
REPORT MACHINE LEARNING AND PATTERN RECOGNITION

Fingerprint spoofing detection

Alessia Intini
s309895
Politecnico di Torino
2023/24

MLPR Project:
Fingerprint spoofing detection

Contents

1	Dataset Analysis	3
1.1	Training and evaluation sets	3
1.2	Features analysis	3
1.3	Features correlation	4
2	Dimensionality reduction	5
2.1	PCA	5
2.2	LDA	7
2.3	Classification	8
3	Multivariate Gaussian Density	8
4	Model evaluation for classification	9
5	Classification model analysis	10
5.1	Gaussian Models	10
5.1.1	MVG Gaussian Classifier	11
5.1.2	Naive Bayes Classifier	11
5.1.3	Tied Bayes Classifier	11
5.1.4	Gaussian Models Comparison	12
5.2	Logistic Regression Classifier	14
5.2.1	Quadratic Logistic Regression(QLR)	14
5.3	SVM Classifier	14
5.3.1	Linear SVM	14
5.3.2	Kernel SVM	14
5.4	GMM Classifier	14
6	Score Calibration	14
6.1	Calibration Analysis on Selected Models	14
6.2	Calibrating Scores for Selected Models	14
7	Experimental Results	14
7.1	Calibration on evaluation score	14
7.2	Considerations	14
8	Conclusions	14

MLPR Project: Fingerprint spoofing detection

The goal of project is to perform a binary classification on fingerprint spoofing detection, that is 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 data is 6-dimensional.

1 Dataset Analysis

1.1 Training and evaluation sets

The datasets provided contain 6000 samples; the first 6 values in each row represent the features, while the last value is the label. Specifically they are:

- Training Set: 2990 samples belonging to the Fake class (label 0) and 3010 samples belonging to the Genuine class (label 1)
- Evaluation Set: 3010 samples belonging to the Fake class (label 0) and 2990 samples belonging to the Genuine class (label 1)

We will use the Training Set to perform all the analysis, during this phase the dataset is divided to use about 60% of it as the training dataset and the remaining 40% for validation. After this step, the most promising models were chosen and the evaluation dataset was used to evaluate them and make the final considerations.

1.2 Features analysis

We can start to analyze all features, here is the graph of each feature, through these histograms it is shown how each feature is distributed:

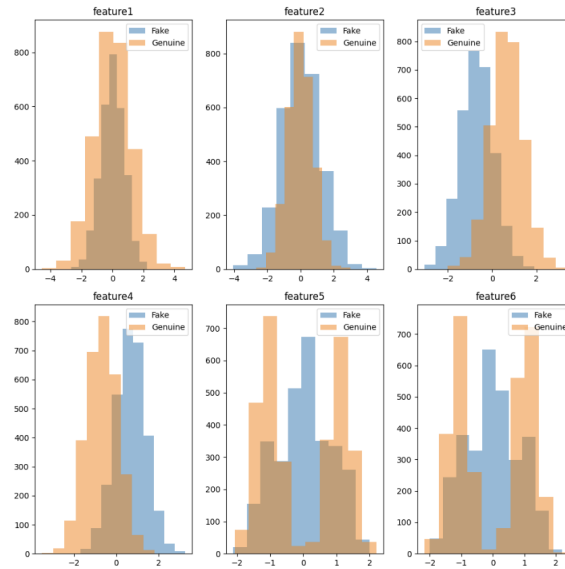


Figure 1: All features

By plotting a histogram for each feature, separately for the Fake and Genuine classes, it is possible

MLPR Project: Fingerprint spoofing detection

to show whether the samples follow a Gaussian distribution and to what extent. Thus, we can say that some features follow a Gaussian distribution. Furthermore, it can be seen that features 4 and 5 may be the most discriminating features based on their ability to separate the data of the two classes.

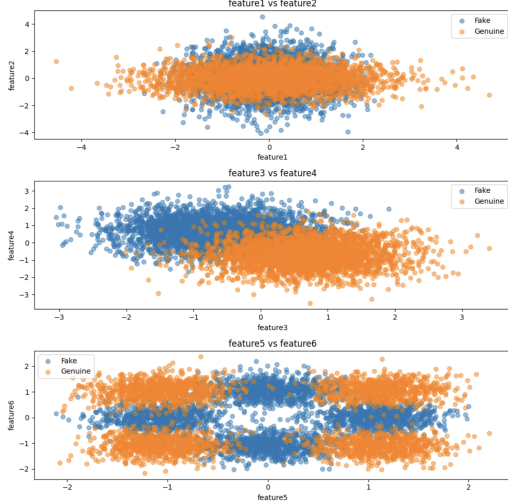


Figure 2: Distribution of feature pairs

Feature	Mean	Variance
1	0.00170711	1.00134304
2	0.00503903	0.9983527
3	-0.00560753	1.0024818
4	0.00109537	0.99029389
5	-0.00700025	1.00119747
6	0.00910515	0.99722374

Figure 3: Mean and variance for each features

After making these calculations we can conclude that the data are already centered since all features have mean close to zero.

Analyzing the pairwise averages and variances of the features, we can make some observations:

- Feature 1 and Feature 2: These two features have averages very close to zero, indicating that their values are similarly distributed around zero. However, the variance of Feature 2 is slightly lower than that of Feature 1, indicating that the values of Feature 2 are slightly less dispersed than those of Feature 1.
- Feature 3 and Feature 4: Feature 3 has a negative mean, while Feature 4 has a positive mean. This might indicate that the values of Feature 3 tend to be lower than those of Feature 4. Also, the variance of Feature 3 is slightly higher than that of Feature 4, indicating that the values of Feature 3 are more dispersed than those of Feature 4.
- Feature 5 and Feature 6: Feature 5 has a negative mean, while Feature 6 has a positive mean. This might indicate that the values of Feature 5 tend to be lower than those of Feature 6. Also, the variance of Feature 5 is slightly higher than that of Feature 6, indicating that the values of Feature 5 are more dispersed than those of Feature 6.

1.3 Features correlation

We can show how correlated the features are by using a heatmap that shows a darker color within the cells for which there is a high correlation between feature i and feature j . The correlation

MLPR Project: Fingerprint spoofing detection

between feature x and feature y is calculated using Pearson's coefficient:

$$Corr_{i,j} = \frac{Cov(i,j)}{\sqrt{Var(i)}\sqrt{Var(j)}} \quad (1)$$

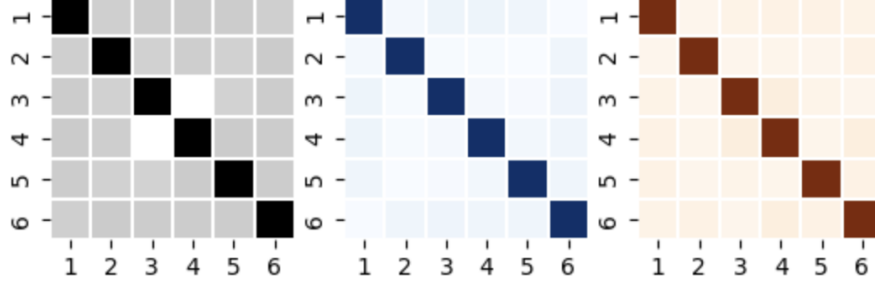


Figure 4: Correlation features: gray scale indicates the correlation between whole dataset, blue is for Fake class and orange for Genuine class

Most of the features do not appear to be particularly correlated; in fact, most of the values in the correlation matrix are close to zero.

Since the features do not appear to be strongly correlated since precisely the values are close to zero. This suggests to us that each feature carries unique information that could be useful for classification. Therefore, it is most likely that in PCA we will use a value of m as high as possible, possibly equal to the number of features in this way it will be possible to preserve as much of the information in the data as possible.

While in the case of Naive Bayes model and Full Covariance Gaussian Model could have similar performance because the features appear to be not strongly correlated. While in the case of the Tied model, which shares the covariance matrix between the classes, it may not have very good performance since the features are not strongly correlated.

2 Dimensionality reduction

Dimensionality reduction is useful in this context to compute a mapping from an n -dimensional feature space to an m -dimensional space; it is applied because it can compress information, remove noise and simplify classification.

2.1 PCA

PCA is an unsupervised dimensionality reduction technique that, given a centered dataset $X = \{x_1, \dots, x_k\}$, it aims to find the subspace of R^n that allows to preserve most of the information, i.e. the directions with the highest variance. We start by calculating the covariance matrix:

$$C = \frac{1}{K} \sum_i (x_i - \bar{x})(x_i - \bar{x})^T \quad (2)$$

MLPR Project: Fingerprint spoofing detection

After that we can compute the eigen-decomposition of $\mathbf{C} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$, where \mathbf{U} is the matrix of eigenvectors and $\mathbf{\Sigma}$ is the diagonal matrix of eigenvalues. Now we can project the data into the new subspace, it is spanned by the m columns of \mathbf{U} corresponding to the m values of the highest eigenvalues.

$$\mathbf{y}_i = \mathbf{P}^T(x_i - \bar{x}) \quad (3)$$

\mathbf{P} is the matrix corresponding to the m columns of \mathbf{U} associated with the m highest eigenvalues of \mathbf{C} . At this point to choose which is the best value of m we need to check how much total variance in the data we are able to retain using different values of m . To do this evaluation, it was decided to represent on a graph the variance of the data as m increases, i.e., we exploited that each eigenvalue corresponds to the variance along the corresponding axis, and the eigenvalues are the elements of the diagonal of the matrix $\mathbf{\Sigma}$. The percentage will be calculated as the ratio of the sum of the first m eigenvalues to the sum of all eigenvalues.

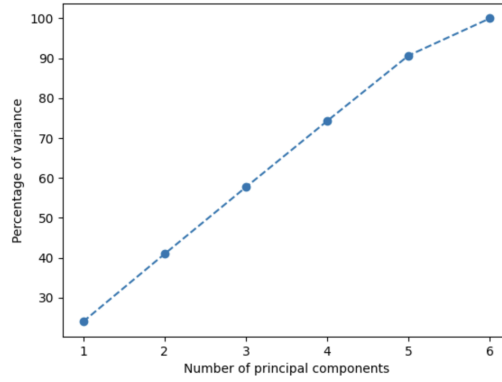


Figure 5: Variance of data for each m values

It is clear from Figure 5 that values of $m < 5$ rapidly decrease the amount of variance retained in the data. Removing a feature is probably the best choice, since this way the percentage of variance remains fairly high.

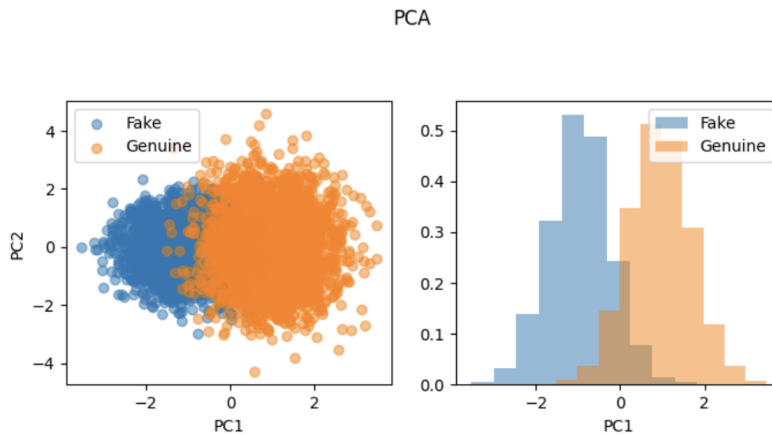


Figure 6: PCA for the two first components

MLPR Project: Fingerprint spoofing detection

After the analysis done to choose which m is best to use, we can represent the first two principal components found by calculating the PCA with $m=6$, visualizing these components can help to understand which features contribute most to the variance in the data. It can actually be observed that there is better separation between classes by applying PCA and choosing an m that has good variance of the data.

2.2 LDA

LDA is a supervised dimensionality reduction technique that aims to find the subspace that maximizes the separation between classes.

To compute the transformation matrix W we need to compute between-class and within-class covariance matrices.

$$\mathbf{S}_b = \frac{1}{N} \sum_{c=1}^K n_c (\mu_c - \mu)(\mu_c - \mu)^T \quad (4)$$

$$\mathbf{S}_w = \frac{1}{N} \sum_{c=1}^K \sum_{i=1}^{n_c} (x_{c,i} - \mu_i)(x_{c,i} - \mu_i)^T \quad (5)$$

where μ_c is the mean of the c -th class, μ is the mean of all the data, and n_c is the number of samples in the c -th class.

The LDA directions can be computed solving the generalized eigenvalue problem $S_b w = \lambda S_w w$, usually to solve the generalized eigenvalue problem we transform the S_w matrix into an identity matrix and then solve the eigenvalue problem for the other S_b matrix. This process allows us to find the discriminant vectors w that maximize the separation between the classes.

In LDA, the maximum number of discriminant directions or dimensions that can be obtained is always equal to $C-1$ where C is the number of classes. This is because the goal of the LDA is to find the directions that maximize the separation between the classes. In our case having only two classes the LDA can find at most one direction that maximizes the separation between these two classes.

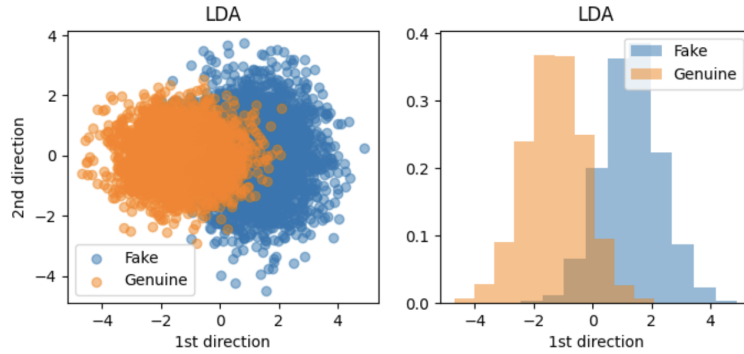


Figure 7: LDA for first direction

Looking at the results of LDA, in Figure 7, we can see that it cannot find an optimal direction that gives us a better separation of classes than the results shown previously.

2.3 Classification

Subsequent to applying dimensionality reduction techniques we used LDA to perform classification, in fact after finding the hyperplane on the training data and projecting the validation data into the new space we used different thresholds to perform classification. After applying dimensionality reduction techniques, we used LDA to perform classification; in fact, after finding the hyperplane on the training data and projecting the validation data into the new space these new data values can be used as scores by comparing them with different thresholds to perform classification, if the value exceeds the threshold it is associated with the label 1 while if it is lower 0. Two different methods were used to calculate the threshold value:

- **First Threshold:** The first method calculates the average of the LDA values for the two classes and uses the average of these two averages as the threshold. This method is simpler, but does not take into account the distribution of the data or the classification error rate. In fact, it may not provide an optimal threshold if the data are not well balanced or if the distributions of the two classes overlap significantly.
- **Second Threshold:** The second method uses a more sophisticated approach to determine the threshold. Instead of simply calculating the mean, a range of possible threshold values are examined and the one that minimizes the classification error rate is chosen. This method is more computationally heavy, but can lead to a more accurate threshold and a lower classification error rate.

Methods	Number of errors	Error rate (%)
LDA - First threshold	275	13.75
LDA - Second threshold	184	9.2
PCA(m=6)+LDA - First threshold	186	9.3
PCA(m=5)+LDA - First threshold	186	9.3
PCA(m=6)+LDA - Second threshold	184	9.2
PCA(m=5)+LDA - Second threshold	185	9.25

Table 1: LDA and PCA+LDA results with different threshold

As can be seen from the Table 1 there is not much real improvement by applying PCA as pre-processing before LDA and that the classification error turns out to be quite high so good classification is not performed using LDA or the combination of LDA with PCA.

3 Multivariate Gaussian Density

The multivariate Gaussian distribution is a generalization of the multidimensional Gaussian normal distribution. It is used to describe the probability of a vector of continuous random variables that can be correlated with each other.

The multivariate Gaussian distribution in d dimensions is characterized by a vector of averages μ and a covariance matrix Σ . The probability density function of the multivariate Gaussian distribution is given by:

$$\mathcal{N}(x|\mu, \Sigma) = \frac{1}{(2\pi)^{M/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right) \quad (6)$$

MLPR Project: Fingerprint spoofing detection

Very often to prevent numerical problems due to exponentiation of large numbers, it is usually recommended to work on the logarithm of the density:

$$\log \mathcal{N}(x|\mu, \Sigma) = -\frac{M}{2} \log(2\pi) - \frac{1}{2} \log |\Sigma| - \frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \quad (7)$$

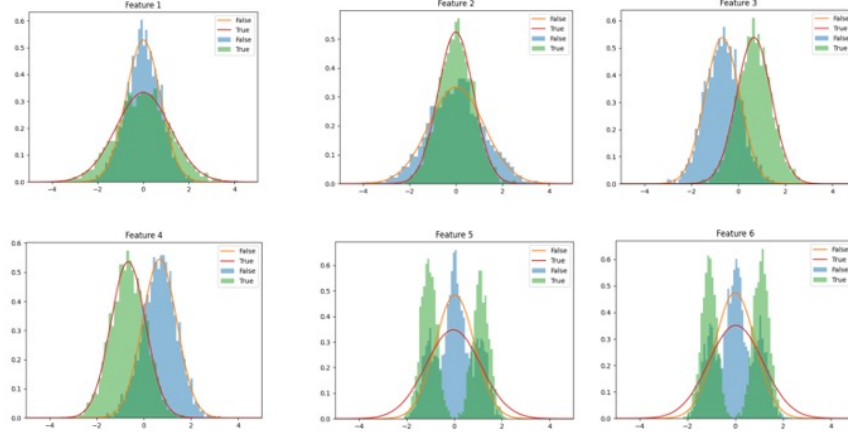


Figure 8: Gaussian density

The curve we can observe on the graph in Figure 8 represents the Gaussian probability density calculated from the data. If the data actually follow a Gaussian distribution, then the curve should fit the histogram of the data well. When this happens we can say that the Gaussian model is a good fit for the data while if the Gaussian curve does not fit the histogram well, then the Gaussian model may not be the most appropriate model for that data.

We can observe that feature 1, feature 2, feature 3 and feature 4 fit the model well and follow a Gaussian distribution.

4 Model evaluation for classification

To understand which model is the most promising and gives us the best results, we divided the dataset as mentioned above into a part for training and a part for validation. This way we can test our model on data other than the training data.

We considered different types of applications i.e. various values of $(\tilde{\pi}, C_{fn}, C_{fp})$. The target application chosen to be balanced will be:

$$(\tilde{\pi}, C_{fp}, C_{fn}) = (0.1, 1, 1) \quad (8)$$

The objective of the following analysis is to choose the model with the best performance. Performance will be measured in terms of normalized minimum detection costs. This technique would allow us to discriminate the best performing model for our target application by providing an assessment of how well the model can detect the class to which it belongs.

MLPR Project:
Fingerprint spoofing detection

It is defined as follows:

$$DCF_u = \sum_{k=0}^{K-1} \frac{\pi_k}{N_k} \sum_{i|c_i=k} C(c_i^*|k) = \pi_T C_{fn} P_{fn} + (1 - \pi_T) C_{fp} P_{fp} \quad (9)$$

where $P_{fn} = \frac{FN}{FN+FP}$ and $P_{fp} = \frac{FP}{FP+TP}$.

From the Equation 4 it is possible to find **minDCF** and **actDCF**, and different thresholds are used to calculate these. In fact in the case of minDCF one uses as a threshold the one that minimizes DCF and there are several methods to be able to find it. Whereas for actDCF you use as a threshold:

$$t' = -\log \frac{\tilde{\pi}}{1 - \tilde{\pi}} \quad (10)$$

5 Classification model analysis

To perform the classification obviously we had to divide the training dataset into a part for training and a part for validation.

5.1 Gaussian Models

The first class of models we will analyze is that of Gaussian generative models. A simple model is to assume that our data, given a class, can be described by a Gaussian distribution:

$$f_{x|C}(x|c) = \mathcal{N}(x|\mu_c, \Sigma_c) \quad (11)$$

Since this is a binary classification task, we will assign a probability score to each sample in terms of the log-posterior class likelihood ratio:

$$llr(x_t) = \log r(x_t) = \log \frac{P(C = h_1|x_t)}{P(C = h_0|x_t)} \quad (12)$$

This expression can be expanded writing:

$$llr(x_t) = \log \frac{f_{X|C}(x_t|C = h_1)}{f_{X|C}(x_t|C = h_0)} + \frac{\pi}{1 - \pi} \quad (13)$$

where $f_{X|C}(x_t|c) = \mathcal{N}(x|\mu_c, \Sigma_c)$

It is necessary to find the parameters θ , μ_c and Σ_c ; this can be done by maximizing the log-likelihood.

Parameter estimation is part of the training phase and is therefore performed on the training part of the dataset. While the estimation phase is to calculate the likelihood ratio for each sample and compare it to a threshold, which will be set at 0(th=0) in this step. After doing this, the error rate can be calculated to evaluate the classification. This analysis has been done on different Gaussian models, the difference between these models is the way the parameters (μ_c, Σ_c) are calculated, more specifically Σ_c .

MLPR Project: Fingerprint spoofing detection

5.1.1 MVG Gaussian Classifier

The first model we will analyze is the multivariate Gaussian classifier; in this model the covariance matrix is assumed to be different for each class. The parameters μ_c and Σ_c are estimated by maximizing the log-likelihood of the training data. Thus they can be obtained as:

$$\mu_c^* = \frac{1}{N_c} \sum_{i|c_i=C} x_i \quad \sigma_c^* = \frac{1}{N_c} \sum_{i|c_i=C} (x_i - \mu_c)(x_i - \mu_c)^T \quad (14)$$

5.1.2 Naive Bayes Classifier

The Naive Bayes hypothesis simplifies the MVG full covariance model. In fact, if we know that the different components of our characteristics are approximately independent, we can simplify the estimation by assuming that the distribution of $x|C$ can be factorized on its components.

The parameters then become:

$$\mu_{c,[j]}^* = \frac{1}{N_c} \sum_{i|c_i=C} x_{i,[j]} \quad \sigma_{c,[j]}^* = \frac{1}{N_c} \sum_{i|c_i=C} (x_{i,[j]} - \mu_{c,[j]})^2 \quad (15)$$

The Naïve Bayes classifier corresponds to the MVG full covariance classifier with a diagonal covariance matrix.

5.1.3 Tied Bayes Classifier

Another common Gaussian Model is Tied Model, it assumes that the covariance matrix is the same for all classes, but each class still has its own mean. It assumes that:

$$f_{X|C}(x|c) = \mathcal{N}(x|\mu_c, \Sigma) \quad (16)$$

While the parameters are estimated as:

$$\mu_c^* = \frac{1}{N_c} \sum_{i|c_i=C} x_i \quad \Sigma^* = \frac{1}{N} \sum_c \sum_{i|c_i=C} (x_i - \mu_c)(x_i - \mu_c)^T \quad (17)$$

where $N = \sum_{c=1}^k N_c$, that is the number of all samples.

This model is strongly related to LDA, if considering the binary log-likelihood ratio of the tied model we obtain a linear decision function:

$$llr(x_t) = \log \frac{f_{X|C}(x|h_1)}{f_{X|C}(x|h_0)} = x^T b + c \quad (18)$$

where b and c are functions of class means and covariance matrix. On the other hand, projecting over the LDA subspace is, up to a scaling factor k , given by:

$$w^t x = k x^T \Lambda (\mu_1 - \mu_0) \quad (19)$$

where $\Lambda(\mu_1 - \mu_0) = b$. The LDA assumption that all the classes have the same within class covariance matrix is related to the assumption done for the tied model.

MLPR Project:
Fingerprint spoofing detection

5.1.4 Gaussian Models Comparison

At this point placing the $P(C = 1) = P(C = 0) = 1/2$, classification was performed as can be seen in Table 2 using Gaussian models.

Features	Model	Error Rate (%)
no PCA		
1,2,3,4,5,6	Full Cov	7.00
1,2,3,4,5,6	Naïve Bayes	7.20
1,2,3,4,5,6	Tied Cov	9.30
1,2,3,4	Full Cov	7.95
1,2,3,4	Naïve Bayes	7.65
1,2,3,4	Tied Cov	9.50
1,2	Full Cov	36.50
1,2	Naïve Bayes	36.30
1,2	Tied	49.45
3,4	Full Cov	9.45
3,4	Naïve Bayes	9.45
3,4	Tied Cov	9.40
PCA m= 5		
1,2,3,4,5,6	Full Cov	7.10
1,2,3,4,5,6	Naïve Bayes	8.75
1,2,3,4,5,6	Tied Cov	9.30
PCA m= 6		
1,2,3,4,5,6	Full Cov	7.00
1,2,3,4,5,6	Naïve Bayes	8.90
1,2,3,4,5,6	Tied Cov	9.30

Table 2: Error Rates for Different Models and Features

Comparing the results with Table 1 , where LDA had been used for classification we can observe that for certain configurations there was an improvement. This could be due to the fact that this method is able to classify better and thus recognize the point to which class it belongs. Analyzing the data provided in Table 2 we can make some considerations about the features used for classification:

- **Features (1,2,3,4,5,6):** the classification error is relatively low for all models, this may suggest that all features provide useful information.
- **Features (1,2,3,4):** the classification error increases slightly, so it can be said that features 5 and 6 probably contain useful classification information.
- **Features (1,2),(3,4):** the classification error increases significantly in the case of features(1,2) because they do not contain particularly important information for classification. While for (3,4),the error remains similar to the others , this might indicate that these two features are particularly informative for classification.

Now we apply to estimate performance the metrics described in the previous section, we began by testing several applications to understand how changing parameters were reflected in perfor-

MLPR Project:
Fingerprint spoofing detection

mance. The prior value represents the a priori probability of the positive class, so if it is higher the value is expected to be more common, while a higher value of C_{fn} indicates that the cost of misclassifying a positive point as negative is higher, and vice versa for C_{fp} .

Model	Full Cov	Naïve Bayes	Tied Cov
Application(π_T, C_{fn}, C_{fp}) : (0.50, 1, 1)			
actDCF	0.141913	0.174987	0.186044
minDCF	0.133144	0.173691	0.181163
Application(π_T, C_{fn}, C_{fp}) : (0.90, 1, 1)			
actDCF	0.398041	0.466014	0.462558
minDCF	0.351238	0.434044	0.445132
Application(π_T, C_{fn}, C_{fp}) : (0.10, 1, 1)			
actDCF	0.304147	0.393017	0.405066
minDCF	0.273825	0.354471	0.364823
Application(π_T, C_{fn}, C_{fp}) : (0.50, 1, 9)			
actDCF	0.304147	0.393017	0.405066
minDCF	0.273825	0.354471	0.364823
Application(π_T, C_{fn}, C_{fp}) : (0.50, 9, 1)			
actDCF	0.398041	0.466014	0.462558
minDCF	0.351238	0.434044	0.445132

Table 3: minDCF and actDCF for Different Models and configurations

So it can be seen from the Table 3 that:

- When the value of Prior increases (from 0.50 to 0.90), a worsening of the models can be seen. This is due to the fact that the model is penalizing false negatives more heavily. This also happens when Prior decreases and therefore you are penalizing false positives more heavily.
- When Cfp increases to 9.00 , the actDCF and minDCF of all models do not change compared to the case when Cfp is 1.00. This suggests that increasing the cost of false positives does not have a significant impact on model performance. While if Cfn increases to 9.00 , the DCFAc and minDCF of all models increase compared to the case where Cfn is 1.00, from this we can derive that the increase in the cost of false negatives worsens the performance of the models.

We now focus on the three applications that have different effective priors (0.1, 0.5, 0.9) and with costs of errors equal to 1.00.

5.2 Logistic Regression Classifier

5.2.1 Quadratic Logistic Regression(QLR)

5.3 SVM Classifier

5.3.1 Linear SVM

5.3.2 Kernel SVM

5.4 GMM Classifier

6 Score Calibration

6.1 Calibration Analysis on Selected Models

6.2 Calibrating Scores for Selected Models

7 Experimental Results

7.1 Calibration on evaluation score

7.2 Considerations

8 Conclusions