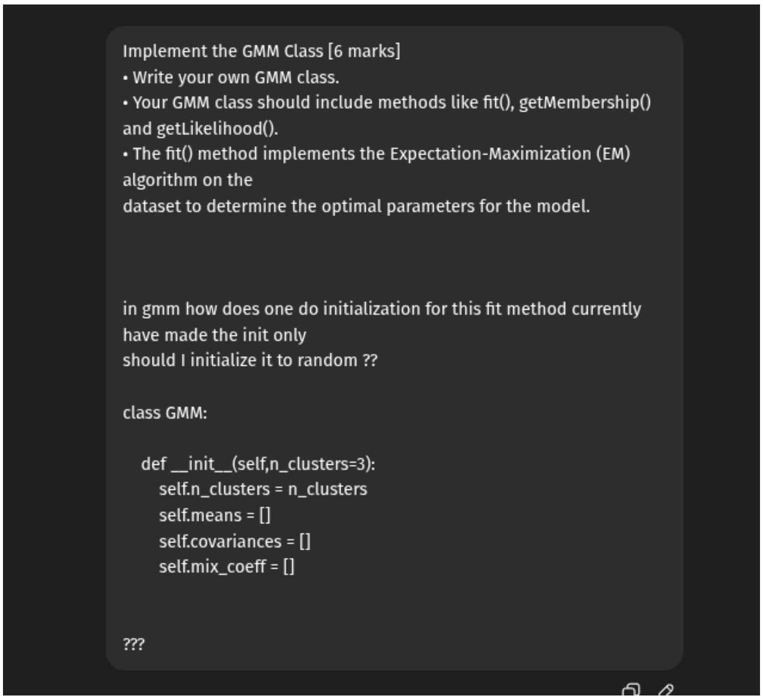
Prompts for assignment 2

Q1

1. Learning rate issues https://chatgpt.com/share/68c85254-6114-8013-9040-8e50edd1b0c5
2. Fixes - https://g.co/gemini/share/52c04a1a5049

Q3

Initalization and categorical variable doubts -

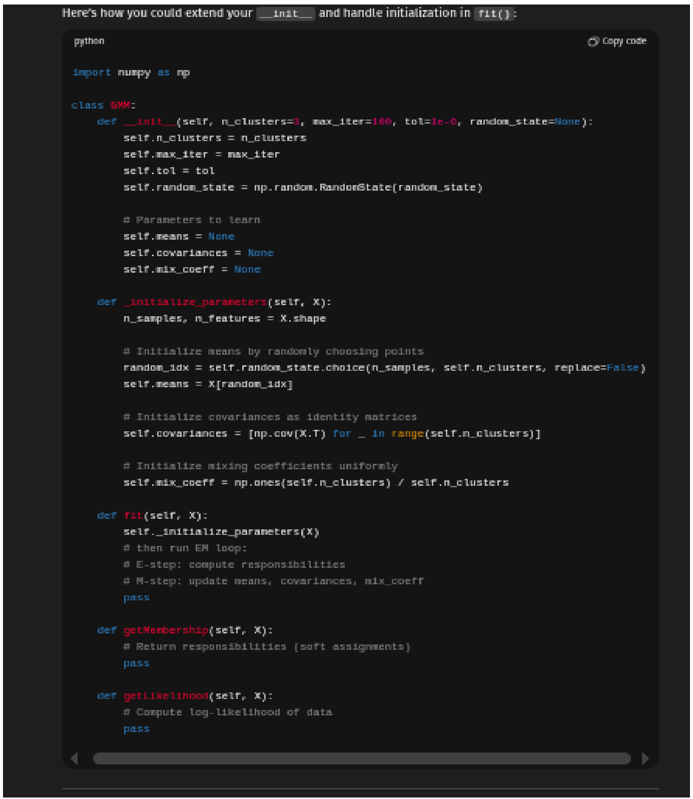


Implement the GMM Class [6 marks]
• Write your own GMM class.
• Your GMM class should include methods like fit(), getMembership() and getLikelihood().
• The fit() method implements the Expectation-Maximization (EM) algorithm on the
dataset to determine the optimal parameters for the model.

in gmm how does one do initialization for this fit method currently have made the init only
should I initialize it to random ??

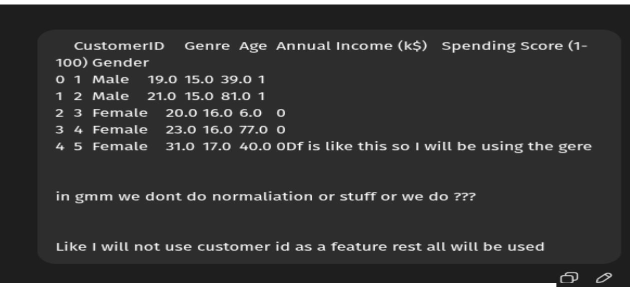
```
class GMM:

    def __init__(self,n_clusters=3):
        self.n_clusters = n_clusters
        self.means = []
        self.covariances = []
        self.mix_coeff = []
```

???



Here's how you could extend your `__init__` and handle initialization in `fit()`:

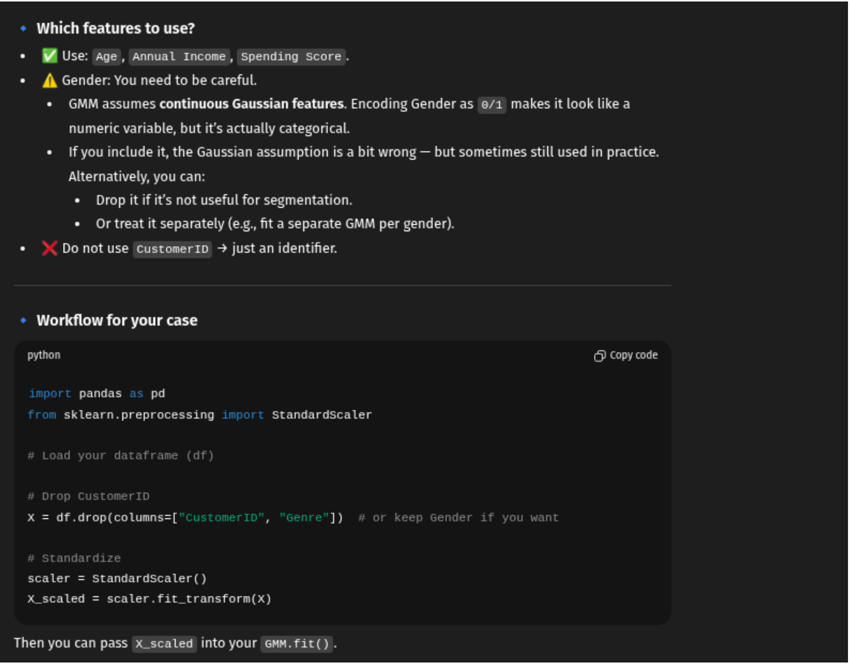
```python
import numpy as np

class GMM:
    def __init__(self, n_clusters=3, max_iter=100, tol=1e-6, random_state=None):
        self.n_clusters = n_clusters
        self.max_iter = max_iter
        self.tol = tol
        self.random_state = np.random.RandomState(random_state)

        # Parameters to learn
        self.means = None
        self.covariances = None
        self.mix_coeff = None

    def _initialize_parameters(self, X):
        n_samples, n_features = X.shape

        # Initialize means by randomly choosing points
        random_idx = self.random_state.choice(n_samples, self.n_clusters, replace=False)
        self.means = X[random_idx]

        # Initialize covariances as identity matrices
        self.covariances = [np.cov(X.T) for _ in range(self.n_clusters)]

        # Initialize mixing coefficients uniformly
        self.mix_coeff = np.ones(self.n_clusters) / self.n_clusters

    def fit(self, X):
        self._initialize_parameters(X)
        # then run EM loop:
        # E-step: compute responsibilities
        # M-step: update means, covariances, mix_coeff
        pass

    def getMembership(self, X):
        # Return responsibilities (soft assignments)
        pass

    def getLikelihood(self, X):
        # Compute log-likelihood of data
        pass
```



```
   CustomerID    Genre  Age  Annual Income (k$)   Spending Score (1-
100) Gender
0  1   Male    19.0  15.0  39.0  1
1  2   Male    21.0  15.0  81.0  1
2  3   Female    20.0  16.0  6.0   0
3  4   Female    23.0  16.0  77.0  0
4  5   Female    31.0  17.0  40.0  0Df is like this so I will be using the gere
```

in gmm we dont do normaliation or stuff or we do ???

Like I will not use customer id as a feature rest all will be used



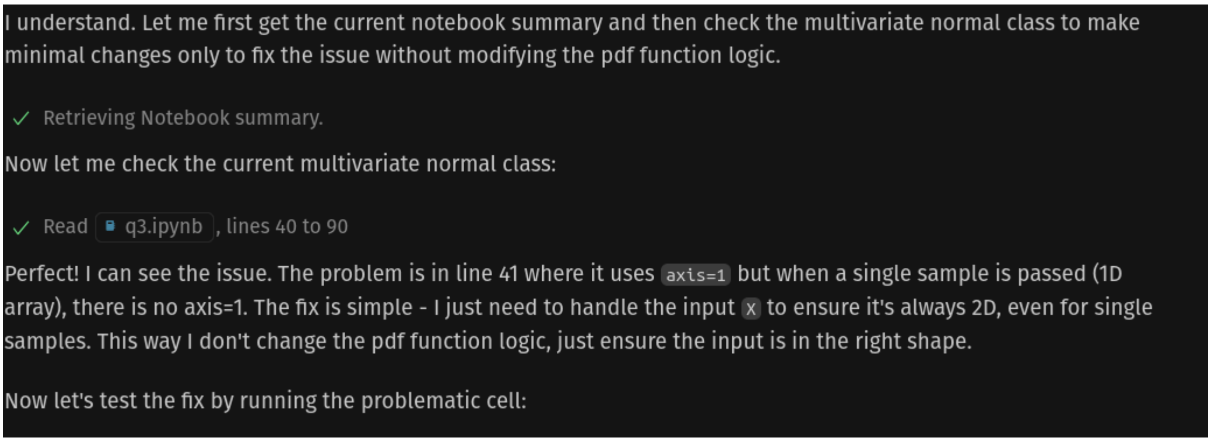
- **Which features to use?**
- ✅ Use: `Age`, `Annual Income`, `Spending Score`.
- ⚠ Gender: You need to be careful.
    - GMM assumes **continuous Gaussian features**. Encoding Gender as `0/1` makes it look like a numeric variable, but it's actually categorical.
    - If you include it, the Gaussian assumption is a bit wrong — but sometimes still used in practice. Alternatively, you can:
        - Drop it if it's not useful for segmentation.
        - Or treat it separately (e.g., fit a separate GMM per gender).
- ❌ Do not use `CustomerID` → just an identifier.

- **Workflow for your case**

```python
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load your dataframe (df)

# Drop CustomerID
X = df.drop(columns=["CustomerID", "Genre"])  # or keep Gender if you want

# Standardize
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Then you can pass `X_scaled` into your `GMM.fit()`.

Multivariate gaussian - https://g.co/gemini/share/075c71a3897d later final fixes done with copilot -



I understand. Let me first get the current notebook summary and then check the multivariate normal class to make minimal changes only to fix the issue without modifying the pdf function logic.

✓ Retrieving Notebook summary.

Now let me check the current multivariate normal class:

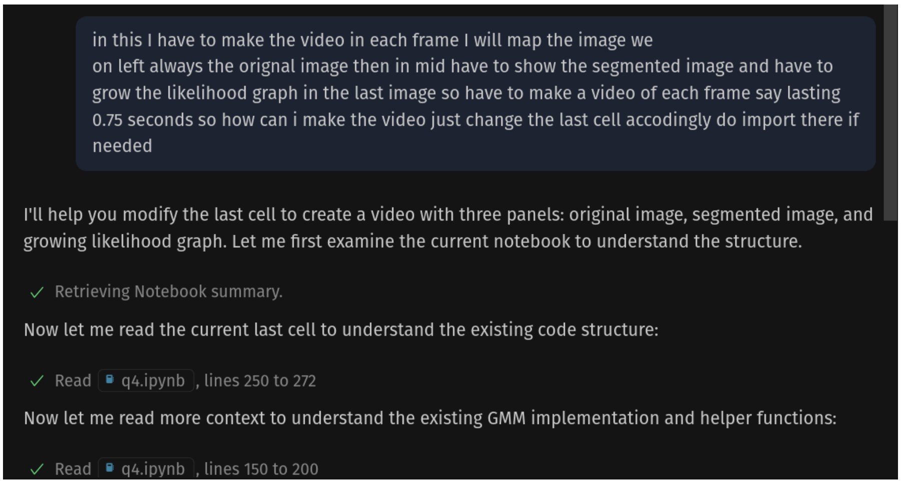
✓ Read q3.ipynb, lines 40 to 90

Perfect! I can see the issue. The problem is in line 41 where it uses `axis=1` but when a single sample is passed (1D array), there is no axis=1. The fix is simple - I just need to handle the input `x` to ensure it's always 2D, even for single samples. This way I don't change the pdf function logic, just ensure the input is in the right shape.

Now let's test the fix by running the problematic cell:

Q4

1. Some bugs and help - https://chatgpt.com/share/68d6f070-dc68-8001-ab24-8bcbcff93a11
2. Video making with claude (copilot)



in this I have to make the video in each frame I will map the image we on left always the orignal image then in mid have to show the segmented image and have to grow the likelihood graph in the last image so have to make a video of each frame say lasting 0.75 seconds so how can i make the video just change the last cell accodingly do import there if needed

I'll help you modify the last cell to create a video with three panels: original image, segmented image, and growing likelihood graph. Let me first examine the current notebook to understand the structure.

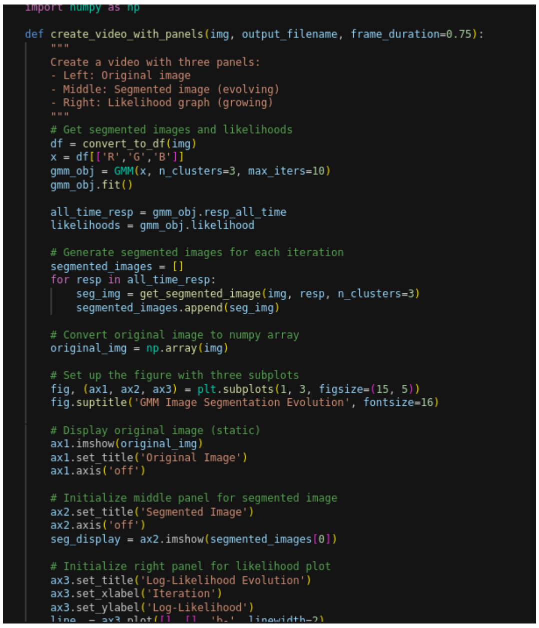
✓ Retrieving Notebook summary.

Now let me read the current last cell to understand the existing code structure:

✓ Read q4.ipynb, lines 250 to 272

Now let me read more context to understand the existing GMM implementation and helper functions:

✓ Read q4.ipynb, lines 150 to 200



```
import numpy as np

def create_video_with_panels(img, output_filename, frame_duration=0.75):
    """
    Create a video with three panels:
    - Left: Original image
    - Middle: Segmented image (evolving)
    - Right: Likelihood graph (growing)
    """
    # Get segmented images and likelihoods
    df = convert_to_df(img)
    x = df[['R','G','B']]
    gmm_obj = GMM(x, n_clusters=3, max_iters=10)
    gmm_obj.fit()

    all_time_resp = gmm_obj.resp_all_time
    likelihoods = gmm_obj.likelihood

    # Generate segmented images for each iteration
    segmented_images = []
    for resp in all_time_resp:
        seg_img = get_segmented_image(img, resp, n_clusters=3)
        segmented_images.append(seg_img)

    # Convert original image to numpy array
    original_img = np.array(img)

    # Set up the figure with three subplots
    fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(15, 5))
    fig.suptitle('GMM Image Segmentation Evolution', fontsize=16)

    # Display original image (static)
    ax1.imshow(original_img)
    ax1.set_title('Original Image')
    ax1.axis('off')

    # Initialize middle panel for segmented image
    ax2.set_title('Segmented Image')
    ax2.axis('off')
    seg_display = ax2.imshow(segmented_images[0])

    # Initialize right panel for likelihood plot
    ax3.set_title('Log-Likelihood Evolution')
    ax3.set_xlabel('Iteration')
    ax3.set_ylabel('Log-Likelihood')
    line, = ax3.plot([], [], 'b-', linewidth=2)
```

Better visualization for this - https://g.co/gemini/share/e0d51758f7e7

Q5

1. Understanding basics and issues - https://chatgpt.com/share/68d79fb6-5328-8001-af1e-e7c59b1d2ef0

Q7

1. Getting the code for the logic I had - https://g.co/gemini/share/05558da52f1b
   https://chatgpt.com/share/68d818ab-8780-800f-90f9-92fe9f336d0c