Report for the Machine Learning & Pattern Recognition Project

Eduard Antonovic Occhipinti
947847

Introduction

The task consists of a binary classification problem, the goal is to perform fingerprint spoofing detection (i.e. to distinguish between real and fake fingerprints). The dataset consists of 6 features. In this first part, we will analyze some statistics of the dataset and the correlation between the features.

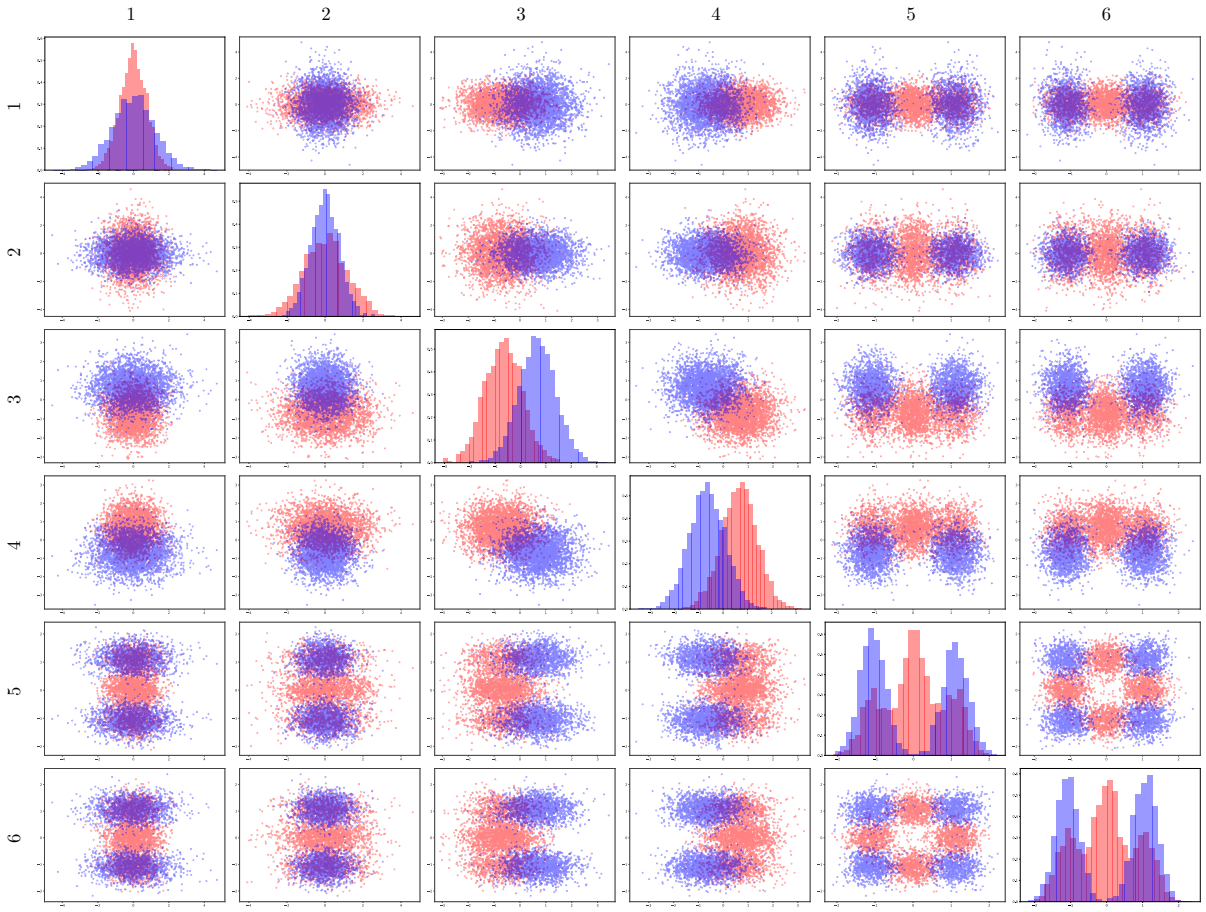

Table 1: Summary of the dataset features plotted against each other, number corresponds to the feature number

I - Features Compared

1. Features 1 and 2

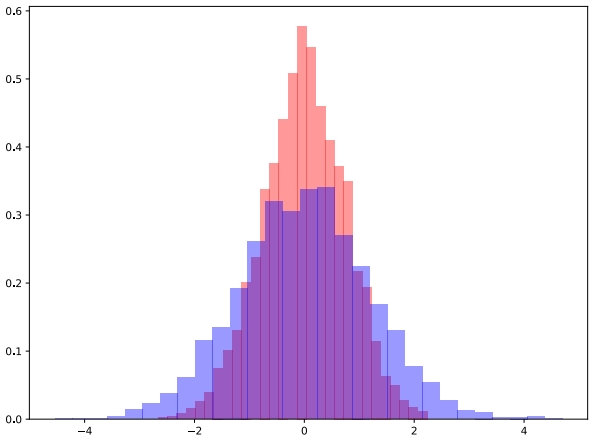

Figure 1: Feature 1

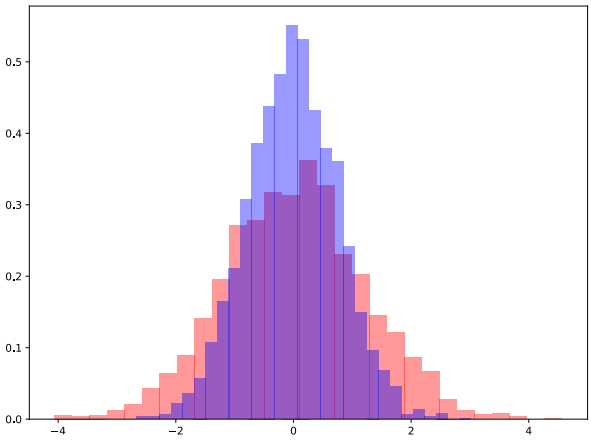

Figure 2: Feature 2

When looking at the first feature we can observe that the classes overlap almost completely. The `Genuine` label has a higher variance than the `Fake` class but the mean is similar. Both classes exhibit one mode in the histogram but the `Fake` class has a higher peak.

Looking at the second feature we can notice the opposite behavior. The `Fake` class has a higher variance than the `Genuine` class but the mean is similar. Both classes exhibit one mode in the histogram but the `Genuine` class has a higher peak. Again, the classes overlap almost completely.

---

2. Features 3 and 4

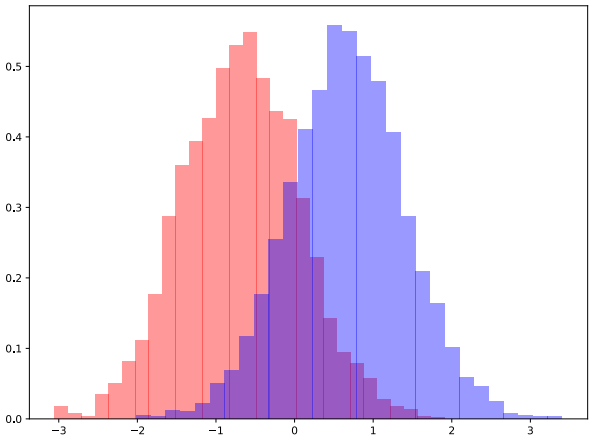

Figure 3: Feature 3

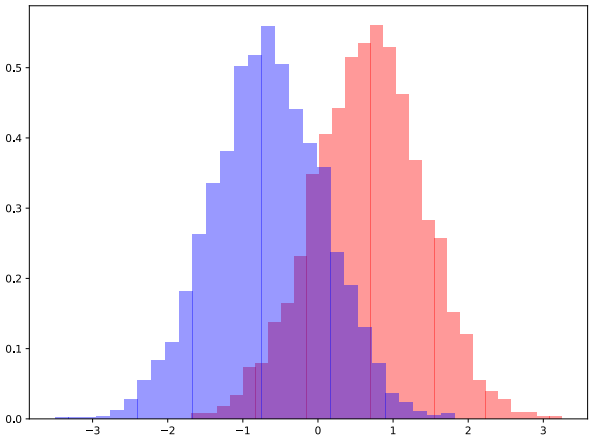

Figure 4: Feature 4

Looking at the plot for the third class we can notice that the two features are much more distinct, they overlap slightly in 0. The `Genuine` class has a peak in -1 while the `Fake` class has a peak in 1.

They both have similar mean and variance. One mode for each class is evident from the histogram.

The fourth feature shows similar characteristics to the third feature.

3. Features 5 and 6

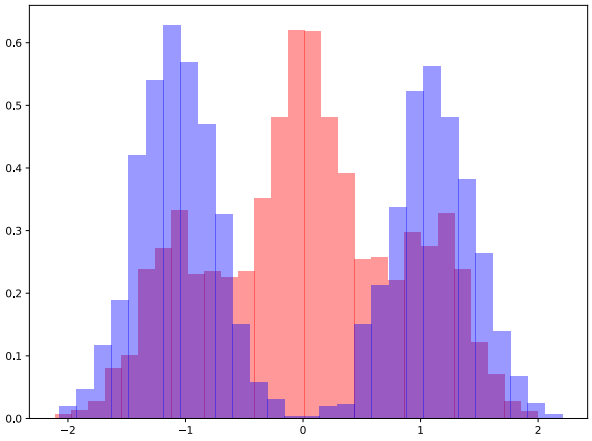

Figure 5: Feature 5

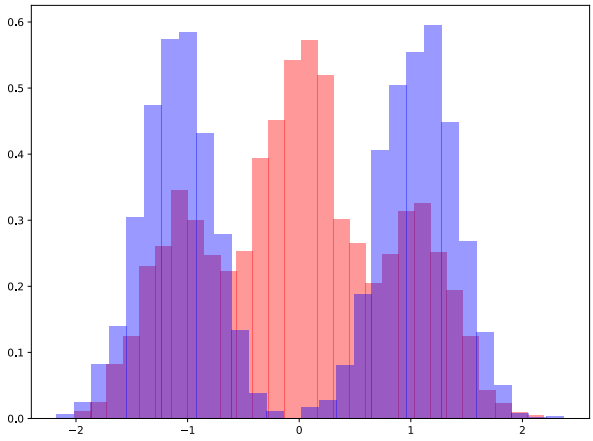

Figure 6: Feature 6

The fifth feature also shows a good distinction between the two classes with an overlap at the edges of the `Fake` class distribution. They exhibit similar variance but with a lower mean for the `Genuine` class. The `Fake` class peaks in 0 while the `Genuine` has two modes and peaks in -1 and 1.

The last feature shows similar characteristics to the fifth feature.

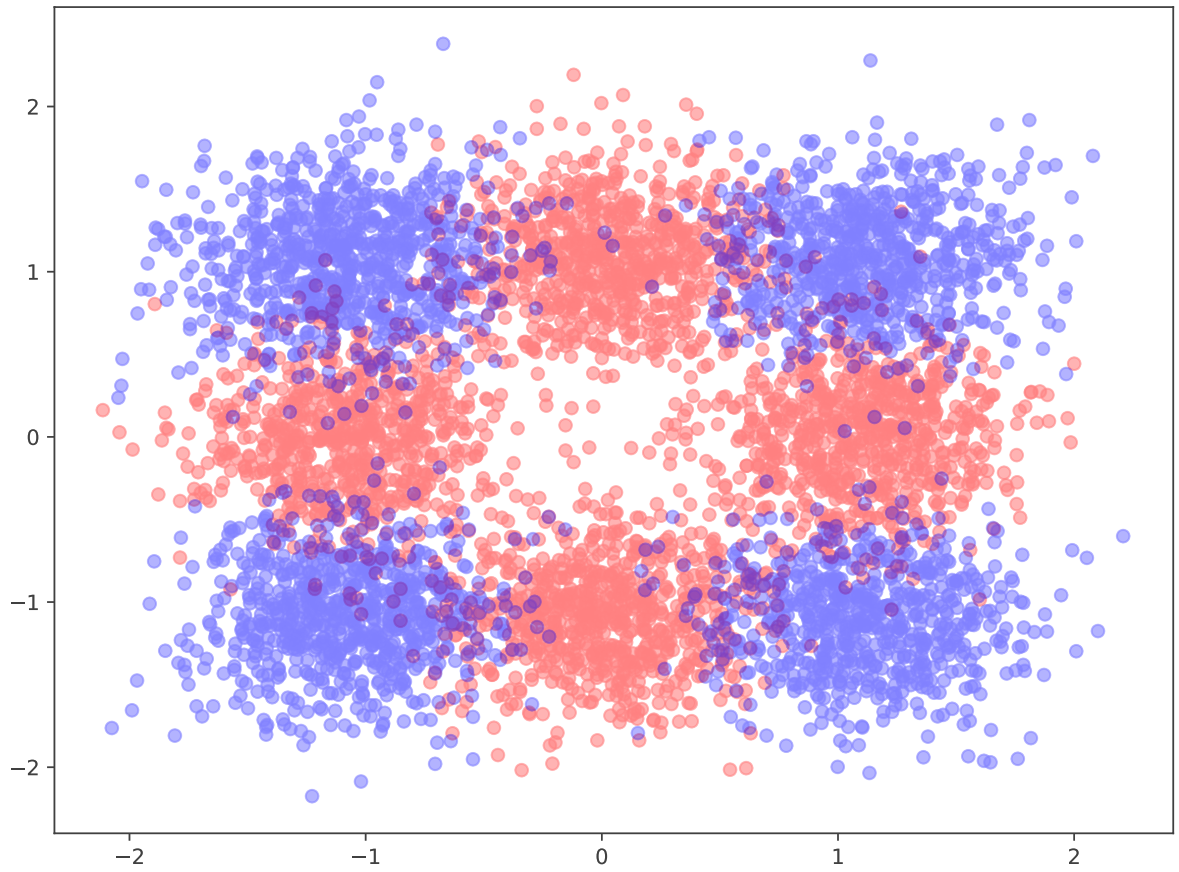

Figure 7: Features 5 and 6 Scatter Plot

Looking at the scatter plot we see that there are four distinct clusters for each of the labels, they overlap slightly at the edges of each cluster.

II - PCA & LDA

1. Principal Component Analysis

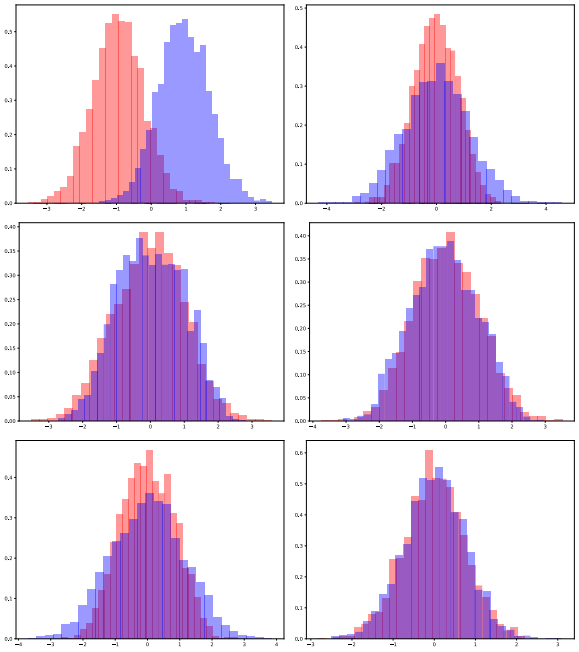

Figure 8: Principal components computed by PCA, ordered from top to bottom, right to left

Looking at the principal components of the dataset we can see that only one results in a clear separation between the two classes and it seems to separate the two classes better than any other feature taken individually.

2. Linear Discriminant Analysis

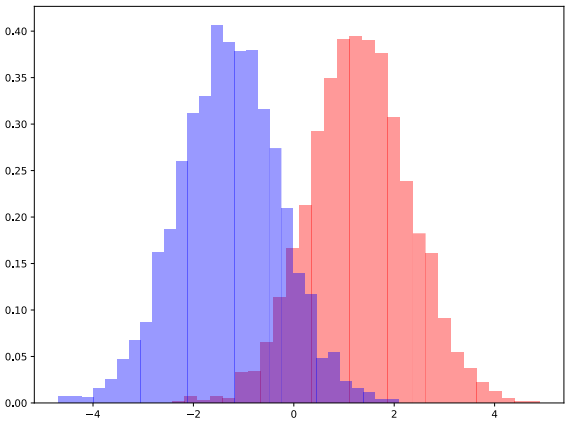

Figure 9: First LDA direction

We see that compared to the first principal the classes are mirrored but the separation is similar between the two methods.

2.a. Applying LDA as a classifier

We now try to apply LDA as a classifier, we start by splitting the dataset in a training and validation set, then we fit the model on the training and evaluation set, then we calculate the optimal threshold for the classifier and finally, we evaluate the model on the validation set.

```python
# Split the dataset into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.33, random_state=0
)

# Fit the LDA model
_, X_train_lda = lda(X_train, y_train, 1)
_, X_val_lda = lda(X_val, y_val, 1)

threshold = (
    X_train_lda[y_train == 0].mean() +
    X_train_lda[y_train == 1].mean()
) / 2.0

# Predict the validation data
y_pred = [
  0 if x ⩾ threshold else 1 for x in X_val_lda.T[0]
]

print(f"Threshold: {threshold:.2f}")
print(f"Error rate: {
  np.sum(y_val ≠ y_pred) / y_val.size * 100:.2f
}%")
```

```
Threshold: -0.02
Error rate: 9.60%
```

Empirically we can find that threshold `0.04` gives a slightly better error rate of `9.34%`.

2.b. Pre-processing the Data with PCA

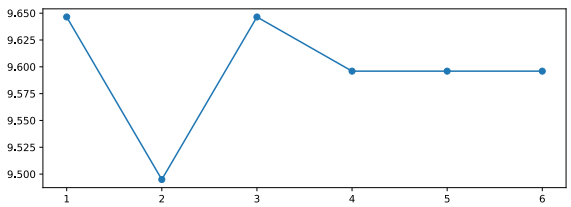

Figure 10: Error rates in percentage as a function of the number of LDA directions

As we can see from the graph, pre-processing the data with PCA proves useful in reducing the error rate of the classifier slightly, in particular when choosing a number `N` of components equal to 2.

III - ML Estimates & Probability Densities

1. Gaussian Models

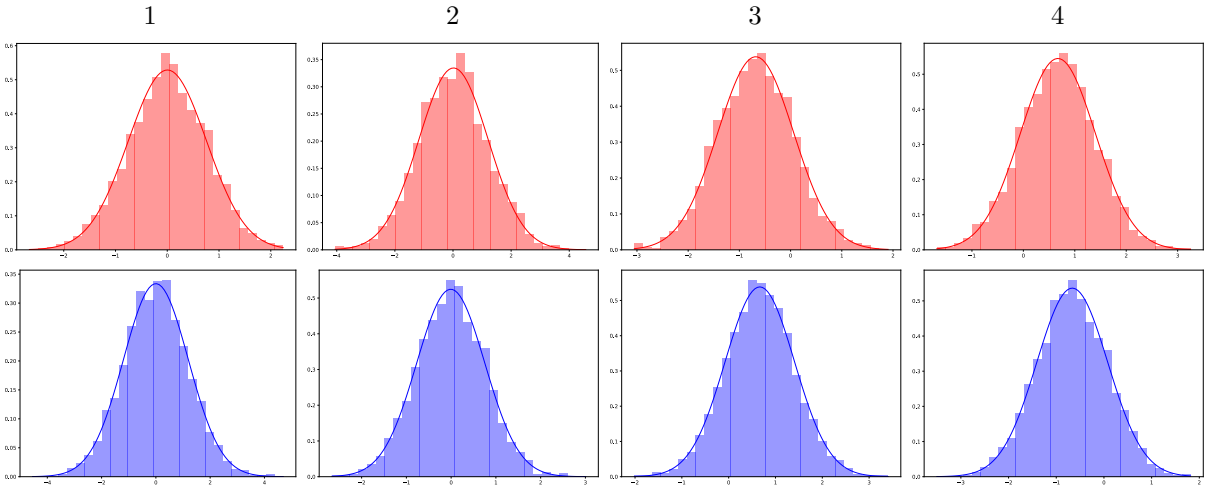

Table 2: Uni-variate Gaussian models with a good fit to the data

It is noticeable that features 1, 2, 3, and 4 fit well to a uni-variate Gaussian model, both for the `Genuine` and `Fake` classes.

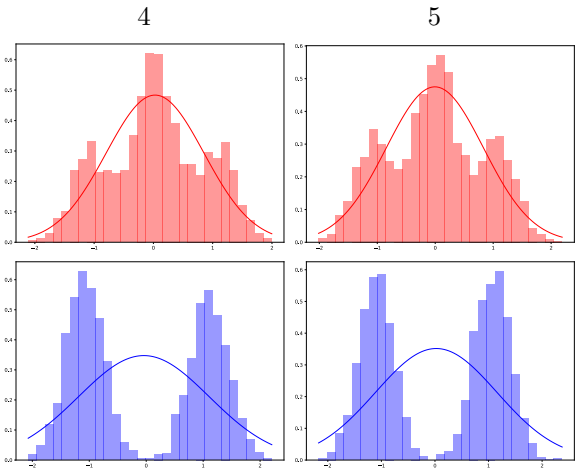

Table 3: Uni-variate Gaussian models with bad fit to the data

When looking at features 5 and 6, on the other hand, we can see that the uni-variate Gaussian model does not fit the data well. The `Genuine` class in particular has a bimodal distribution for both features so it results in a particularly bad fit.

2. Maximum Likelihood Estimates

$$\mu_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \sigma^2_{\text{ML}} = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu_{\text{ML}})^2$$

The ML estimates for the parameters of a `Univariate Gaussian` model correspond to the dataset mean and variance for each feature.

The following table summarizes the ML estimates for the dataset features.

μ_{ML}		
#	**Fake**	**Genuine**
1	$2.87744301 * 10^{-3}$	$5.44547838 * 10^{-4}$
2	$1.86931579 * 10^{-2}$	$-8.52437392 * 10^{-3}$
3	$-6.80940159 * 10^{-1}$	$6.65237846 * 10^{-1}$
4	$6.70836195 * 10^{-1}$	$-6.64195349 * 10^{-1}$
5	$2.79569669 * 10^{-2}$	$-4.17251858 * 10^{-2}$
6	$-5.82740035 * 10^{-3}$	$2.39384879 * 10^{-2}$

σ^2_{ML}		
#	**Fake**	**Genuine**
1	0.56958105	1.43023345
2	1.42086571	0.57827792
3	0.54997702	0.54890260
4	0.53604266	0.57827792
5	0.68007360	0.55334275
6	0.70503844	1.28702609

IV - Generative Models for Classification

	Multi Variate Gaussian	Tied Gaussian	Naive Bayes
Accuracy	92.47%	90.35%	92.37%
Error Rate	7.53%	9.65%	7.63%