Assignment 1
ECE 657A Winter 2022
Group 1

Jehanzeb Mirza
21115879

Mehmet Pelit
21113622

2024-02-16

Introduction

Our objective for this assignment is to apply methods discussed in our lectures for data preprocessing. We are provided two datasets *Dataset A* and *Dataset B* to analyze and process. There are three specific goals for this lab covered in the following sections.

1. Datacleaning and outlier detection
2. Feature extraction and dimensionality reduction
3. Invesitage the biases in a parameter estimator

Our invesitgation makes use of code written in Python and the source files, generated figures, and datasets are provided as supplementary files in addition to this report. We extensively use Numpy [1] and Panadas and Scypi for our functions.

1 Data Cleaning and Preprocessing

For this section we investigate *Dataset A*. *Dataset A* is a comma-seperate-value file (`DataA.csv`) which contains a time-series dataset from a set of motion sensors. The dataset is in time order with 19,000 samples and 81 seperate features. There are many missing values in the dataset denoted as NA (Not Available).

1.1 Trimming

Our first analysis was to simply count how many of the values were missing, which using a simple count yielded the following results:

Total	Valid	Missing	%Missing
1,539,000	1,414,947	124,053	8.1%

As a significant percentage, but not majority, of our data is missing. This suggests we may be able to apply simple techniques such as deleting invalid rows and can do without performing sophisticated interpolation techniques.

Next we graphically plot the missing values to determine any patterns or regularity which may be observed.

Three graphs created were: a bar graph representing the number of missing values for each feature, a line graph representing the number of missing values for each time, and finally a scatterplot of each missing value (feature, time).

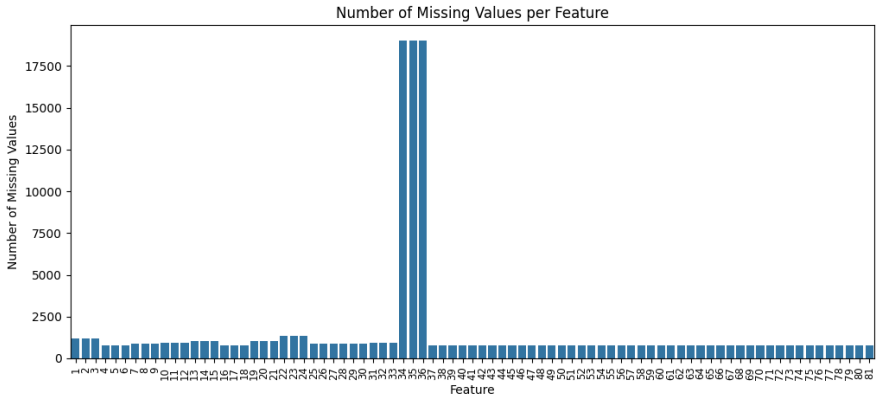

Figure 1: Number of Missing Values per Feature

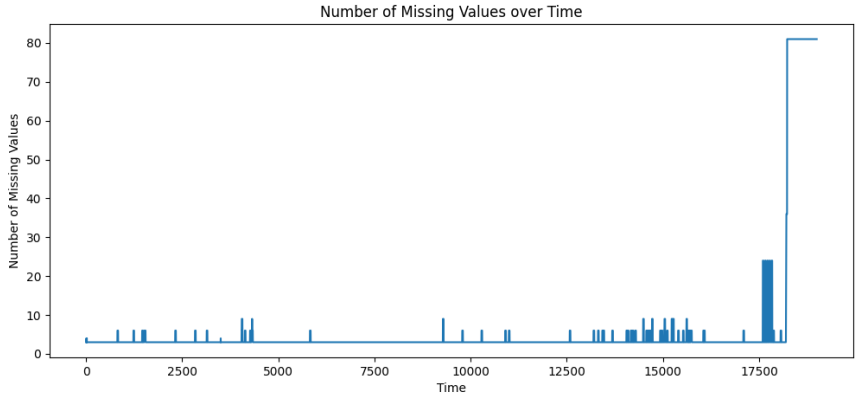

Figure 2: Number of Missing Values over Time

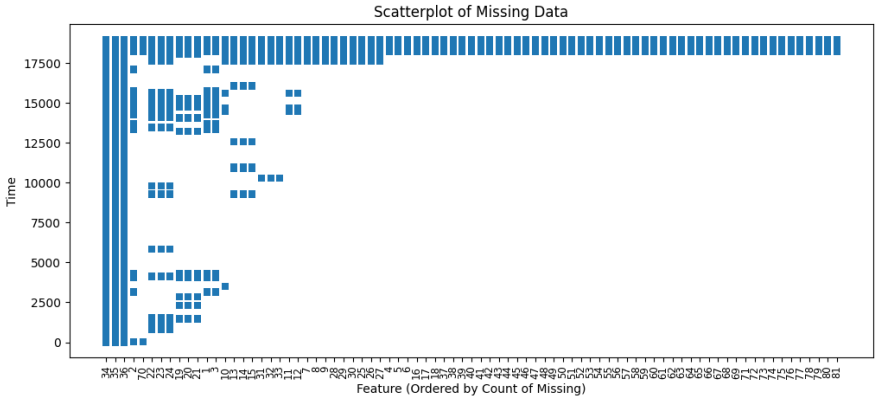

Figure 3: Scatterplot of All Missing Values

In Figure 1 we identify that most features have a relatively similar number of missing values while features 34, 35, and 36 are almost entirely missing. A similar observation in Figure 2 shows that values at time $t >= 18228$ are invalid for every feature.

Based on these observations a first step could be to trim the data to drop these features and time values. Doing so changes our distribution of invalid data to the following:

Total	Valid	Missing	%Missing
1,421,706	1,414,944	6762	0.48%

We replot Figure 3 after trimming to reveal any remaining patterns in our missing data.

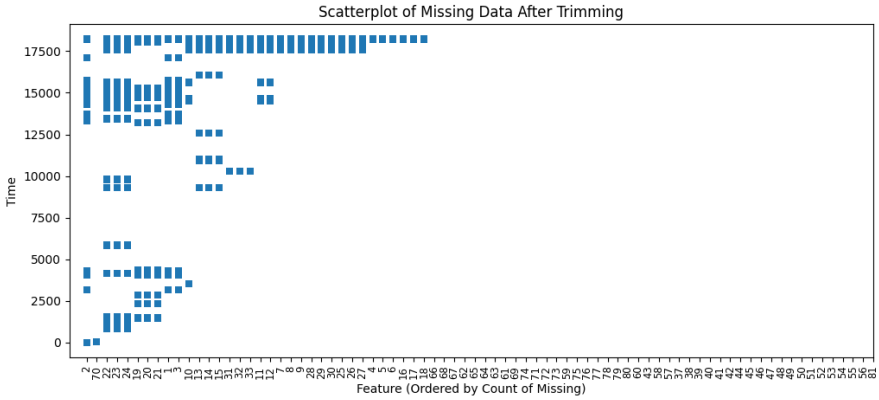

Figure 4: Scatterplot of All Missing Values After Data Trimming

Figure 4 shows that post trimming many of our features have no missing values and mainly features 2, 70, 22, 23, 24, 19, 20, 21, 1, and 3 are the ones which contribute most of the remaining missing values.

1.2 Outliers

In order to develop a method to remove outliers we first plot each of our trimmed features f_i over time and in a histogram to observe what patterns they hold.

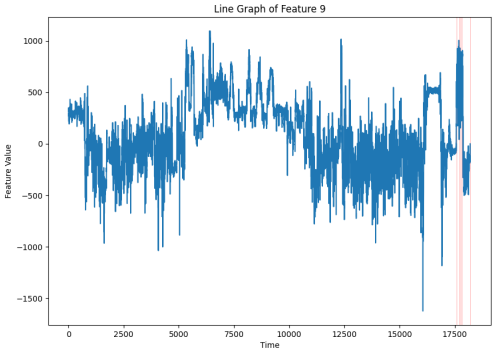

Figure 5: Feature 9 Time Series

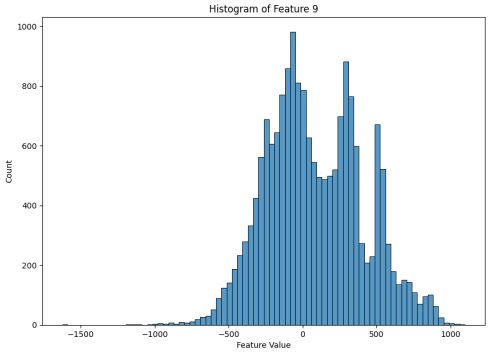

Figure 6: Feature 9 Histogram

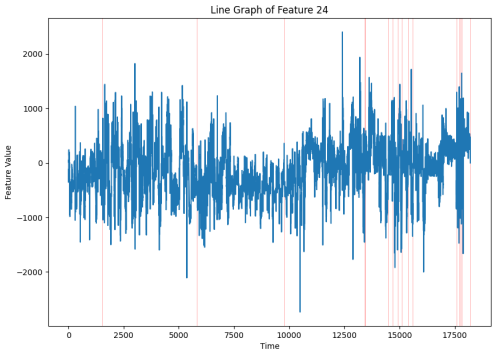

Figure 7: Feature 24 Time Series

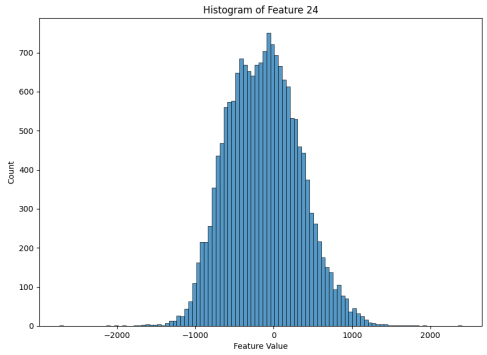

Figure 8: Feature 24 Histogram

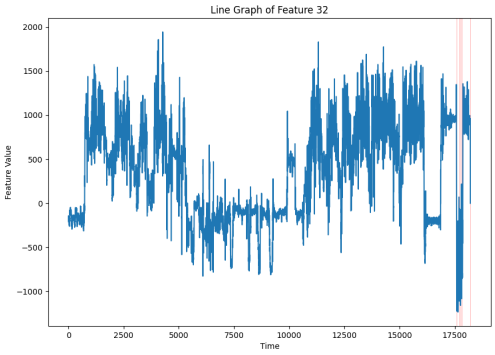

Figure 9: Feature 32 Time Series

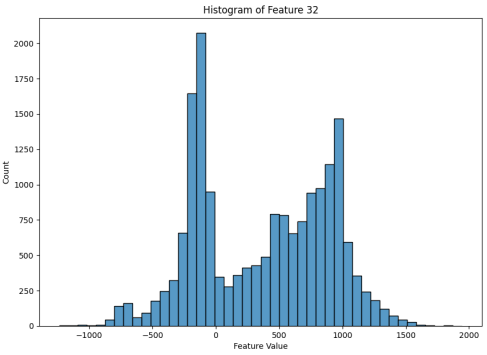

Figure 10: Feature 32 Histogram

We note that the features f_i are extremely noisy and contain many clear outliers when observered in time series. When observed as a histogram we can clearly see that the data is not normally distributed and the time dependance of the feature value is significant.

Our motivation for finding outliers is to modify Median Absolute Deviation in a way that still works with time series data. We choose to have a window of size $2k + 1$ around each value and replace each value with the median to develop a filtered version of each feature $f_i^{median(k)}$. We experimented with values of $k = 10, 20, 50, 75$ based on the rough width of the spikes seen in Figures.

$$f_i^{median(k)} = median\left\{f_{i-k}, f_{i-k+1}, \cdots, f_{i+k-1}, f_{i+k}\right\} \tag{1}$$

This replacement resulted in a median filtered feature which we then subtract from the original to produce a median difference feature $f_i^{median\_diff(k)}$.

$$f_i^{median\_diff(k)} = f_i - f_i^{median(k)} \tag{2}$$

We produce a histogram using the difference feature and discover that it is very close to normally distributed for all series.

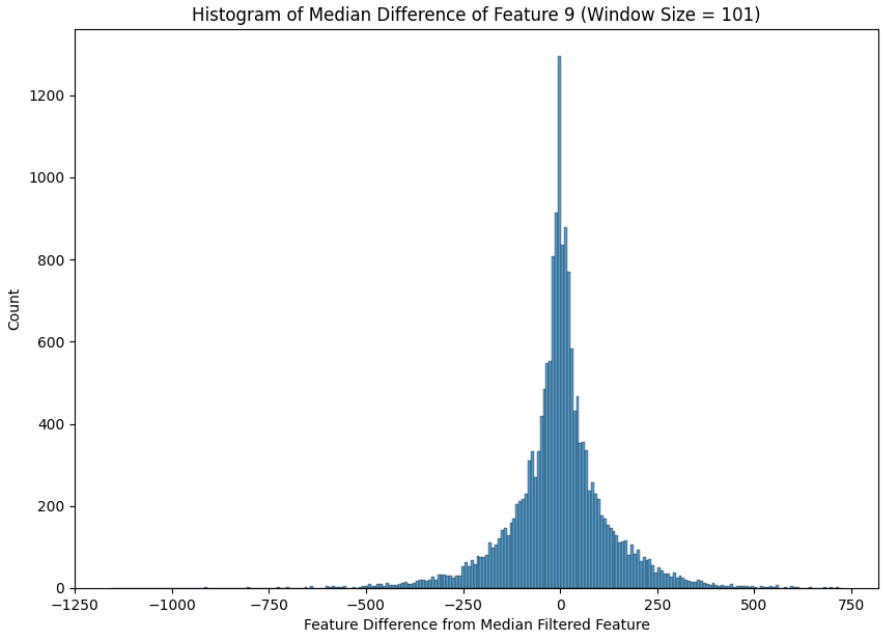

Figure 11: Feature 9 Histogram of Median Difference with k=50

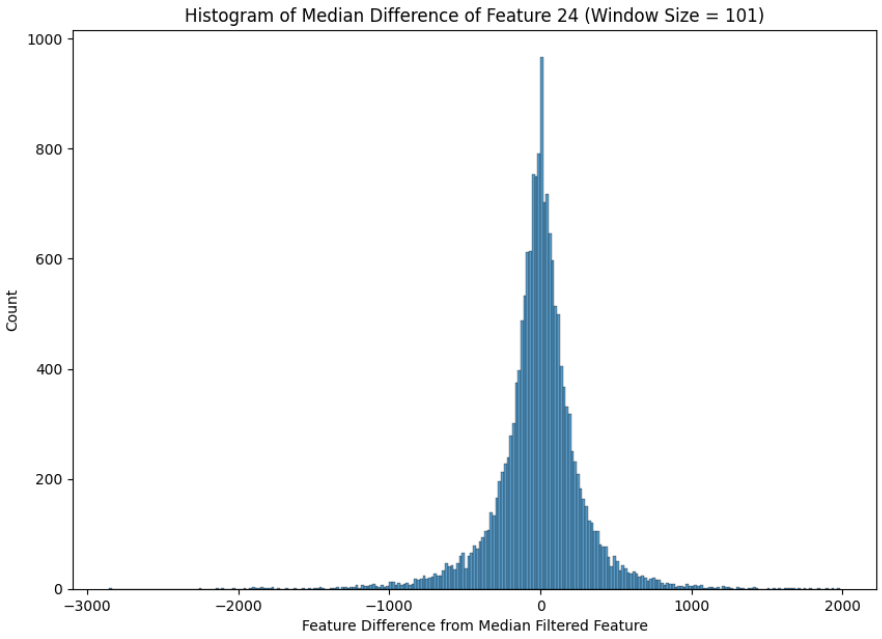

Figure 12: Feature 24 Histogram of Median Difference with k=50

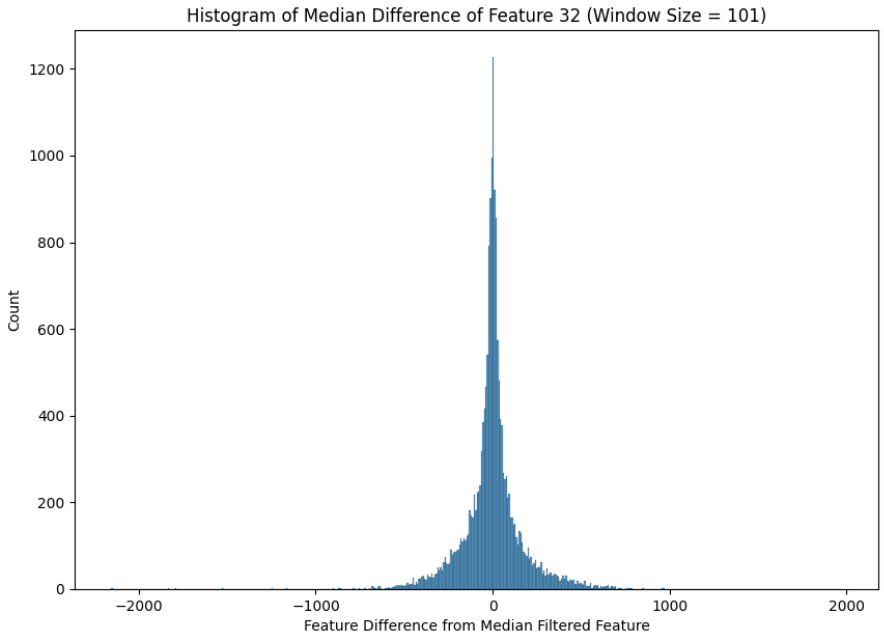

Figure 13: Feature 32 Histogram of Median Difference with k=50

We assume that the data in the histogram is normally distributed and use z-score normalization to give it a standard deviation of 1 and a mean of 0. Our outliers in the data are considered as those values which have an absolute z score of greater than 2.

We produce a bar chart for the number of outliers produced for each feature for every given value of k.

Empirically we observe that $k = 50$ corresponding to a window size of 101 seems to give good results so we chose it as our final window size. Our choice of replacement is to replace each outlier with its value from the median filtered feature f_i. The resulting line plots for some features are presented below.

As a final step to remove the final missing values from the dataset we use pandas ffill which replaces every missing value with the last valid value before it. This is a fairly simple fill function that seems to produce acceptable results.

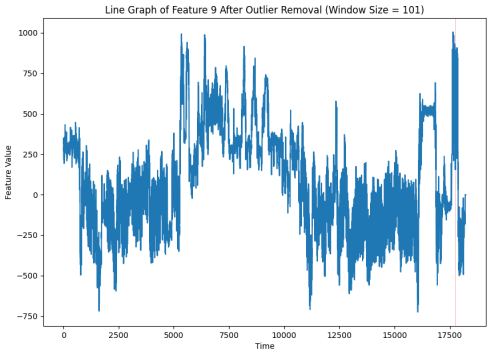

Figure 14: Feature 9 Processed Time Series

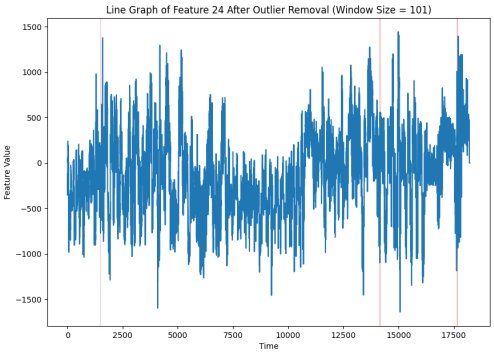

Figure 15: Feature 24 Processed Time Series

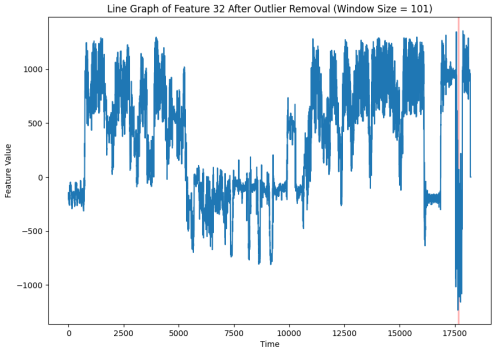

Figure 16: Feature Feature 32 Processed
Time Series

Figure 17: Feature 48 Processed Time Series

Lastly we explore z-score and min-max normalization for features 9 and 24. Here are the following histograms before and after we normalize them using the following equations.

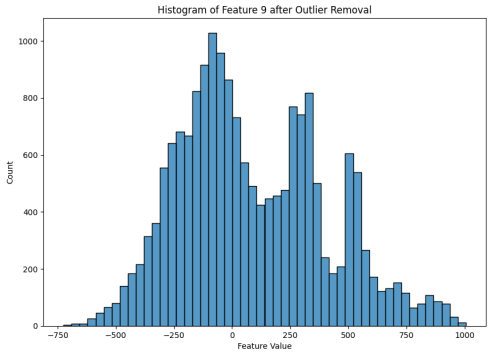

Figure 18: Feature 9
Before Normalization

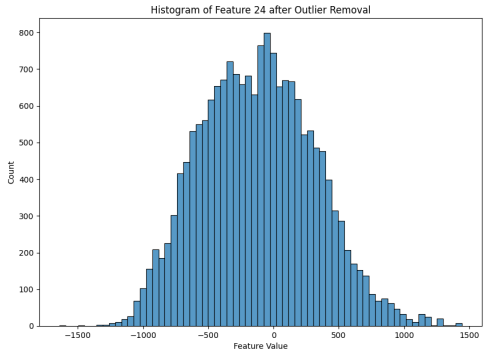

Figure 19: Feature 24
Before Normalization

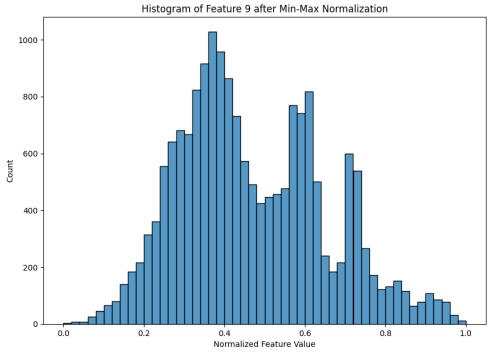

Figure 20: Feature 9
Min-Max Normalization

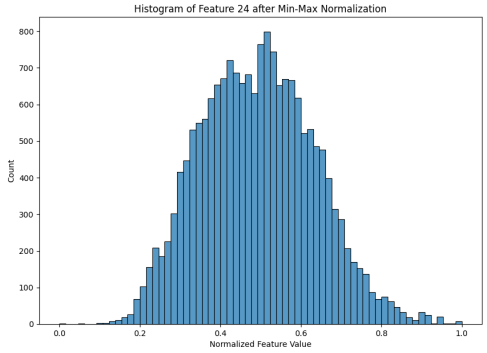

Figure 21: Feature 24
Min-Max Normalization

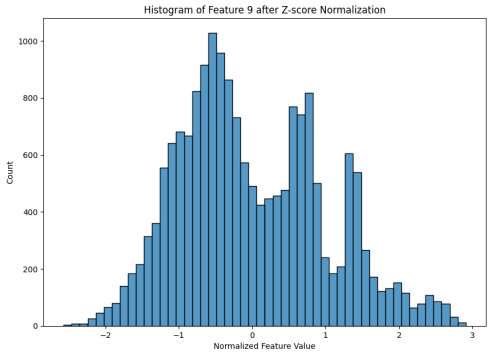

Figure 22: Feature 9
Z-Score Normalization

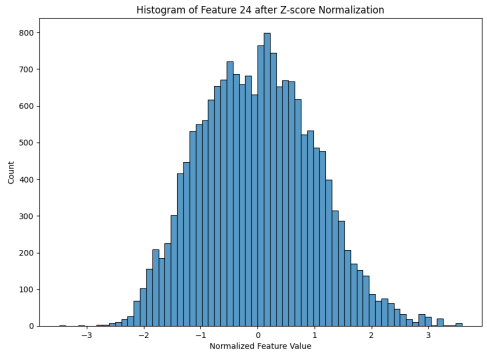

Figure 23: Feature 24
Z-Score Normalization

As we can see normalization only changes the scale on the X-axis and doesn't change the relative count of any values or the shape of the histogram.

2 Feature Extraction

In this part, 28x28 pixel gray scale images will be processed. These images show handwritings of digits ranging from 0 to 4. The data of the pixels are stored in DataB.csv file. For simplicty, the images are flatten to length of 784 pixels. In addition to the value of each pixel, the number which handwriting is also attached at the end of the image rows. Since Dataset B is labeled already, applying a supervised classification like LDA is possible.

2.1 Principal Component Analysis (PCA)

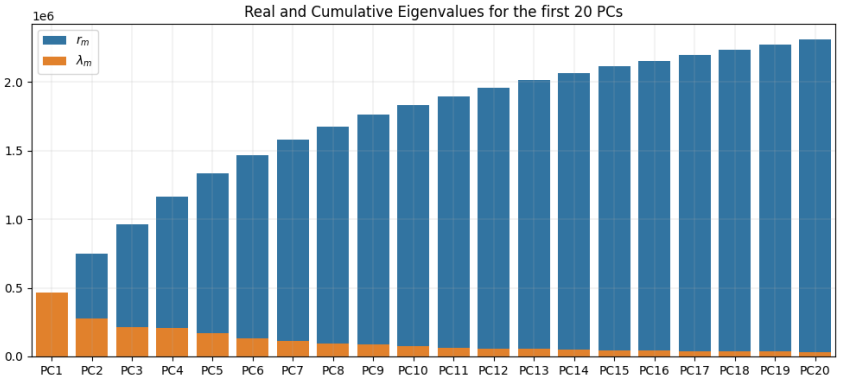

Figure 24: Real and Cumulative Eigenvalues for the first 20 PCs

2.2 Component Choice

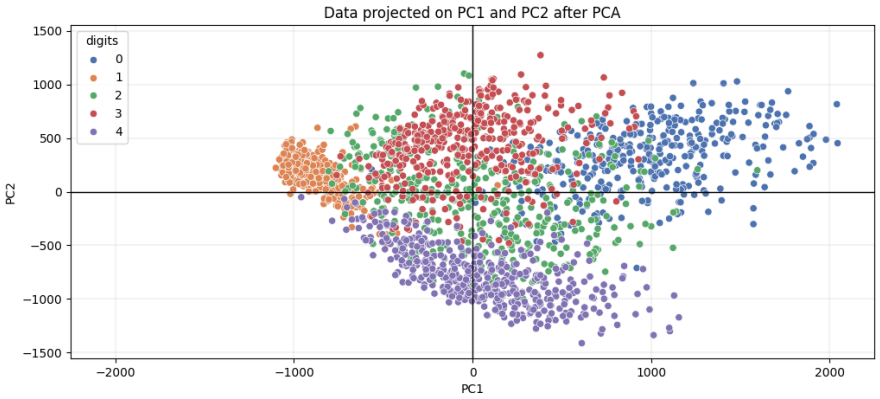

Figure 25: Data projected on PC1 and PC2 after PCA

After applying Principal Component Analysis, the projection of data on PC1 and PC2 illustrated on Figure 25. The classes seem hardly gathered but they are still into each other. If we choose the components having smaller eigenvalues, i.e., choosing the smaller varying directions, the projected data would be harder to be discriminated, like in Figure 26.

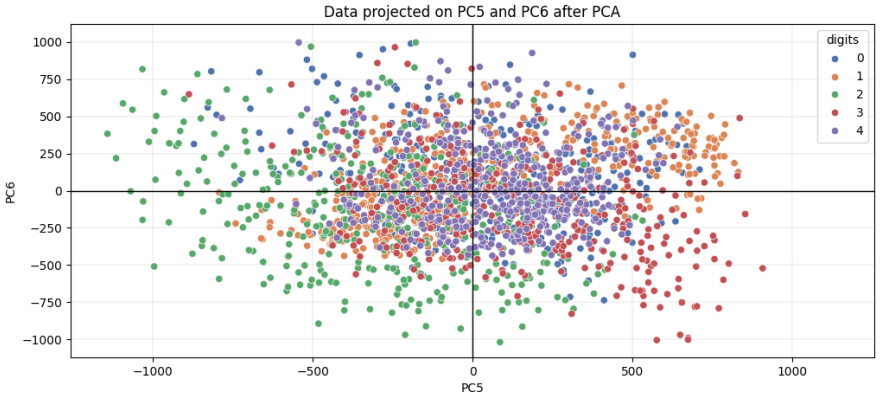

Figure 26: Data projected on PC5 and PC6 after PCA

2.3 Naive Bayes Classifier

In Figure 27, the classification error of the Naive Bayes classifier is shown for different PCA components. The plot results in that only the first 10 PCs would yield pretty much low error rate comparing the other amounts.

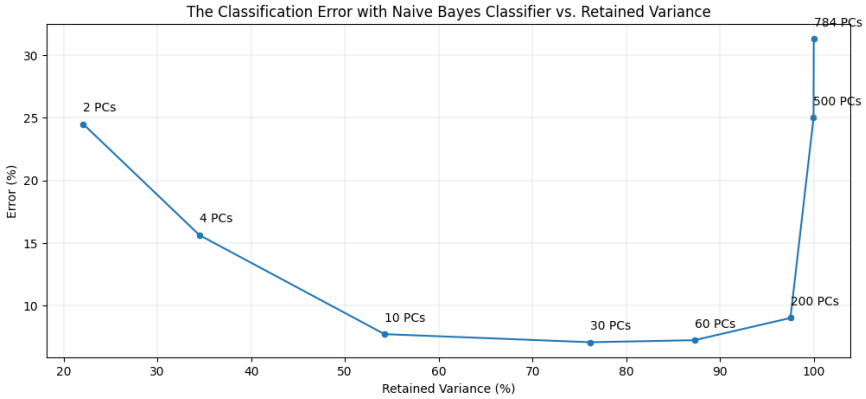

Figure 27: The Classification Error with Naive Bayes Classifier vs. Retained Variance

2.4 Linear Discriminant Analysis (LDA)

The resulting projection of data on PC1 and PC2 of LDA is plotted on Figure 28. The beautiful grouping of the classes is much more observable than Figure 25. The classes have been taken apart from each other and clustered. Thus, this enables a classifier to make more accurate decisions.

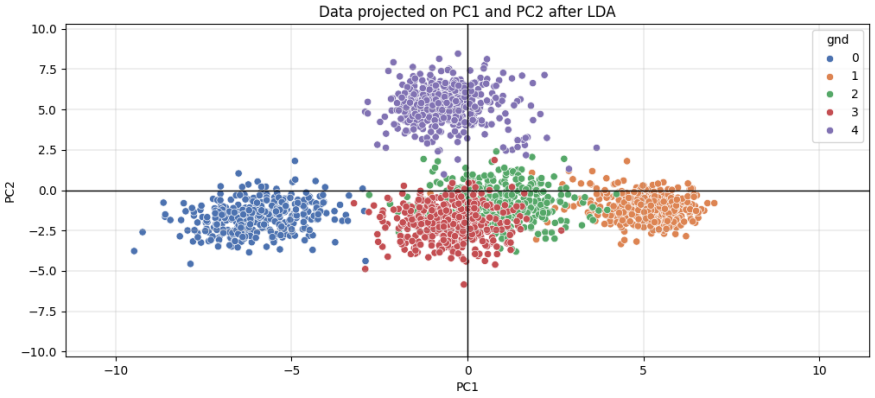

Figure 28: Data projected on PC1 and PC2 after LDA

2.5 PCA vs. LDA

In PCA, the inverse of the transformation can be achieved with use of orthonormal bases. Hence, we can reconstruct the images with projected values as in Figure 29 and Figure 30. The property of the handwriting is still noticable. Whereas, in LDA, understanding the reconstructed image is almost impossible. This is because the information of the data has been lost and the variance has been changed by separating and clustering the data. However, this make the LDA a better tool for classification while making PCA for transformation.

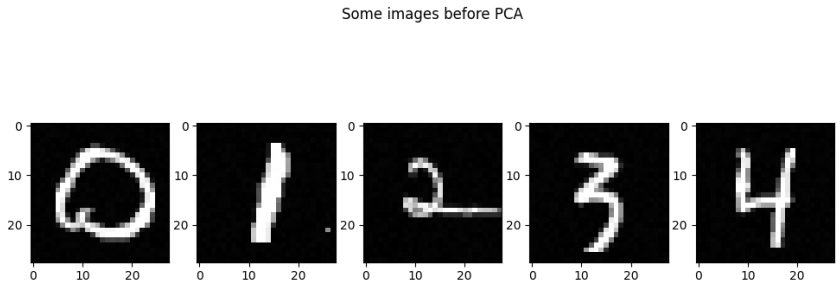

Figure 29: Some images before PCA

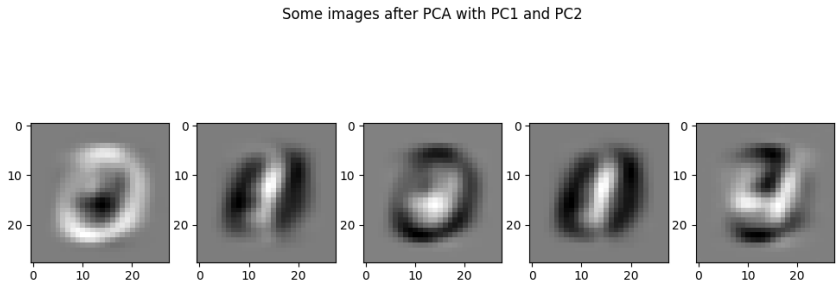

Figure 30: Some images after PCA with PC1 and PC2

2.6 PCA is the Best Linear Transformation

The following is a proof that PCA is the best linear transform under the following constraints.

Ommitted because we ran out of time, latex was hard

3 Parameter Estimation

Our goal for this section is to determine if the following estimatator for variance σ^2 is biased.

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{3}$$

Where n is the sample size, μ represents sample mean, and x_i are the samples.

Proof:

$$\begin{aligned}
B(\hat{\sigma}^2) &= E[\hat{\sigma}^2] - \sigma^2 \\
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \hat{\mu})^2\right] - \sigma^2 \\
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}((x_i - \mu) - (\mu - \hat{\mu})^2\right] - \sigma^2 \\
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}\left((x_i - \mu)^2 - 2(x_i - \mu)(\mu - \hat{\mu}) - (\mu - \hat{\mu})^2\right)\right] - \sigma^2 \\
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}\left((x_i - \mu)^2 - 2(x_i - \mu)(\mu - \hat{\mu}) - (\mu - \hat{\mu})^2\right)\right] - \sigma^2 \\
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}\left((x_i - \mu)^2\right) - \frac{2(\hat{\mu} - \mu)}{n-1}\sum_{i=1}^{n}(x_i - \mu)\right. \\
&\qquad \left. + \frac{1}{n-1}\sum_{i=1}^{n}(\hat{\mu} - \mu)^2)\right] - \sigma^2
\end{aligned}$$

$$\text{Apply } \hat{\mu} - \mu = \tfrac{1}{n}\textstyle\sum_{i=1}^{n}(x_i - \mu)$$

$$\begin{aligned}
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2 - \frac{2n}{n-1}(\hat{\mu} - \mu)^2 + \frac{n}{n-1}(\hat{\mu} - \mu)^2)\right] - \sigma^2 \\
&= E\left[\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)^2\right] - \frac{n}{n-1}E\left[\hat{\mu} - \mu)^2\right] - \sigma^2
\end{aligned}$$

Computing $E\left[(\hat{\mu}^2 - \mu)^2\right]$

$$\begin{aligned}
E\left[(\hat{\mu}^2 - \mu)^2\right] &= var(\hat{\mu}) \\
&= var\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n} var(x_i) \\
&= \frac{1}{n^2}\sum_{i=1}^{n} \sigma^2 = \frac{\sigma^2}{n}
\end{aligned}$$

Continuing with proof

$$\begin{aligned}
\cdots &= \frac{1}{n-1}\sum_{i=1}^{n} E\left[(x_i - \mu)^2\right] - \frac{n}{n-1}\left(\frac{\sigma^2}{n}\right) - \sigma^2 \\
&= \frac{1}{n-1}\sum_{i=1}^{n} \sigma^2 - \frac{n}{n-1}\left(\frac{\sigma^2}{n}\right) - \sigma^2 \\
&= \frac{n}{n-1}\sigma^2 - \frac{1}{n-1}\sigma^2 - \sigma^2 \\
&= \sigma^2 - \sigma^2 \\
&= 0
\end{aligned}$$

We find that $B(\hat{\sigma}^2) = 0$ and therefore we can conclude that the estimator (3) is unbiased.

4 Conclusion

In conculusion we were successfully able to meet our objectives. However the report is poor because we ran out of time.

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J.

Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2