ECE 657 - Assignment IV

Odunayo Adekoya

August 2, 2024

Question 1: MLE and MAP Estimation

1.1a Write down the likelihood function $L(\lambda)$ for the exponential distribution

$L(\lambda) = \lambda e^{-\lambda \cdot 2.1} \cdot \lambda e^{-\lambda \cdot 2.5} \cdot \lambda e^{-\lambda \cdot 1.8} \cdot \lambda e^{-\lambda \cdot 2.4} \cdot \lambda e^{-\lambda \cdot 2.0}$

$L(\lambda) = \lambda^5 e^{-\lambda(2.1+2.5+1.8+2.4+2.0)}$

$L(\lambda) = \lambda^5 e^{-\lambda \cdot 10.8}$

1.1b Derive the log-likelihood function $\ell(\lambda)$

$\ell(\lambda) = ln(\lambda^5 e^{-\lambda \cdot 10.8})$

$\ell(\lambda) = ln(\lambda^5) + ln(e^{-\lambda \cdot 10.8})$

$\ell(\lambda) = ln(\lambda^5) + (-\lambda \cdot 10.8)$

$\ell(\lambda) = 5ln(\lambda) - \lambda \cdot 10.8$

1.1c Determine the maximum likelihood estimate (MLE) for the rate parameter λ

Taking the derivative of : $\ell(\lambda) = 5ln(\lambda) - \lambda \cdot 10.8$

$\frac{d\ell(\lambda)}{dt} = \frac{d}{d}(5ln(\lambda) - 10.8\lambda)$

$\frac{d\ell(\lambda)}{d\lambda} = \frac{5}{\lambda} - 10.8$

$\frac{5}{\lambda} - 10.8 = 0$

$\frac{5}{\lambda}$=10.8

$\lambda = \frac{5}{10.8}$

$\hat{\lambda} \approx 0.463$

1.1d Calculate the MLE using the provided data

$mean = \frac{(2.1+2.5+1.8+2.4+2.0)}{5}$

$mean = \frac{10.8}{5}$

$mean = 2.16$

$MLE = \frac{1}{mean}$

$\lambda = \frac{1}{2.16}$

$\hat{\lambda} \approx 0.463$

1.2a Write down the prior distribution $P(\lambda)$ for the Gamma distribution with a shape parameter $\alpha = 2$ and a rate parameter $\beta = 1$

$P(\lambda) = \frac{1^2\lambda^{21}e^{-1\lambda}}{\Gamma(2)}$

$P(\lambda) = \frac{1\lambda^1 e^{-\lambda}}{\Gamma(2)}$

Since $\Gamma(2) = (2-1)! = 1! = 1:$

$P(\lambda) = \lambda e^{-\lambda}$

1.2b Derive the log-posterior distribution log P (—X) by combining the log-likelihood function from the previous sub-question with the log-prior

Log likelihood : $\ell(\lambda) = 5ln(\lambda) - 10.8\lambda$

Log Prior function : $logP(\lambda) = log(\lambda e^{-\lambda})$
$= log(\lambda) + log(e^{-\lambda})$
$= log(\lambda)\lambda$

Log Posterior Function: $logP(\lambda|X) = \ell(\lambda) + logP(\lambda)$
$logP(\lambda|X) = (5ln(\lambda)10.8\lambda) + (log(\lambda)\lambda)$
$logP(\lambda|X) = 5ln(\lambda)10.8\lambda + log(\lambda)\lambda$
$logP(\lambda|X) = 6ln(\lambda) - 11.8\lambda$

1.2c Determine the maximum a posteriori (MAP) estimate for the rate parameter

$\frac{d}{d\lambda}(6ln(\lambda) - 11.8\lambda) = 0$

By taking derivatives:

$\frac{d}{d\lambda}(6ln(\lambda)) = \frac{6}{\lambda}$

$\frac{d}{d\lambda}(-11.8\lambda) = -11.8$

Therefore: $\frac{6}{\lambda} - 11.8 = 0$

1.2d Calculate the MAP estimate using the provided data

Solving for λ:

$\lambda = \frac{6}{11.8}$

$\lambda_{map} \approx 0.508$

1.3 In the context of parameter estimation in statistical models, what are the key differences between Maximum A Posteriori (MAP) estimation and Maximum Likelihood Estimation (MLE), and how does each method handle prior information and data scarcity?

Answer: **MLE** relies solely on the observed data and does not use any prior information. It can be highly sensitive to data scarcity and may produce estimates with high variance. **MAP** on the other hand, combines both observed data and prior information, making it more robust to data scarcity and potentially leading to better parameter estimates by leveraging prior knowledge.

Question 2: Naive Bayes

2.1 Given the following small dataset of weather conditions and the corresponding decision to play tennis, use Naive Bayes to predict whether to play tennis based on the conditions: Weather = Sunny and Temperature = Cool

2.1

Play Tennis : $P(Yes) : 5/10 = 1/2$

$P(No) : 5/10 = 1/2$

$P(Rainy \mid Yes) = 2/5$	$P(Rainy \mid No) = 2/5$
$P(Sunny \mid Yes) = 1/5$	$P(Sunny \mid No) = 3/5$
$P(Overcast \mid Yes) = 2/5$	$P(Overcast \mid No) = 0/5$
$P(Hot \mid Yes) = 0$	$P(Hot \mid No) = 1/5$
$P(Mild \mid Yes) = 2/5$	$P(Mild \mid No) = 2/5$
$P(Cool \mid Yes) = 3/5$	$P(Cool \mid No) = 2/5$

$\therefore P(Yes) \times P(Sunny \mid Yes) \times P(Cool \mid Yes)$

$= 1/2 \times 1/5 \times 3/5 = 3/50 = 0.06$

$P(No) \times P(Sunny \mid No) \times P(Cool \mid No)$

$= 1/2 \times 3/5 \times 2/5 = 6/50 = 0.12$

$\therefore$ Argmax $\Rightarrow$ To play tennis when Weather = Sunny & Temperature = Cool
$(0.06, 0.12)$ = **NO**

2.2

2. Which Naive Bayes Classifier did you choose and why?

From the histograms, we can observe the following:

- Most features appear to follow a roughly Gaussian (normal) distribution, although some may have slight skewness or kurtosis.
- None of the features are discrete counts or binary/boolean values.

Given that the features are continuous and approximately normally distributed, Gaussian Naive Bayes is the most appropriate choice for this dataset. This classifier assumes that the features follow a Gaussian distribution, making it well-suited for the Wine dataset's characteristics.

7. Display the accuracy scores for the classifier without scaling, with StandardScaler, and with MinMaxScaler

```python
print("Accuracy without scaling:", accuracy)
print("Standard Scaler Accuracy with Scaling:", accuracy_scaled)
print("MinMAx Accuracy with Scaling:", min_max_accuracy)
```

```
Accuracy without scaling: 0.9814814814814815
Standard Scaler Accuracy with Scaling: 0.9814814814814815
MinMAx Accuracy with Scaling: 0.9814814814814815
```

8. How do the accuracy scores compare with each other? It is generally said that the Naive Bayes Classifier is independent of feature scaling, why is it said so? As per our test, can we say that the Naive Bayes classifier is independent of feature scaling? Explain your answer in detail.

Naive Bayes computes likelihoods and prior probabilities from the training data. These probabilities are ratios that compare how likely a particular feature value is given a class, relative to other feature values for that class. Scaling changes the numeric range but doesn't alter the ratios between different feature values within the same class, thus preserving the relative probabilities crucial for classification.
The identical accuracy scores suggest that scaling did not improve or degrade the classifier's performance, which aligns with the theoretical understanding of Naive Bayes i.e. Naive Bayes is indeed independent of feature scaling. This characteristic makes Naive Bayes particularly robust when dealing with datasets where feature scaling might be challenging or unnecessary.

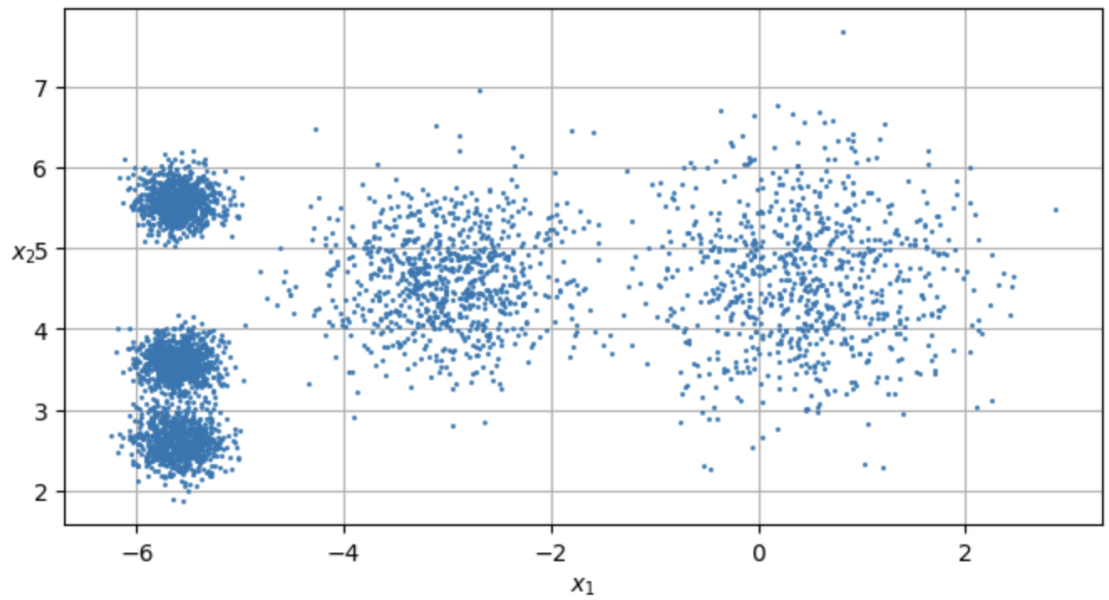
9. Now, for the original dataset multiply the entire ‘magnesium’ column by 10,000,000 and then perform steps 2 to 8 again.

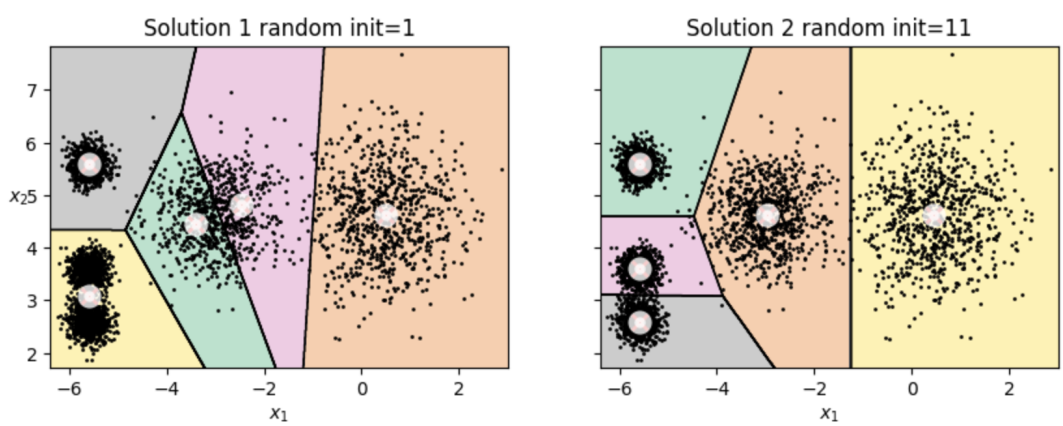
```
New Accuracy without scaling: 0.46296296296296297
New Standard Scaler Accuracy with Scaling: 0.9814814814814815
New MinMAx Accuracy with Scaling: 0.9814814814814815
```

When features are not scaled (raw values), the accuracy drops significantly. When features are scaled using Standard Scaler or MinMax Scaler, the accuracy dramatically improves and remains consistent.
Therefore, despite the varied accuracy scores with and without scaling, the key observation is that scaling impacts the raw accuracy values but doesn’t affect the relative performance of Naive Bayes across different scaling methods. This reaffirms that Naive Bayes is robust against feature scaling variations, as its classification decisions primarily rely on the relative distributions and conditional probabilities rather than the absolute values of features. Hence, we can confidently say that the Naive Bayes classifier demonstrates independence from feature scaling based on the consistency and performance observed across different scaling scenarios.

Question 3: K-means Limitations

3.1 First, train K-means on this data with n_clusters=5, n_init=1, init="random", and random_state=1. Repeat this process using the same hyperparameters, but modified random_state=11. Plot the results and explain any potential issues.





A potential issue is that the plots may show different clustering results for random-state=1 and random-state=11, illustrating that the algorithm has converged to different local optima. This is a significant issue because it means that the final clustering depends heavily on the initial centroid positions. If the initial centroids are not well-placed, the algorithm may end up with suboptimal clusters, leading to poor performance.

3.2: Suggest a good set of initial centroids and explain why you think it is a good initializiation for this problem.

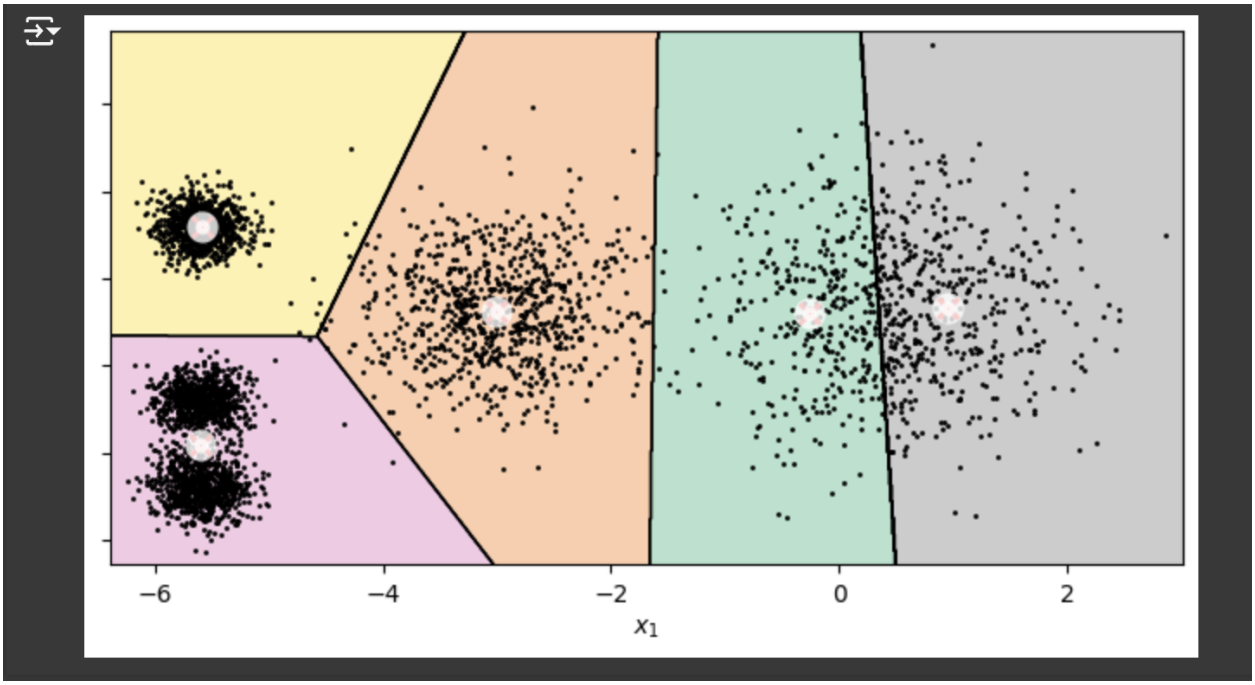
```
KMeans
KMeans(init=array([[ 0.5,  2.5],
       [ 1.5, -1. ],
       [-1.5,  2. ],
       [-2. , -1. ],
       [ 2. ,  2. ]]),
       n_clusters=5, n_init=1, random_state=42)
```

We can choose initial centroids that are likely to be close to the true cluster centers. This approach is feasible when dealing with low-dimensional data that can be easily visualized.

Spread Out: The manually selected initial centroids are spread out across the data space, which helps to cover different regions of the data distribution. This increases the likelihood of the algorithm converging to a global optimum.
Approximate Centers: The centroids are chosen to be approximately in the center of each visible cluster. This reduces the number of iterations required for convergence and improves the quality of the final clusters.
Data-Driven: The selection is based on the visual inspection of the data, ensuring that the centroids are relevant to the actual data distribution.

By carefully selecting initial centroids based on the data distribution, we can improve the performance of the K-means algorithm. This approach is particularly useful for low-dimensional data where visualization is possible. For higher-dimensional data, automated methods like K-means++ are recommended to achieve a good initialization.

3.3: Train your algorithm based on the suggested initialization, plot the results, and discuss what has changed.



Discussion of Changes in Results

The clustering results with manually selected initial centroids are more stable and consistent compared to the random initializations. The clusters are well-separated, and the centroids are closer to the actual cluster centers.
Using manually selected initial centroids has likely led to faster convergence since the centroids are already placed near the true centers of the clusters. This reduces the number of iterations needed to reach the final clustering.
The quality of clusters obtained with manual initialization is generally better, as evidenced by well-defined clusters and reduced overlap between clusters. This contrasts with the random initializations, where some clusters may be less well-defined or overlap more.
The results from random-state=1 and random-state=11 show that random initialization can lead to different clustering outcomes, demonstrating the sensitivity of K-means to initial centroid placement. Some clusters may be poorly defined or wrongly assigned, especially if the initial centroids are not well-spread out.

3.4: Based on what you did on the previous parts, suggest what can be done to avoid the problem that happened on part 1, give at least 2 solutions

To avoid the problem of K-means converging to a local optimum due to poor centroid initialization, there are several strategies that can be employed. Two of them are:
- K-means++ Initialization: is an initialization algorithm that improves the chances of finding a better set of initial centroids by spreading them out.
- Increasing the Number of Initializations (n-init): Running the algorithm multiple times with different initializations increases the chances of finding a global optimum.

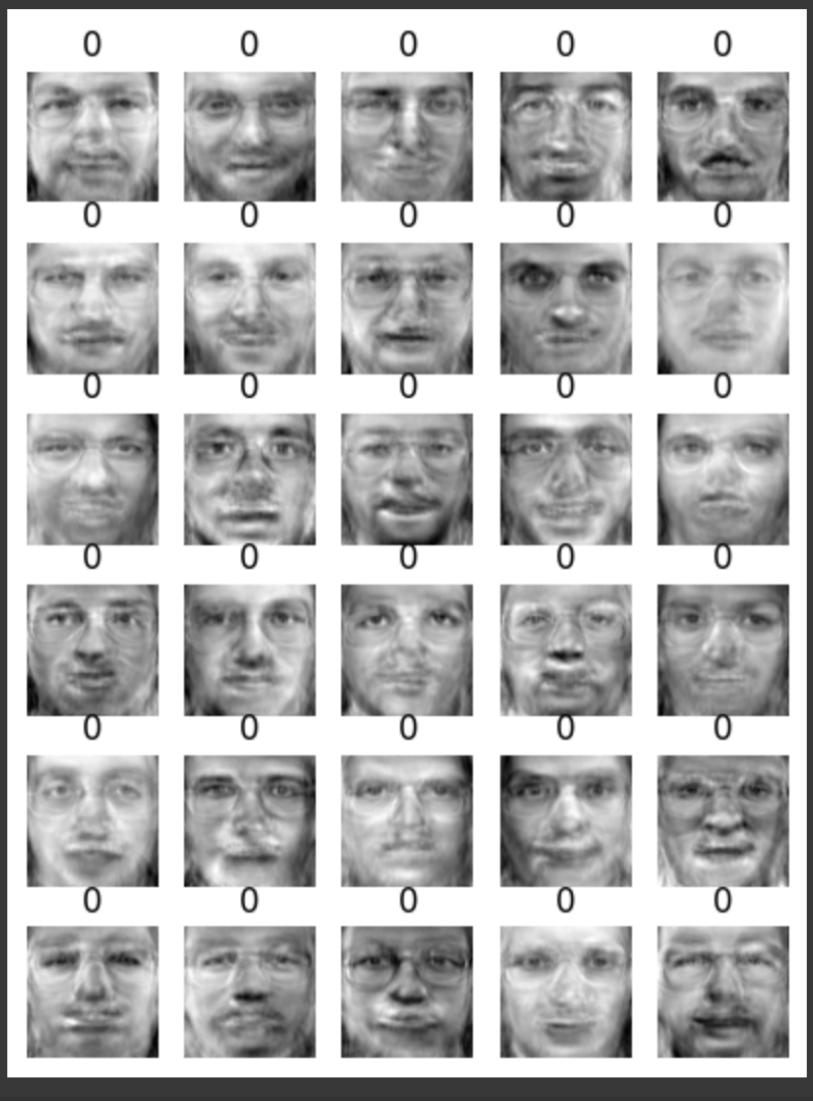
Question 4: Gaussian Mixture Models for Generating New Faces

4.1

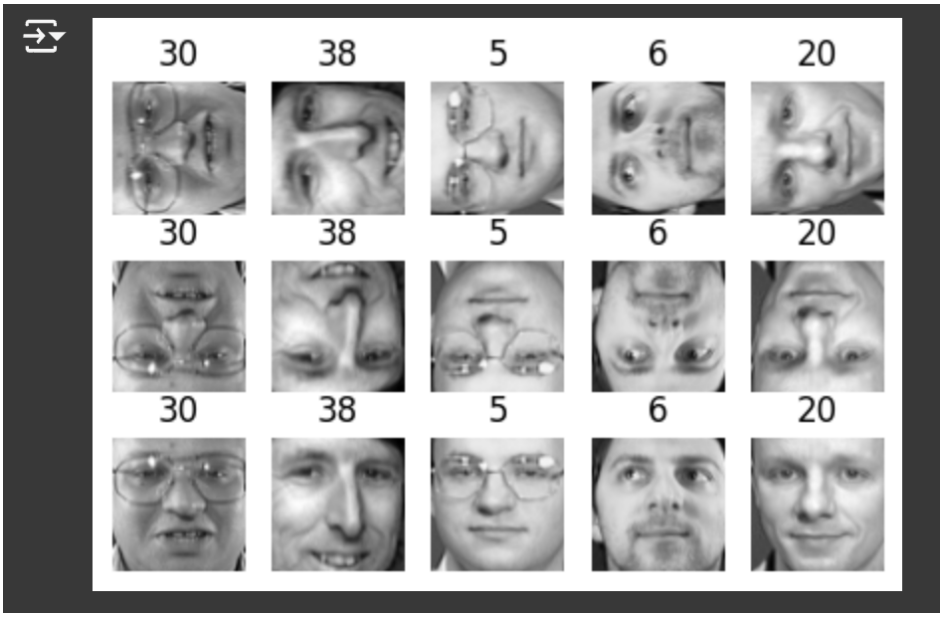
```
(280, 4096) (280,)
(80, 4096) (80,)
(40, 4096) (40,)
```

4.2: Generate 30 new faces by sampling from your model and visualize them

```
plot_faces(generated_faces, y_gen_faces)
```



4.3 : Plot the 15 resulting images



4.4: Compute the log-likelihood of each sample. (you can use the gm methods.) Compare these log-likelihoods with the main data scores (maybe for the first 15 samples). Explain the results and describe how we use GMMs for anomaly detection

```
Log-likelihoods of the first 15 main data samples:
 [ -93.11769236  -44.89985091  -58.18466248 -100.15162822  -32.38941528
  -50.61067803  -50.10760498  -67.51532567  -22.6264372   -52.44929163
  -64.32004817  -91.91671652  -45.35031923  -50.61110031  -39.6306175 ]
Log-likelihoods of the 15 transformed images:
 [ -84.66114528  -77.70396213  -88.90884291 -133.43542404  -89.17417693
 -107.17807592  -82.20333448 -119.53859853 -160.58447336 -105.81951043
  -57.90648653  -66.20373302  -46.91520224  -51.88311901  -56.92828712]
```

The original images (main data) should have higher log-likelihoods because they are similar to the data used to train the GMM while the transformed images (rotated, flipped, darkened) should have lower log-likelihoods because they deviate from the distribution of the original training data.

GMMs can be used for anomaly detection by evaluating the log-likelihood of each sample. Samples with significantly lower log-likelihoods compared to the majority of the data are considered anomalies.

Question 5: Principal Component Analysis (PCA)

5.1 Compute the covariance matrix of the data

5.1

$$\text{mean}_i = \frac{8+5+1+17}{4} = \frac{31}{4} = 7.75$$

$$\text{mean}_{ii} = \frac{12+1+15+4}{4} = \frac{32}{4} = 8$$

STD	Feature 1	Feature 2
	$8 - 7.75 = 0.25$	$12 - 8 = 4$
	$5 - 7.75 = -2.75$	$1 - 8 = -7$
	$1 - 7.75 = -6.75$	$15 - 8 = 7$
	$17 - 7.75 = 9.25$	$4 - 8 = -4$

$$\begin{aligned}
Cov_1 &= \left[(0.25)^2 + (-2.75)^2 + (-6.75)^2 + (9.25)^2\right]/3 \\
&= \left[(0.0625) + (7.5625) + (45.5625) + (85.5625)\right]/3 \\
&= 138.75/3 \\
&= 46.25
\end{aligned}$$

$$\begin{aligned}
Cov_2 &= \left[(4)^2 + (-7)^2 + (7)^2 + (-4)^2\right]/3 \\
&= (16+49+49+16)/3 \\
&= 130/3 \\
&= 43.33
\end{aligned}$$

$$\begin{aligned}
Cov_{1,2} &= \left[(0.25)\cdot(4) + (-2.75)\cdot(-7) + (-6.75)\cdot(7) + (9.25)(-4)\right]/3 \\
&= (1 + 19.25 - 47.25 - 37)/3 \\
&= -64/3 \\
&= -21.3
\end{aligned}$$

Covariance matrix : $\begin{bmatrix} 46.25 & -21.3 \\ -21.3 & 43.33 \end{bmatrix}$

5.2 Calculate the eigenvalues of the covariance matrix

5.2

$$|A - \lambda I| = 0$$

$$\begin{bmatrix} 46.25 & -21.3 \\ -21.3 & 43.33 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0$$

$$\therefore \det \begin{vmatrix} 46.25 - \lambda & -21.3 \\ -21.3 & 43.33 - \lambda \end{vmatrix} = 0$$

$$(46.25 - \lambda)(43.33 - \lambda) - (-21.3)^2 = 0$$

$$46.25 \cdot 43.33 - 46.25\lambda - 43.33\lambda + \lambda^2 - 453.69 = 0$$

$$2{,}004.01 - 89.58\lambda + \lambda^2 - 453.69 = 0$$

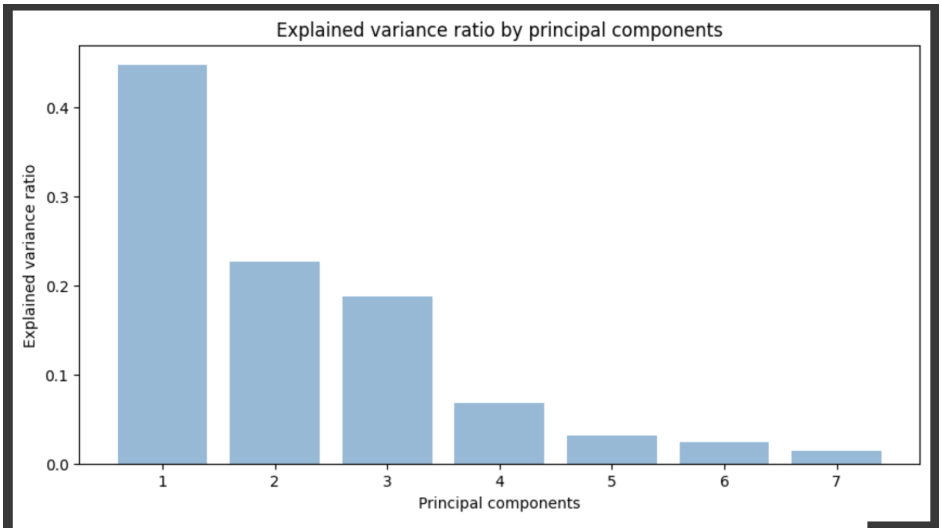
$$\lambda^2 - 89.58\lambda + 1550.32 = 0$$

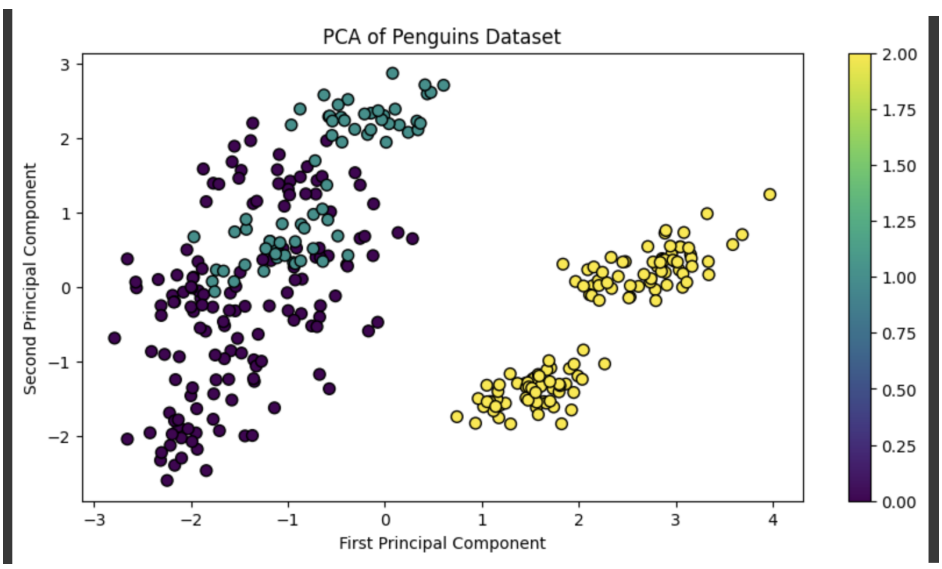
$$\lambda_1 = 66.14 \quad \& \quad \lambda_2 = 23.44$$

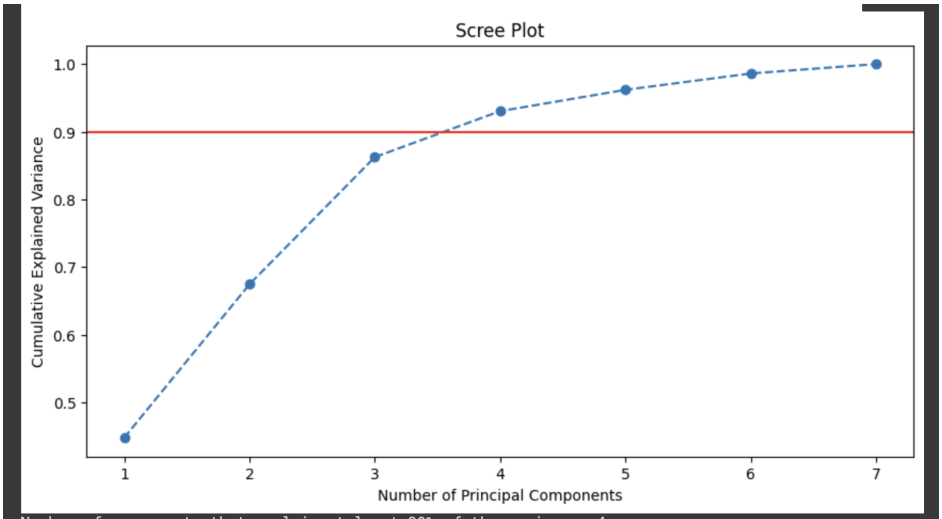
5.3 Calculate the normalized eigenvectors from the eigenvalues and covariance matrix

5.4 Choose the principal component with the highest eigenvalue and then reduce the original dataset to one dimension

5.5


Explained variance ratio by principal components
Explained variance ratio
0.0
0.1
0.2
0.3
0.4
1
2
3
4
5
6
7
Principal components


PCA of Penguins Dataset
Second Principal Component
First Principal Component
−3
−2
−1
0
1
2
3
4
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00




```
Number of components that explain at least 90% of the variance: 4
```

Question 6: Denoising Autoencoders

6.1 Print the encoder model summary and plot the encoder model.

```
Model: "encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_25 (InputLayer)       [(None, 28, 28, 1)]       0

 conv2d_80 (Conv2D)          (None, 28, 28, 32)        320

 max_pooling2d_35 (MaxPooli  (None, 14, 14, 32)        0
 ng2D)

 conv2d_81 (Conv2D)          (None, 14, 14, 64)        18496

 max_pooling2d_36 (MaxPooli  (None, 7, 7, 64)          0
 ng2D)

 conv2d_82 (Conv2D)          (None, 7, 7, 128)         73856

=================================================================
Total params: 92672 (362.00 KB)
Trainable params: 92672 (362.00 KB)
Non-trainable params: 0 (0.00 Byte)
```

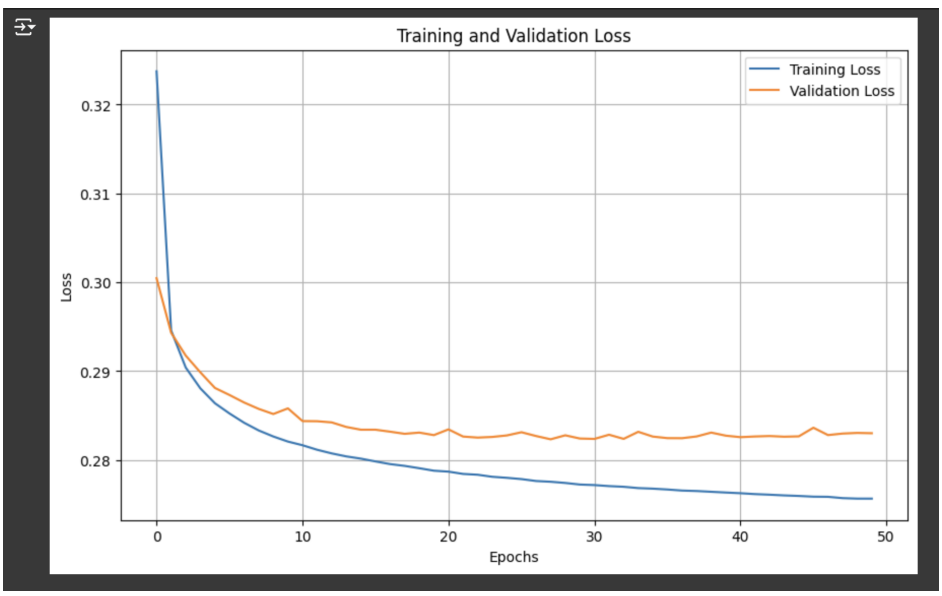
6.2 Print the decoder model summary and plot the decoder model.

```
_________________________________________________________________
Model: "decoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_26 (InputLayer)       [(None, 7, 7, 128)]       0

 conv2d_83 (Conv2D)          (None, 7, 7, 128)         147584

 up_sampling2d_31 (UpSampli  (None, 14, 14, 128)       0
 ng2D)

 conv2d_84 (Conv2D)          (None, 14, 14, 64)        73792

 up_sampling2d_32 (UpSampli  (None, 28, 28, 64)        0
 ng2D)

 conv2d_85 (Conv2D)          (None, 28, 28, 32)        18464

 conv2d_86 (Conv2D)          (None, 28, 28, 1)         289

=================================================================
Total params: 240129 (938.00 KB)
Trainable params: 240129 (938.00 KB)
Non-trainable params: 0 (0.00 Byte)
```

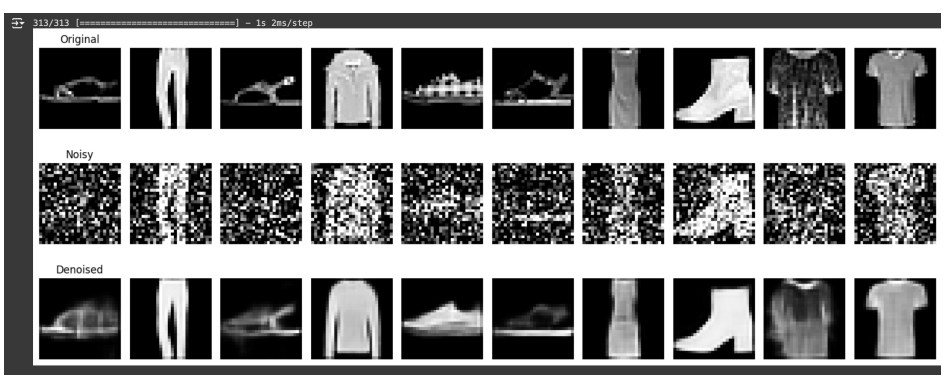
6.3 Build the Autoencoder

```
Model: "autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_25 (InputLayer)       [(None, 28, 28, 1)]       0

 encoder (Functional)        (None, 7, 7, 128)         92672

 decoder (Functional)        (None, 28, 28, 1)         240129

=================================================================
Total params: 332801 (1.27 MB)
Trainable params: 332801 (1.27 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

6.4 Plot a graph of the training and validation losses against the number of epochs



6.5



6.6

2 areas where Denoising Autoencoders can be used include:
- Image Restoration: used in image processing to clean up noisy images. This application is valuable in various fields, including medical imaging (e.g., MRI scans), satellite imaging, and photography.
- Speech Enhancement: In audio processing, denoising autoencoders can be applied to improve the quality of speech signals by removing background noise. This is useful in telecommunications, hearing aids, and voice recognition systems.
Specifically in encryption, Autoencoders has the ability to compress and reconstruct data. The encoder part of an autoencoder can be used to transform plaintext data into a compressed, encoded representation that is not directly interpretable. This encoded representation can serve as the encrypted data. The encoder could map the original data into a latent space (compressed representation), which is then used as the "ciphertext". The decoder would then be required to reconstruct the original data from this encoded form. Subsequently, to decrypt the data, the decoder reconstructs the original data from the encoded (encrypted) representation. The decoder essentially reverses the encoding process, thus revealing the original data.
I don't think autoencoders would be effective because unlike traditional encryption algorithms, autoencoders do not inherently include mechanisms for key management, key generation, key exchange, and cryptographic security which is crucial for secure encryption.

Question 7: Discount Factors and Q-Learning

7.1: What should be seen are transition probabilities, rewards and possible actions

```
[1] transition_probabilities = {
        ('s0', 'a0', 's0'): 0.7,
        ('s0', 'a0', 's1'): 0.3,
        ('s0', 'a1', 's0'): 1.0,
        ('s0', 'a2', 's1'): 0.2,
        ('s0', 'a2', 's0'): 0.8,
        ('s1', 'a0', 's1'): 1.0,
        ('s1', 'a2', 's2'): 1.0,
        ('s2', 'a1', 's0'): 0.8,
        ('s2', 'a1', 's1'): 0.1,
        ('s2', 'a1', 's2'): 0.1,
    }

    rewards = {
        ('s0', 'a0', 's0'): 10,
        ('s1', 'a2', 's2'): 50,
        ('s2', 'a1', 's0'): 40,
    }

    possible_actions = {
        's0': ['a0', 'a1', 'a2'],
        's1': ['a0', 'a2'],
        's2': ['a1'],
    }
```

7.2: Take your discount factor to be 0.9. Perform Q-learning and report the Q-values for each (state, action) pair. Based on that, what is the optimal policy?

```
[7] Q_values

array([[39.49939329, 30.15078333, 30.15078333],
       [ 0.        ,        -inf,  0.        ],
       [       -inf,  0.        ,        -inf]])
```

```
[8] Q_values.argmax(axis=1)  # optimal action for each state

array([0, 0, 1])
```

7.3: Perform the same procedure but this time with a discount factor of 0.95. Did your optimal policy change? Explain your results.

```
[23] gamma2 = 0.95  # the discount factor
     alpha2 = 0.1  # the learning rate
     q_learning(Q_values2, gamma2, alpha)
```

```
Q_values2

array([[75.08036238, 65.70418176, 65.70418176],
       [ 0.        ,        -inf,  0.        ],
       [       -inf,  0.        ,        -inf]])
```

```
[25] Q_values2.argmax(axis=1)

array([0, 0, 1])
```

7.4: Take a look at the definition of the discount factor and say what it shows? Can the optimal policy change if you modify the discount factor?

The discount factor dictates how much the agent cares about future rewards. A higher discount factor makes the agent more far-sighted, potentially altering its decisions to favor long-term benefits over immediate gains.
Yes, the optimal policy can change if you modify the discount factor because it alters the agent's valuation of future rewards.

Question 8: Kaggle Competition

Kindly evaluate attached file