on the validation samples, i.e., after splitting the dataset in model training and validation sets). What do you observe? Can you explain the results in this case? Remember that lower values of the regularizer imply larger risk of overfitting, while higher values of the regularizer reduce overfitting, but may lead to underfitting and to scores that lose their probabilistic interpretation

A graph with red line and blue line

Description automatically generated

 **Low Lambda** The Actual DCF remains relatively stable but higher than the Minimum DCF, indicating that the model is likely overfitting to the limited training data.

 **Moderate Lambda Values:** The Actual DCF starts to decrease slightly but then increases significantly. This indicates the point where regularization begins to effectively combat overfitting. As lambda increases, the model starts to generalize better, but excessive regularization leads to underfitting.

 **High Lambda Values:** The Actual DCF reaches its peak and becomes close to 1. This suggests that the model is severely underfitting

#### 6.3In the following we will again consider only the full dataset. Repeat the analysis with the prior-weighted version of the model (remember that, in this case, to transform the scores to LLRs you need to remove the log-odds of the prior that you chose when training the model). Are there significant differences for this task? Are there advantages using the prior-weighted model for our application (remember that the prior-weighted model requires that we know the target prior when we build the model)?

A graph of a logistic regression

Description automatically generated

Compared to the model trained with unweighted model in the full dataset there is no significant difference.

#### 6.4Repeat the analysis with the quadratic logistic regression model (again, full dataset only). Expand the features, train and evaluate the models (you can focus on the standard, non prior-weighted model only, as the results you would obtain are similar for the two models), again considering different values for λ. What do you observe? In this case is regularization effective? How does it affect the two metrics?

A graph of a graph with a line

Description automatically generated with medium confidence

In this case the regularization effect is very effective for small values of lambda.

Actual DCF converges closer to minDCF for small values of Lambda but then diverges for large values of Lambda.

#### 6.5 The non-regularized model is invariant to affine transformations of the data. However, once we introduce a regularization term affine transformations of the data can lead to different results. Analyze the effects of centering (optionally, you can also try different strategies, including Z-normalization and whitening, as well as PCA) on the model results. You can restrict the analysis to the linear model. Remember that you have to center both datasets with respect to the model training dataset mean, i.e., you must not use the validation data to estimate the pre-processing transformation. For this task, you should observe only minor variations, as the original features were already almost standardized.

A graph with red line and blue line

Description automatically generated

#### 6.6 For the moment, we focus on selecting the models that optimize the minimum DCF on our validation set. Compare all models that you have trained up to now, including Gaussian models, in terms of minDCF for the target application πT = 0.1. Which model(s) achieve(s) the best results? What kind of separation rules or distribution assumptions characterize this / these model(s)? How are the results related to the characteristics of the dataset features?

Application (pi=0.1, Cfn=1.0, Cfp=1.0)

MVG - Error rate: 13.8%

MVG - DCF: 0.305

MVG - minDCF: 0.263

Application (pi=0.1, Cfn=1.0, Cfp=1.0)

Tied Gaussian - Error rate: 16.8%

Tied Gaussian - DCF: 0.406

Tied Gaussian - minDCF: 0.363

Application (pi=0.1, Cfn=1.0, Cfp=1.0)

Naive Bayes - Error rate: 13.6%

Naive Bayes - DCF: 0.302

Naive Bayes - minDCF: 0.257

PCA with m=5

Application (pi=0.1, Cfn=1.0, Cfp=1.0)

MVG with PCA (m=5) - Error rate: 13.7%

MVG with PCA (m=5) - DCF: 0.304

MVG with PCA (m=5) - minDCF: 0.274

Application (pi=0.1, Cfn=1.0, Cfp=1.0)

Tied Gaussian with PCA (m=5) - Error rate: 16.8%

Tied Gaussian with PCA (m=5) - DCF: 0.405

Tied Gaussian with PCA (m=5) - minDCF: 0.365

Application (pi=0.1, Cfn=1.0, Cfp=1.0)

Naive Bayes with PCA (m=5) - Error rate: 16.6%

Naive Bayes with PCA (m=5) - DCF: 0.393

Naive Bayes with PCA (m=5) - minDCF: 0.354

Logistic Regression unweighted :

minDCF = 0.364

Logistic Regression unweighted downsampled :   
 minDCF = 0.445

Logistic regression prior weighted version:

minDCF = 0.372

Quadratic Logistic regression :

minDCF = 0.260

Centered Dataset :

minDCF = 0.364

In terms of minDCF , Gaussian Naïve bayes classifier performs best with minDCF = 0.257 for the application pt = 0.1 .

The Naive Bayes classifier assumes feature independence given the class label and that each feature follows a Gaussian distribution. This model's separation rule is based on calculating the probability of each feature independently and then combining them.

The dataset features are likely to be nearly independent and normally distributed within each class.

## 7.SVM

#### 7.1 Apply the SVM to the project data. Start with the linear model (to avoid excessive training time we consider only the models trained with K = 1.0). Train the model with different values of C. As for logistic regression, you should employ a logarithmic scale for the values of C. Reasonable values are given by numpy.logspace(-5, 0, 11). Plot the minDCF and actDCF (πT = 0.1) as a function of C (again, use a logarithmic scale for the x-axis). What do you observe? Does the regularization coefficient significantly affect the results for one or both metrics (remember that, for SVM, low values of C imply strong regularization, while large values of C imply weak regularization)? Are the scores well calibrated for the target application? What can we conclude on linear SVM? How does it perform compared to other linear models? Repeat the analysis with centered data. Are the result significantly different?

A graph with a blue line

Description automatically generatedA graph with a line

Description automatically generated

Observations on Regularization Coefficient Effects

Linear SVM (Uncentered Data)

* **High regularization (low C)**: Both actual DCF and minimum DCF are high, indicating that the model is overly constrained and underfitting the data.
* **Medium regularization (medium C)**: Both actual DCF and minimum DCF decrease significantly, suggesting that the model starts to fit the data better.
* **Low regularization (high C)**: Minimum DCF stabilizes, while actual DCF shows improvement but still remains higher than minimum DCF, suggesting some overfitting.

Same for centered data

Effect of Regularization Coefficient

* **Significant impact**: The regularization coefficient significantly affects the actual DCF more than the minimum DCF. Strong regularization (low C) leads to high actual DCF (underfitting), while weak regularization (high C) reduces actual DCF but may lead to overfitting.
* **Minimum DCF**: The minimum DCF decreases with decreasing regularization (increasing C), indicating that the model fits the training data better with less regularization

**For high values of C the model scores are well calibrated**

**For low and mid values poorly calibrated.**

Compared to the other models :

In the best case with high values of C meaning low regularization it still has a higher DCF min value and higher distance between actual DCF and minDCF so I also more poorly calibrated.

#### 7.2 We now consider the polynomial kernel. For simplicity, we consider only the kernel with d = 2, c = 1 (but better results may be possible with different configurations), and we set ξ = 0, since the kernel already implicitly accounts for the bias term (due to c = 1). We also consider only the original, non-centered features (again, different pre-processing strategies may lead to better results). Train the model with different values of C, and compare the results in terms of minDCF and actDCF. What do you observe with quadratic models? In light of the characteristics of the dataset and of the classifier, are the results consistent with previous models (logistic regression and MVG models) in terms of minDCF? What about actDCF?

A graph of a graph

Description automatically generated

In terms of minDCF = 0.258 is almost equal to our best model yet :

Naive Bayes - DCF: 0.302

Naive Bayes - minDCF: 0.257

But actual DCF at that C is 0.388   
which suggests that it is less calibrated.

#### 7.3

#### For RBF kernel we need to optimize both γ and C (since the RBF kernel does not implicitly account for the bias term we set ξ = 1). We adopt a grid search approach, i.e., we consider different values of γ and different values of C, and try all possible combinations. For γ we suggest you analyze values γ ∈ e −4 , e−3 , e−2 , e−1 , while for C, to avoid excessive time but obtain a good coverage of possible good values we suggest log-spaced values numpy.logspace(-3, 2, 11) (of course, you are free to experiment with other values if you so wish). Train all models obtained by combining the values of γ and of C. Plot minDCF and actDCF as a function of C, with a different line for each value of γ (i.e., four lines for minDCF and four lines for actDCF). Analyze the results. Are there values of γ and C that provide better results? Are the scores well calibrated? How the result compare to previous models? Are there characteristics of the dataset that can be better captured by RBF kernels?

A graph with different colored lines

Description automatically generated

For gamma = 0.1353 and C 30 we have the minDCF = 0.176 and actDCF = 0.4216  
  
which is the best minDCf we have until now but with large distance to actDCF and indicates poor calibration.

The RBF kernel is advantageous in scenarios where the data exhibits non-linear relationships, local dependencies, and complex distributions. By transforming the data into a higher-dimensional space, the RBF kernel allows for more flexible and accurate classification boundaries, often leading to better performance in terms of both minDCF and actDCF.

## 8.Gaussian Mixture Models

### In this section we apply the GMM models to classification of the project data.

#### 8.1 For each of the two classes, we need to decide the number of Gaussian components (hyperparameter of the model). Train full covariance models with different number of components for each class (suggestion: to avoid excessive training time you can restrict yourself to models with up to 32 components). Evaluate the performance on the validation set to perform model selection (again, you can use the minimum DCF of the different models for the target application). Repeat the analysis for diagonal models. What do you observe? Are there combinations which work better? Are the results in line with your expectation, given the characteristics that you observed in the dataset? Are there results that are surprising? (Optional) Can you find an explanation for these surprising results?

A graph with red and blue lines

Description automatically generatedA graph with red and blue lines

Description automatically generated

The GMM with full covariance matrix performs best with 16 components :  
minDCF = 0.163

actDCF = 0.175

The GMM with diagonal covariance matrix performs best in two points :

8 components:

minDCF = 0.146

actDCF = 1.81

4 components:

minDCF = 0.148

actDCF = 0.168

better calibrated.

I expected the GMM with full covariance matrix to perform better because it can capture data relationships better and eventhough it has a higher minDCF it is better calibrated with

actDCF-minDCF = 0.12

.

For the GMM with diagonal Covariance matrix the model performs good even with two components but it is more poorly calibrated and may indicate overfitting .

#### 8.2 We have analyzed all the classifiers of the course. For each of the main methods (GMM, logistic regression, SVM — we ignore MVG since its results should be significantly worse than those of the other models, but feel free to test it as well) select the best performing candidate. Compare the models in terms of minimum and actual DCF. Which is the most promising method for the given application?

Linear Regression   
Best performance model :   
Quadratic Logistic regression :

minDCF = 0.260

A graph of a graph with a line

Description automatically generated with medium confidence

SVM   
Best performance model :

Quadratic polynomial kernel svm with C = 10  
A graph of a graph

Description automatically generated

minDCF = 0.258 but act dcf is close to 0.4 so it is poorly calibrated

RBF kernel svm :   
For gamma = 0.1353 and C 30 we have the minDCF = 0.176 and actDCF = 0.4216

GMM

The GMM with full covariance matrix performs best with 16 components :  
minDCF = 0.163

actDCF = 0.175

The GMM with diagonal covariance matrix performs best in two points :

8 components:

minDCF = 0.146

actDCF = 1.81

4 components:

minDCF = 0.148

actDCF = 0.168

better calibrated.

We conclude that the GMM performs best in our dataset   
I would choose GMM model with full covariance matrix and 16 components.

#### 8.3 Now consider possible alternative applications. Perform a qualitative analysis of the performance of the three approaches for different applications (keep the models that you selected in the previous step). You can employ a Bayes error plot and visualize, for each model, actual and minimum DCF over a wide range of operating points (e.g. log-odds ranging from −4 to +4). What do you observe? In terms of minimum DCF, are the results consistent, preserving the relative ranking of the systems? What about actual DCF? Are there models that are well calibrated for most of the operating point range? Are there models that show significant miscalibration? Are there models that are harmful for some applications? We will see how to deal with these issue in the last laboratory

Selecting Quadratic Logistic regression with lambda = 0.005

A graph with red lines

Description automatically generated

Selecting RBF kernel dual svm with gama = 0.135 and C = 30  
A graph with red lines

Description automatically generated

GMM

The GMM with full covariance matrix performs best with 16 components .

A graph with blue lines

Description automatically generated

In terms of DCF the results are consistent with our previous ranking, gmm full matrix with 16 components still performs best and is well calibrated.

For the Selecting RBF kernel dual svm with gama = 0.135 and C = 30 the model seems poorly calibrated for log odds -4 to -1 and -1 to -4

The quadratic linear regression is poorly calibrated for the range -4 -1 then it performs better.

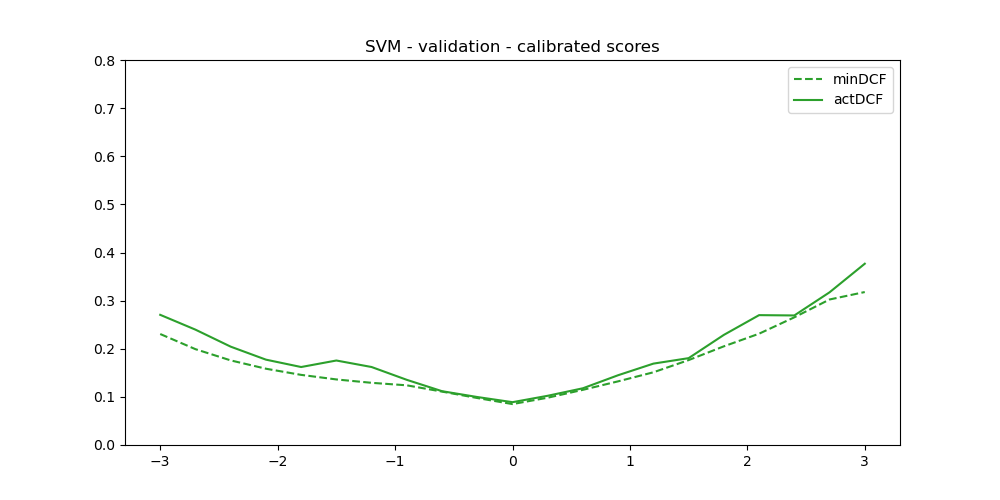
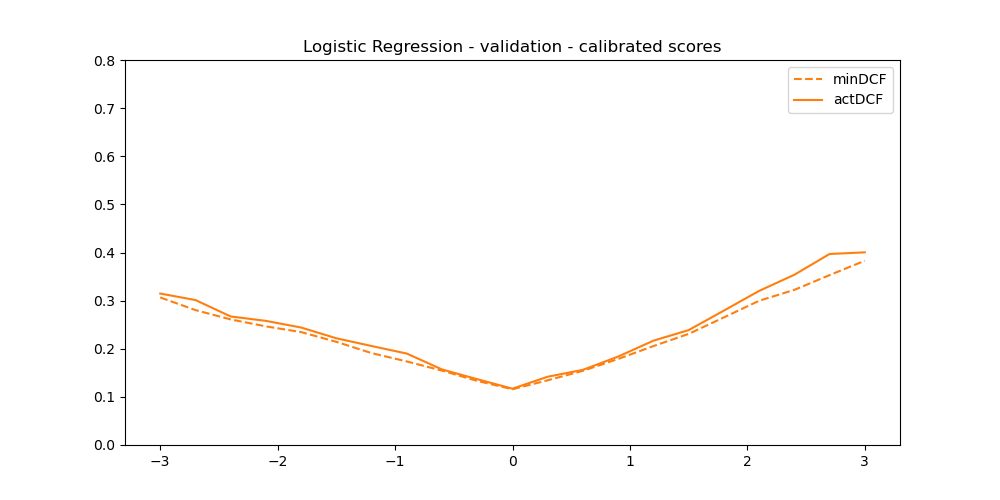
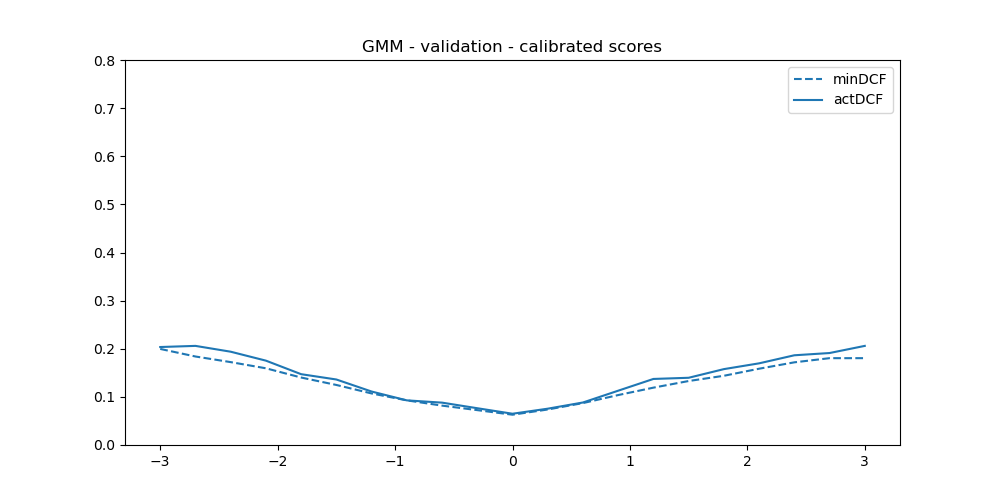
The svm with RBF kernel can me harmful for some applications .

## 9.Calibration and fusion

#### 9.1 Calibration and fusion Consider the different classifiers that you trained in previous laboratories. For each of the main methods (GMM, logistic regression, SVM — see Laboratory 10) compute a calibration transformation for the scores of the best-performing classifier you selected earlier. The calibration model should be trained using the validation set that you employed in previous laboratories (i.e., the validation split that you used to measure the systems performance). Apply a K-fold approach to compute and evaluate the calibration transformation. You can test different priors for training the logistic regression model, and evaluate the performance of the calibration transformation in terms of actual DCF for the target application (i.e., the training prior may be different than the target application prior, but evaluation should be done for the target application). For each model, select the best performing calibration transformation (i.e. the one providing lowest actual DCF in the K-fold cross validation procedure for the target application). Compute also the minimum DCF, and compare it to the actual DCF, of the calibrated scores for the different systems. What do you observe? Has calibration improved for the target application? What about different applications (Bayes error plots)?

A graph of different colored lines

Description automatically generated



GMM with full covariance matrix and 16 component:

minDCF (0.1) = 0.158

actDCF (0.1) = 0.179

GMM - Calibrated with full covariance matrix and 16 component:

minDCF (0.1) = 0.166

actDCF (0.1) = 0.178

Quadratic Logistic Regression for lambda = 0.005

minDCF (0.1) = 0.250

actDCF (0.1) = 0.304

Quadratic Logistic Regression for lambda = 0.005 - Calibrated:

minDCF (0.1) = 0.251

actDCF (0.1) = 0.265

SVM with RBF kernel and C = 30:

minDCF (0.1) = 0.171

actDCF (0.1) = 0.430

SVM – Calibrated with RBF kernel and C = 30:

minDCF (0.1) = 0.164

actDCF (0.1) = 0.191

Calibration has improved the target application in all models.

And also for other applications it is more noticeable for svm uncalibrated where for logpriors = -3 ,-2

It has a large distance from actdfc to mindfc and as we can see on the calibrated version it is improved and even on the target application it reduces the act dcf significantly.

#### 9.2 Compute a score-level fusion of the best-performing models. Again, you can try different priors for training logistic regression, but you should select the best model in terms of actual DCF computed for the target application. Compute also the minimum DCF of the resulting model. How is the fusion performing? Is it improving actual DCF with respect to single systems? Are the fused scores well calibrated?

A graph with red lines

Description automatically generated

Fusion:

minDCF (0.1) = 0.134

actDCF (0.1) = 0.164

It is performing well and it is improving actDCF compared to single system and the scores seem well calibrated.

#### 9.3 Choose the final model that will be used as “delivered” system, i.e. the final system that will be used for application data. Justify your choice.

 GMM:

* The GMM model performs well, with a relatively low actDCF of 0.179. Calibration does not significantly improve the DCF, suggesting the model is already well-calibrated.

 Quadratic Logistic Regression:

* The Logistic Regression model has higher DCF values both before and after calibration. Calibration does improve the actDCF significantly, but it still lags behind other models.

 SVM:

* The SVM model shows a drastic improvement in actDCF after calibration (from 0.430 to 0.191). This indicates that the initial model was poorly calibrated, and calibration greatly improved its performance.

 Fusion:

* The fusion of all models achieves the best performance, with the lowest minDCF of 0.134 and actDCF of 0.164.

 Chosen Final Model: Fusion of GMM, Quadratic Logistic Regression, and SVM with RBF kernel.

 Reason: It provides the best performance in terms of both minimum DCF and actual DCF for the target application, indicating superior and more consistent performance.

## 10.Evaluation

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

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

A graph with red lines

Description automatically generated

Fusion of the 3 models:

minDCF (0.1) = 0.199

actDCF (0.1) = 0.209

The fusion model demonstrates the best calibration with the lowest minDCF and actDCF, indicating it is well-calibrated for the target application and also for the other applications.

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

GMM with full covariance matrix and 16 component:

minDCF (0.1) = 0.202 - actDCF (0.1) = 0.206

SVM:

minDCF (0.1) = 0.265 - actDCF (0.1) = 0.408

Logistic Regression:

minDCF (0.1) = 0.350 - actDCF (0.1) = 0.361

A graph with lines and numbers

Description automatically generated

A graph with blue and orange lines

Description automatically generated

A graph with blue and orange lines

Description automatically generated

My final choice is more effective than any individual model but is very close to the gaussian mixture model with 16 components.

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

The fusion model's Bayes error plot shows it has the lowest DCFs across a wide range of prior log-odds.

Individual models benefit from the calibration but still show higher DCFs compared to the fusion.

The fusion model is well calibrated .

Calibration strategy was effective for the SVM with RBF kernel as we can see it is not well calibrated even for our application comparet to logistic regression and GMM .

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

I am going to consider The Gaussian Mixture Model Full Covariance Matrix with 16 components

As it is the best performing among the three methods I employed and it is almost as good as the fusion model which I crowned as the best model .   
A graph with red and blue lines

Description automatically generatedA graph with red and blue lines

Description automatically generated

As we can see for the GMM full covariance matrix with full covariance with 2\*\*4 components

We have a low DCF value and is better calibrated than GMM diagonal covariance with 4 or 8 components.

GMM (in the training set )

The GMM with full covariance matrix performs best with 16 components :  
minDCF = 0.163

actDCF = 0.175

The GMM with diagonal covariance matrix performs best in two points :

8 components:

minDCF = 0.146

actDCF = 1.81

4 components:

minDCF = 0.148

actDCF = 0.168

better calibrated.

GMM with diagonal covariance matrix with 4, 8 components performs better in terms of minDCF but I didn’t choose them because they are more poorly calibrated but maybe choosing them and then calibrating would have given better results.

Evaluating GMM with diagonal covariance matrix and 8 components :  
GMM: minDCF (0.1) = 0.204

I think my choice for the GMM with 16 components and full covariance matrix was optimal.

The fusion model performs slightly better but GMM is less complex to train and performs reasonably good.

In the training set K FOLD validation:   
The GMM with full covariance matrix performs best with 16 components :  
minDCF = 0.163

actDCF = 0.175

In the evaluation set :   
GMM with full covariance matrix and 16 component:

minDCF (0.1) = 0.202 - actDCF (0.1) = 0.206