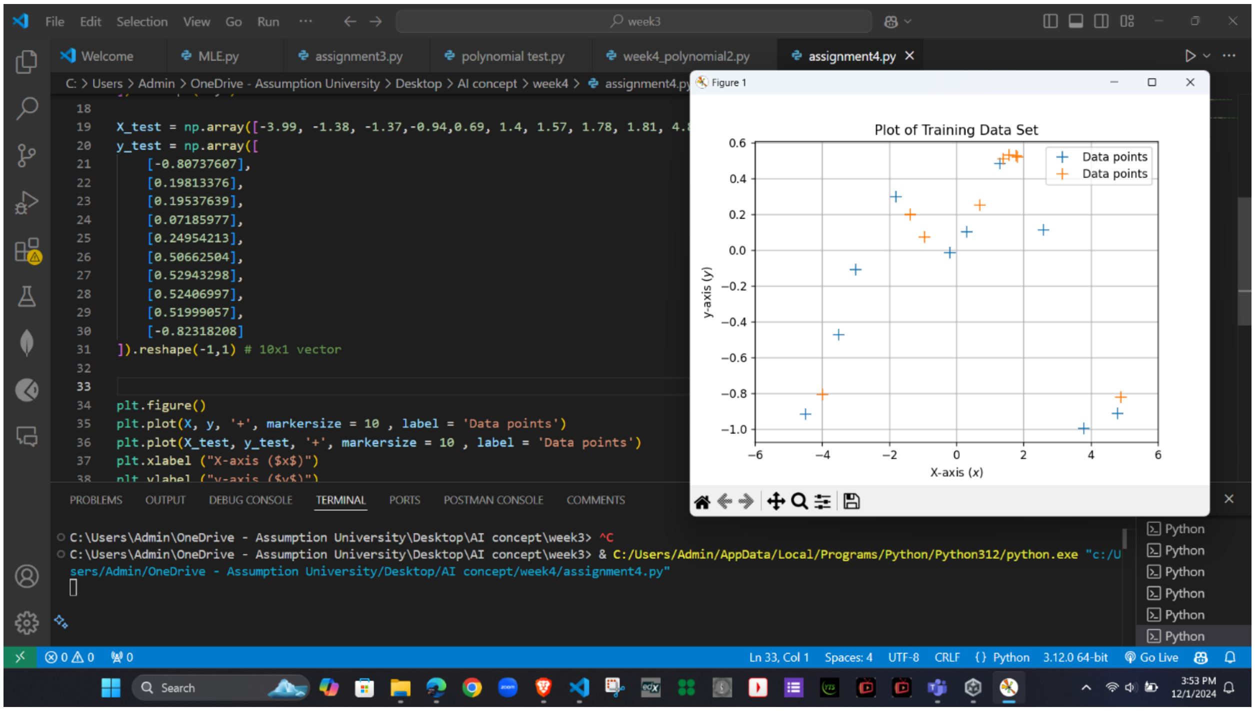
Assignment 4

Q1. Maximum Likelihood Estimation with Features

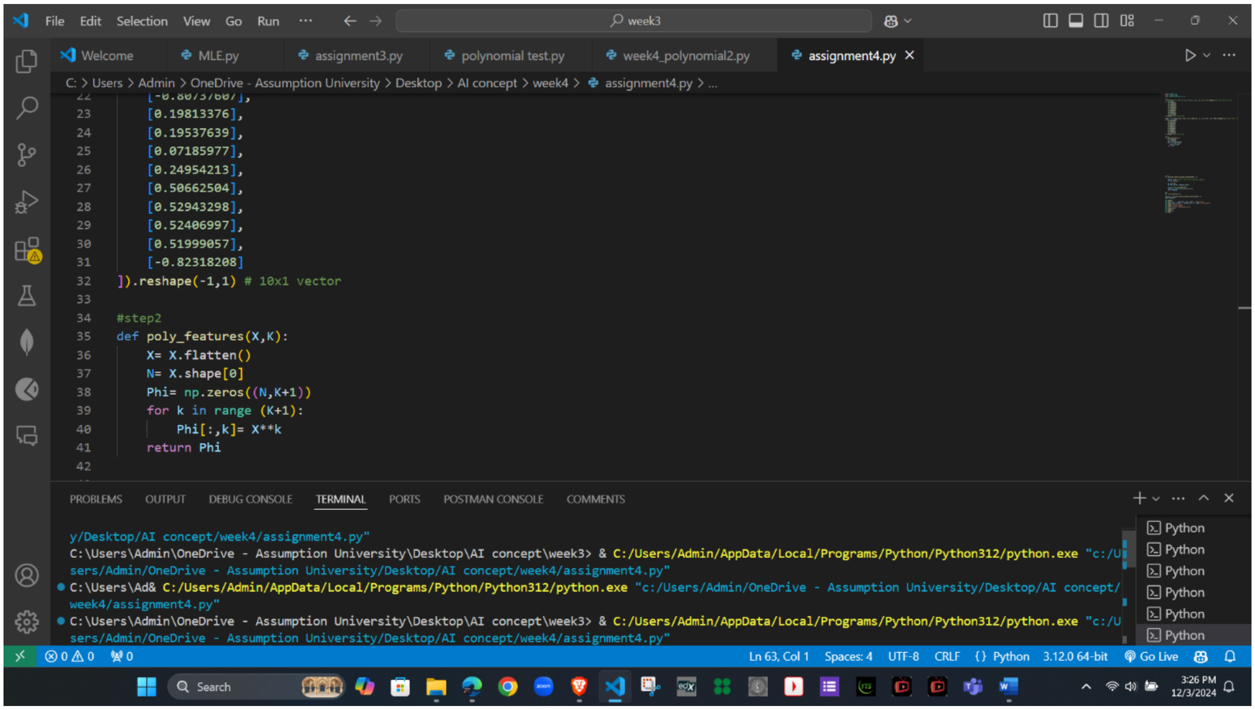
Step 1: Understanding and Plotting the Data



```python

X_test = np.array([-3.99, -1.38, -1.37,-0.94,0.69, 1.4, 1.57, 1.78, 1.81, 4.
y_test = np.array([
    [-0.80737607],
    [0.19813376],
    [0.19537639],
    [0.07185977],
    [0.24954213],
    [0.50662504],
    [0.52943298],
    [0.52406997],
    [0.51999057],
    [-0.82318208]
]).reshape(-1,1) # 10x1 vector



plt.figure()
plt.plot(X, y, '+', markersize = 10 , label = 'Data points')
plt.plot(X_test, y_test, '+', markersize = 10 , label = 'Data points')
plt.xlabel ("X-axis ($x$)")
plt.ylabel ("y-axis ($y$)")
```

```
○ C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3> ^C
○ C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "c:/U
sers/Admin/OneDrive - Assumption University/Desktop/AI concept/week4/assignment4.py"
```

Step 2: Polynomial Feature Transformation



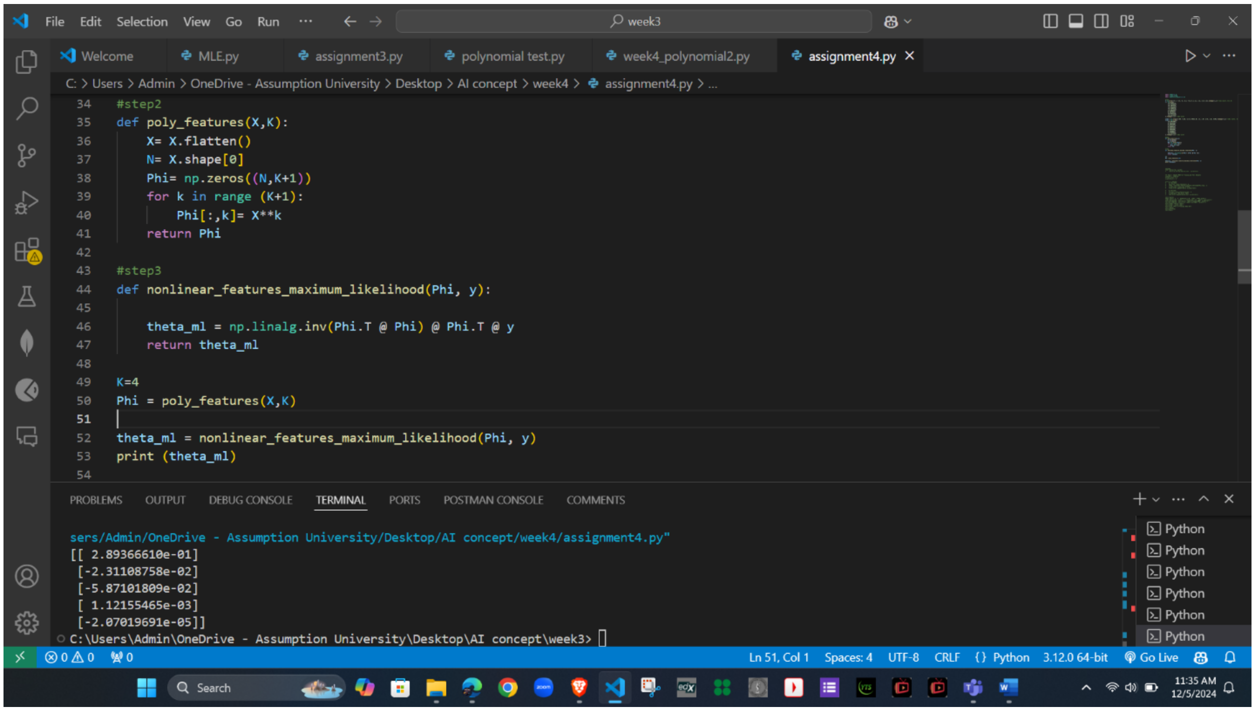
```python
    [-0.80737607],
    [0.19813376],
    [0.19537639],
    [0.07185977],
    [0.24954213],
    [0.50662504],
    [0.52943298],
    [0.52406997],
    [0.51999057],
    [-0.82318208]
]).reshape(-1,1) # 10x1 vector

#step2
def poly_features(X,K):
    X= X.flatten()
    N= X.shape[0]
    Phi= np.zeros((N,K+1))
    for k in range (K+1):
        Phi[:,k]= X**k
    return Phi

```

```
y/Desktop/AI concept/week4/assignment4.py"
C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "c:/U
sers/Admin/OneDrive - Assumption University/Desktop/AI concept/week4/assignment4.py"
● C:\Users\Ad& C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/Admin/OneDrive - Assumption University/Desktop/AI concept/
week4/assignment4.py"
● C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "c:/U
sers/Admin/OneDrive - Assumption University/Desktop/AI concept/week4/assignment4.py"
```

The purpose and the effect of this transformation:

Polynomial transformation enables linear models to fit nonlinear data by adding higher-degree features. It improves flexibility but risks underfitting with low degrees or overfitting with high degrees. The right degree balances accuracy and complexity for better generalization.

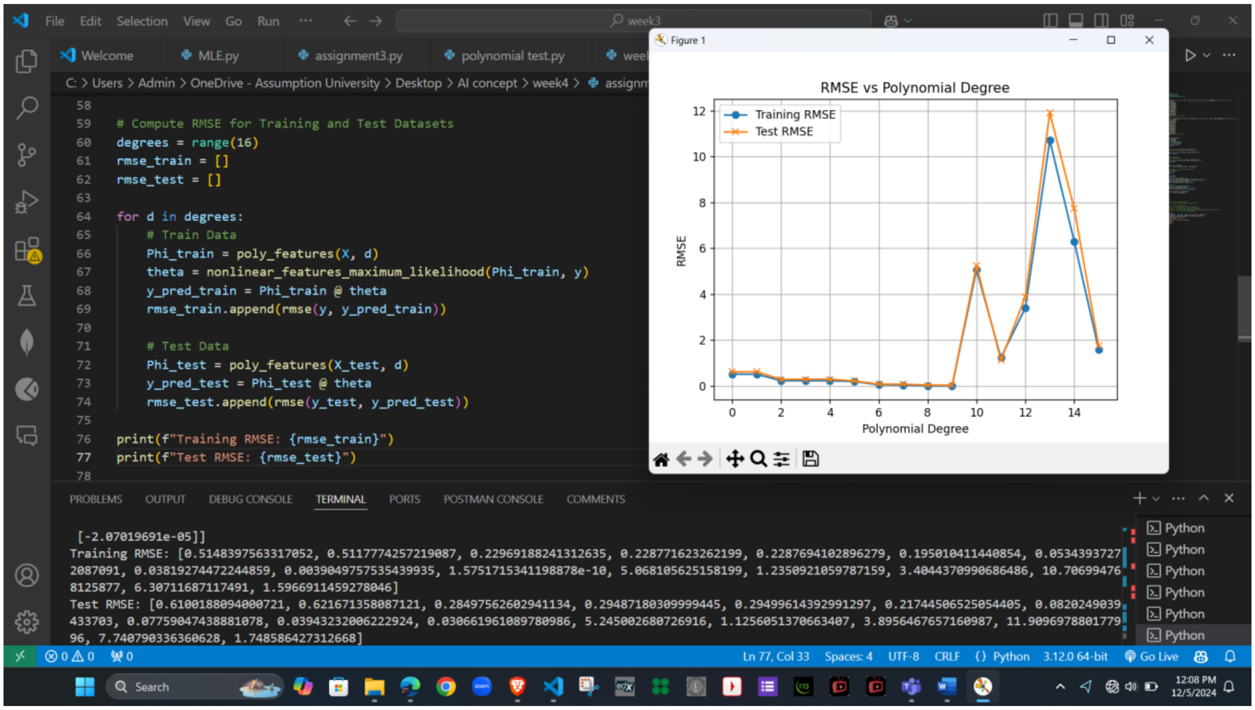
Step 3: Fitting the Model Using Maximum Likelihood


File Edit Selection View Go Run
week3
Welcome
MLE.py
assignment3.py
polynomial test.py
week4_polynomial2.py
assignment4.py
C: > Users > Admin > OneDrive - Assumption University > Desktop > AI concept > week4 > assignment4.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE COMMENTS
Python
Python
Python
Python
Python
Python
Ln 51, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.0 64-bit Go Live
Search
11:35 AM
12/5/2024

```python
#step2
def poly_features(X,K):
    X= X.flatten()
    N= X.shape[0]
    Phi= np.zeros((N,K+1))
    for k in range (K+1):
        Phi[:,k]= X**k
    return Phi

#step3
def nonlinear_features_maximum_likelihood(Phi, y):

    theta_ml = np.linalg.inv(Phi.T @ Phi) @ Phi.T @ y
    return theta_ml

K=4
Phi = poly_features(X,K)

theta_ml = nonlinear_features_maximum_likelihood(Phi, y)
print (theta_ml)

```

```
sers/Admin/OneDrive - Assumption University/Desktop/AI concept/week4/assignment4.py"
[[ 2.89366610e-01]
 [-2.31108758e-02]
 [-5.87101809e-02]
 [ 1.12155465e-03]
 [-2.07019691e-05]]
C:\Users\Admin\OneDrive - Assumption University\Desktop\AI concept\week3>
```

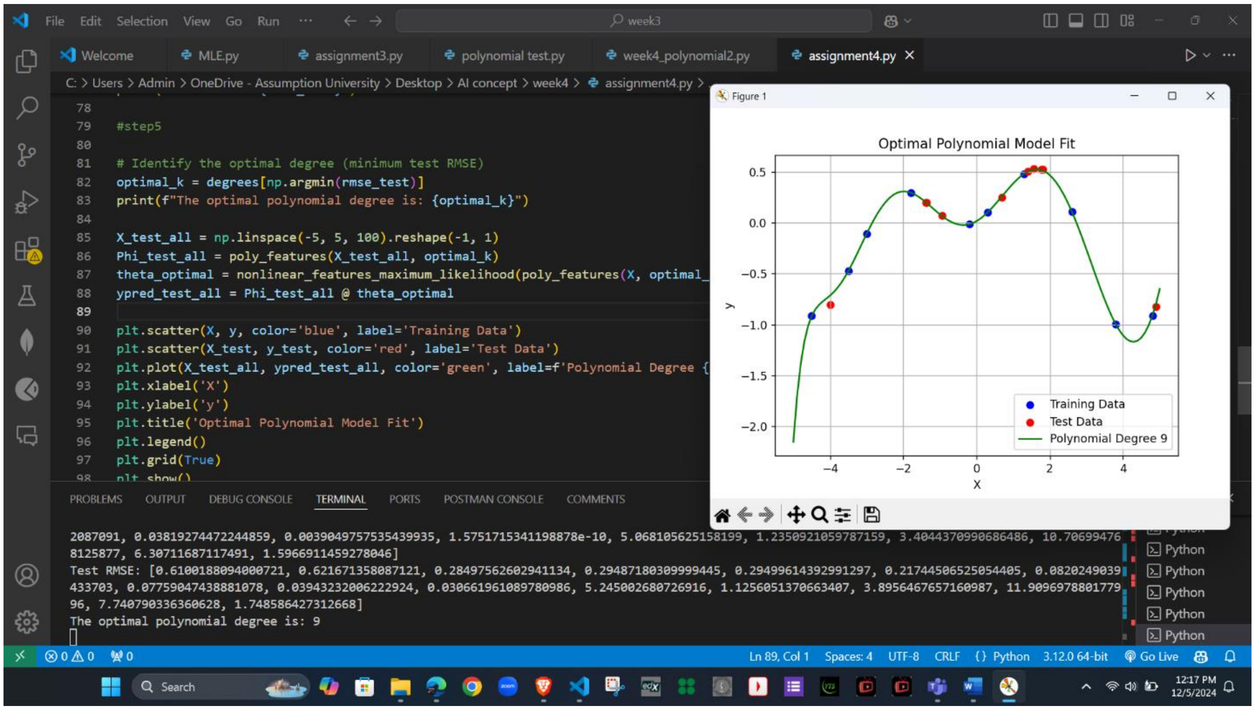
Step 4: Model Evaluation Using RMSE



```python

# Compute RMSE for Training and Test Datasets
degrees = range(16)
rmse_train = []
rmse_test = []

for d in degrees:
    # Train Data
    Phi_train = poly_features(X, d)
    theta = nonlinear_features_maximum_likelihood(Phi_train, y)
    y_pred_train = Phi_train @ theta
    rmse_train.append(rmse(y, y_pred_train))

    # Test Data
    Phi_test = poly_features(X_test, d)
    y_pred_test = Phi_test @ theta
    rmse_test.append(rmse(y_test, y_pred_test))

print(f"Training RMSE: {rmse_train}")
print(f"Test RMSE: {rmse_test}")

```

```
 [-2.07019691e-05]]
Training RMSE: [0.5148397563317052, 0.5117774257219087, 0.22969188241312635, 0.228771623262199, 0.2287694102896279, 0.195010411440854, 0.0534393727
2087091, 0.03819274472244859, 0.0039049757535439935, 1.5751715341198878e-10, 5.068105625158199, 1.2350921059787159, 3.4044370990686486, 10.70699476
8125877, 6.30711687117491, 1.5966911459278046]
Test RMSE: [0.6100188094000721, 0.621671358087121, 0.28497562602941134, 0.29487180309999445, 0.29499614392991297, 0.21744506525054405, 0.0820249039
433703, 0.07759047438881078, 0.03943232006222924, 0.030661961089780986, 5.245002680726916, 1.1256051370663407, 3.8956467657160987, 11.9096978801779
96, 7.740790336360628, 1.748586427312668]
```

Step 5: Selecting the Best Model



File Edit Selection View Go Run
week3
Welcome
MLE.py
assignment3.py
polynomial test.py
week4_polynomial2.py
assignment4.py
C: > Users > Admin > OneDrive - Assumption University > Desktop > AI concept > week4 > assignment4.py >
Figure 1
Optimal Polynomial Model Fit
0.5
0.0
−0.5
−1.0
−1.5
−2.0
y
−4
−2
0
2
4
X
● Training Data
● Test Data
Polynomial Degree 9
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE COMMENTS
Python
Ln 89, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.12.0 64-bit Go Live
Search
12:17 PM
12/5/2024

```
#step5

# Identify the optimal degree (minimum test RMSE)
optimal_k = degrees[np.argmin(rmse_test)]
print(f"The optimal polynomial degree is: {optimal_k}")

X_test_all = np.linspace(-5, 5, 100).reshape(-1, 1)
Phi_test_all = poly_features(X_test_all, optimal_k)
theta_optimal = nonlinear_features_maximum_likelihood(poly_features(X, optimal_
ypred_test_all = Phi_test_all @ theta_optimal

plt.scatter(X, y, color='blue', label='Training Data')
plt.scatter(X_test, y_test, color='red', label='Test Data')
plt.plot(X_test_all, ypred_test_all, color='green', label=f'Polynomial Degree {
plt.xlabel('X')
plt.ylabel('y')
plt.title('Optimal Polynomial Model Fit')
plt.legend()
plt.grid(True)
plt.show()
```

```
2087091, 0.03819274472244859, 0.0039049757535439935, 1.5751715341198878e-10, 5.068105625158199, 1.2350921059787159, 3.4044370990686486, 10.70699476
8125877, 6.30711687117491, 1.5966911459278046]
Test RMSE: [0.6100188094000721, 0.621671358087121, 0.28497562602941134, 0.29487180309999445, 0.29499614392991297, 0.21744506525054405, 0.0820249039
433703, 0.07759047438881078, 0.03943232006222924, 0.030661961089780986, 5.245002680726916, 1.1256051370663407, 3.8956467657160987, 11.9096978801779
96, 7.740790336360628, 1.748586427312668]
The optimal polynomial degree is: 9
```

Why was this degree optimal?

- The optimal degree minimizes the RMSE for the test dataset, striking a balance between underfitting (too simple) and overfitting (too complex).
- Degrees that are too low (e.g., 0 or 1) underfit, being too simple and cannot catch the small details in the data, so they don’t fit well.
- Degrees that are too high (e.g., >10) overfit, capturing noise rather than the true relationship.

Model Complexity and Overfitting/Underfitting:

- Lower degrees: Underfitting, as the model cannot capture complex patterns.
- Higher degrees: Overfitting, as the model tries to fit every data point, including noise.
- Optimal degree: Provides the best generalization for unseen data (test set).

Step 6: Conclusion

Summary of Findings:

- The polynomial regression model's performance depends heavily on the degree of the polynomial.
- Low-degree models (e.g., 0 or 1): These models were too simple and underfit the data, missing important patterns and resulting in high training and test RMSE.
- High-degree models (e.g., >10): These models overfit the training data, capturing noise instead of meaningful trends, which led to high test RMSE.
- The optimal degree was the one that minimized test RMSE, achieving a balance between underfitting and overfitting.

Trade-offs Between Model Complexity and Prediction Accuracy:

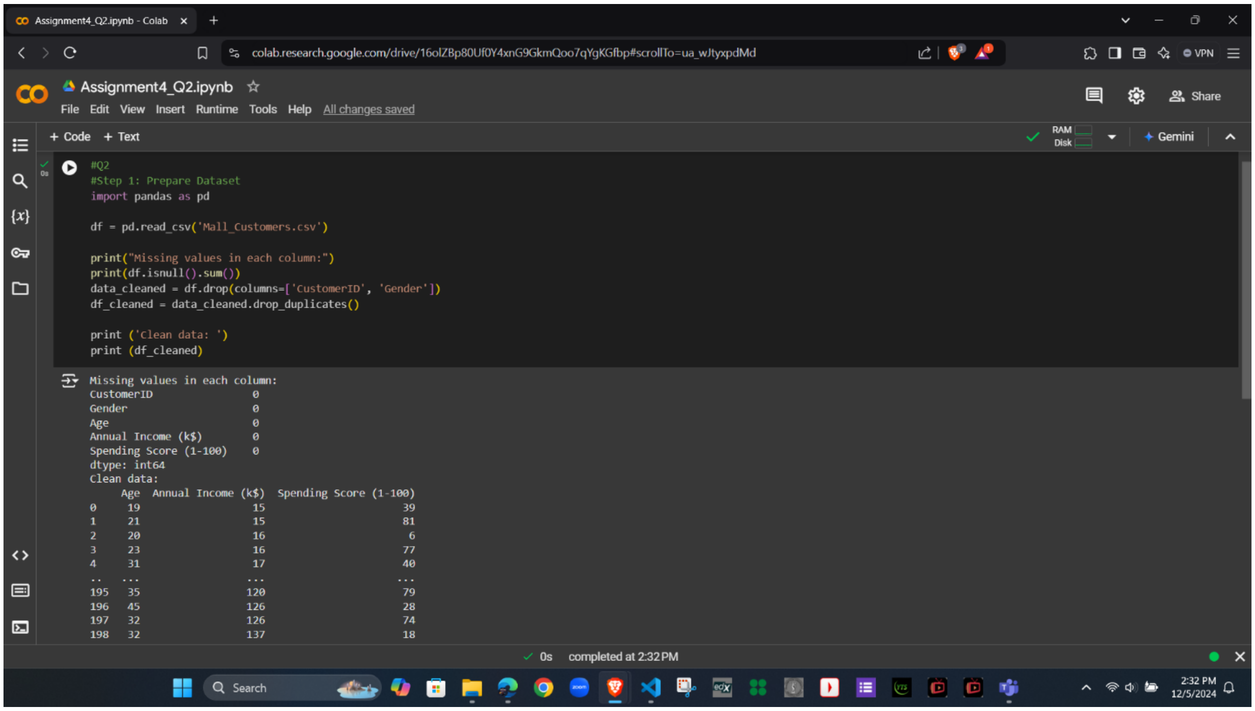
- Low Complexity (Low Degree):
    - Advantages: Simple, easier to interpret, and less likely to overfit.
    - Disadvantages: Poor performance if the relationship between variables is complex (underfitting).
- High Complexity (High Degree):
    - Advantages: Can model complex relationships in the data and fit the training data very well.

  - Disadvantages: Risk of overfitting, where the model captures noise instead of general patterns, leading to poor test performance.
- Balanced Complexity (Optimal Degree):
  - The optimal degree balances the trade-off, capturing the true patterns in the data without fitting the noise, resulting in better generalization to unseen data.

Q2. K-means

Step 1: Prepare Dataset

- Clean the unnecessary data, duplicate data and missing value

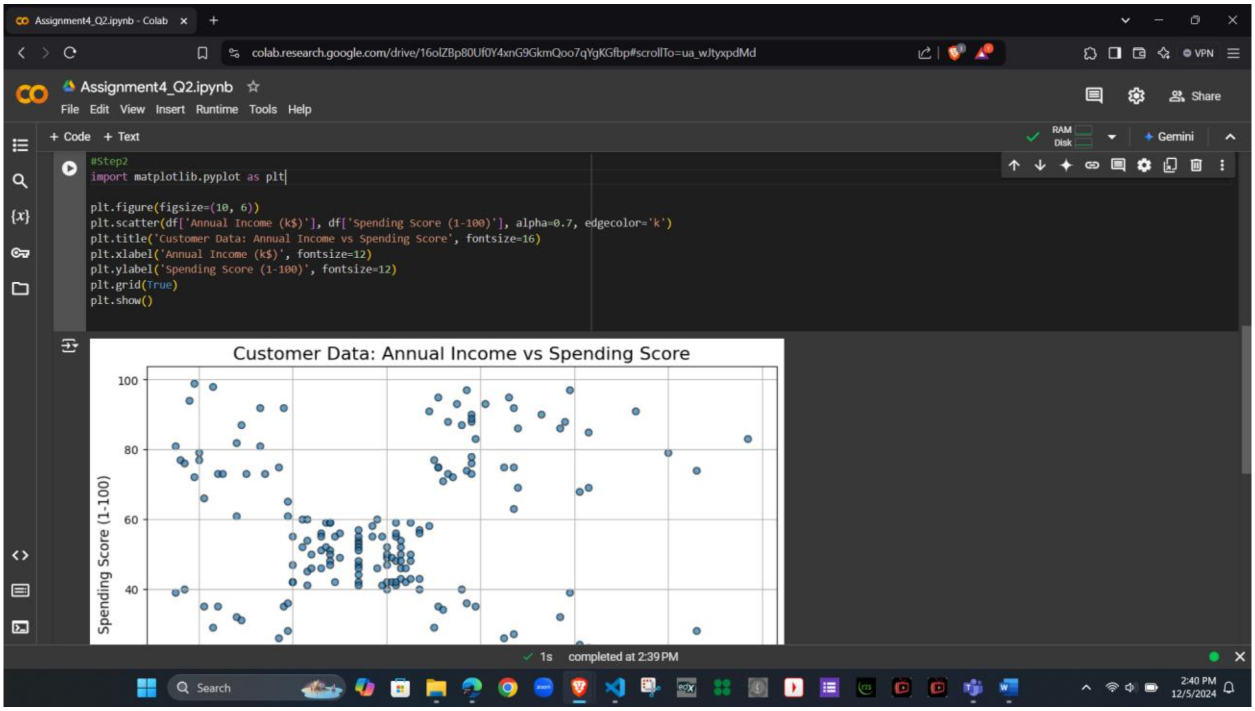


Assignment4_Q2.ipynb - Colab
colab.research.google.com/drive/16olZBp80Uf0Y4xnG9GkmQoo7qYgKGfbp#scrollTo=ua_wJtyxpdMd
Assignment4_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM
Disk
Gemini
0s completed at 2:32 PM
Search
2:32 PM
12/5/2024

```
#Q2
#Step 1: Prepare Dataset
import pandas as pd

df = pd.read_csv('Mall_Customers.csv')

print("Missing values in each column:")
print(df.isnull().sum())
data_cleaned = df.drop(columns=['CustomerID', 'Gender'])
df_cleaned = data_cleaned.drop_duplicates()

print ('Clean data: ')
print (df_cleaned)
```

```
Missing values in each column:
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
Clean data:
     Age  Annual Income (k$)  Spending Score (1-100)
0     19                  15                      39
1     21                  15                      81
2     20                  16                       6
3     23                  16                      77
4     31                  17                      40
..   ...                 ...                     ...
195   35                 120                      79
196   45                 126                      28
197   32                 126                      74
198   32                 137                      18
```

Step 2: Visualize Data Before Clustering

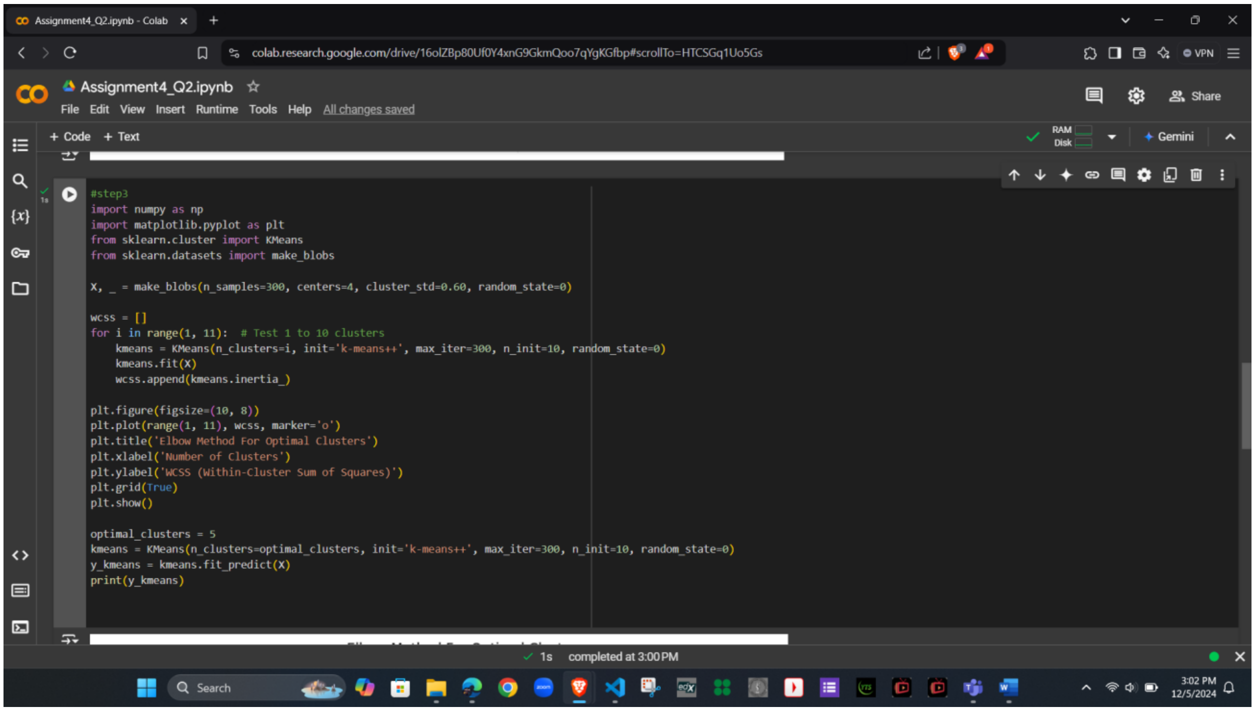


Assignment4_Q2.ipynb - Colab
colab.research.google.com/drive/16olZBp80Uf0Y4xnG9GkmQoo7qYgKGfbp#scrollTo=ua_wJtyxpdMd
Assignment4_Q2.ipynb
File Edit View Insert Runtime Tools Help
Share
+ Code + Text
RAM
Disk
Gemini
Customer Data: Annual Income vs Spending Score
Spending Score (1-100)
100
80
60
40
1s completed at 2:39 PM
Search
2:40 PM
12/5/2024

```
#Step2
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], alpha=0.7, edgecolor='k')
plt.title('Customer Data: Annual Income vs Spending Score', fontsize=16)
plt.xlabel('Annual Income (k$)', fontsize=12)
plt.ylabel('Spending Score (1-100)', fontsize=12)
plt.grid(True)
plt.show()
```

Step 3: Implement K-means Clustering and Determine Optimal Clusters

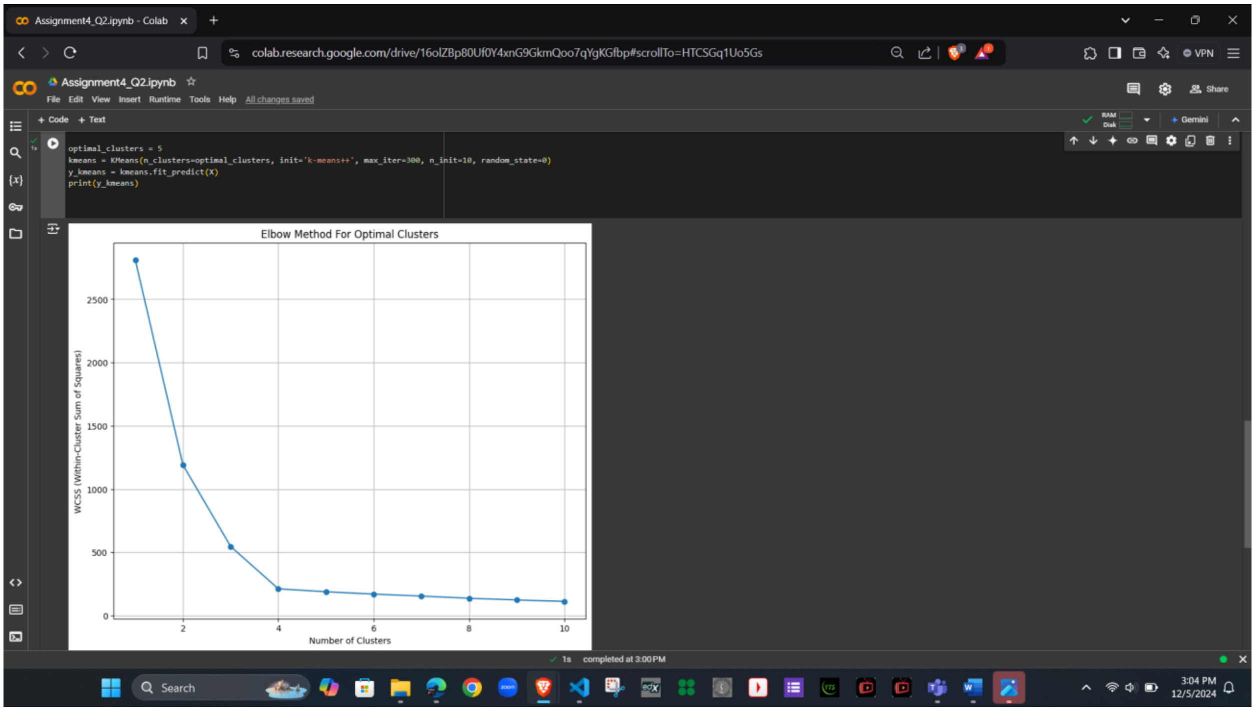
- Code for step3 using Elbow Method Code



Assignment4_Q2.ipynb - Colab
colab.research.google.com/drive/16olZBp80Uf0Y4xnG9GkmQoo7qYgKGfbp#scrollTo=HTCSGq1Uo5Gs
Assignment4_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
+ Code + Text
RAM
Disk
Gemini
✓ 1s completed at 3:00 PM
Search
3:02 PM
12/5/2024

```
#step3
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs

X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

wcss = []
for i in range(1, 11):  # Test 1 to 10 clusters
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 8))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method For Optimal Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS (Within-Cluster Sum of Squares)')
plt.grid(True)
plt.show()

optimal_clusters = 5
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++', max_iter=300, n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X)
print(y_kmeans)
```

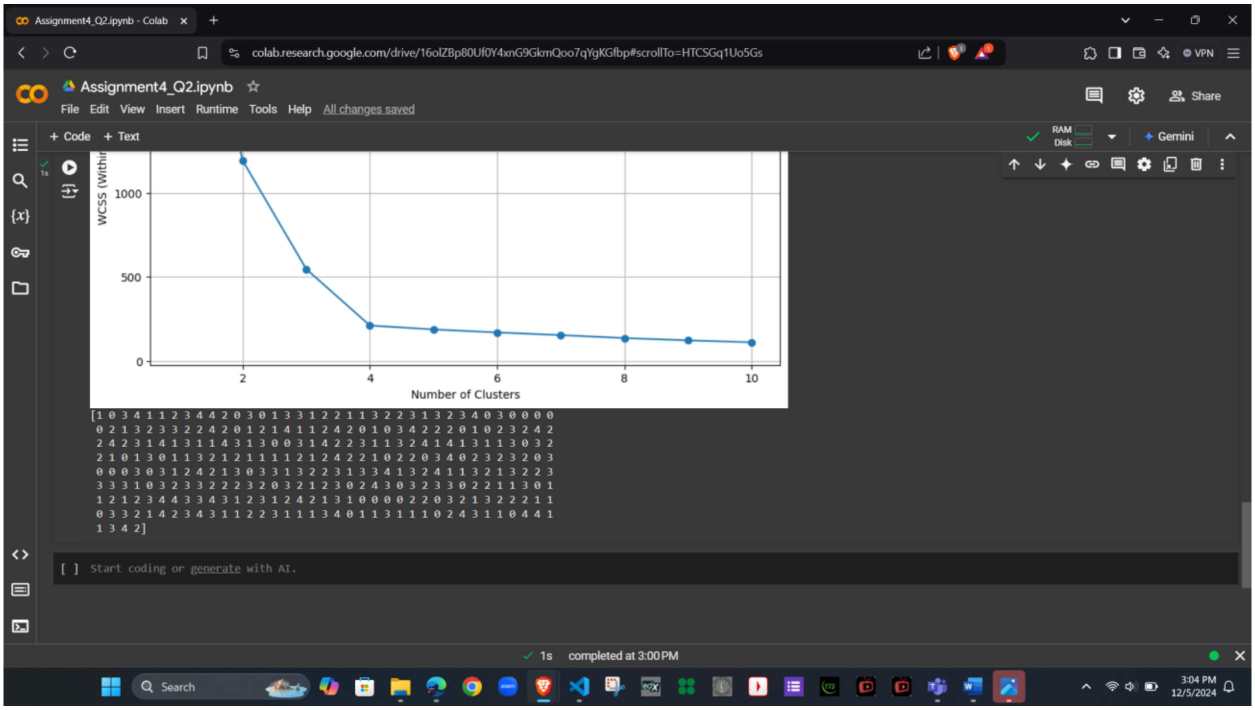
- Plot the Elbow Method Graph



```
optimal_clusters = 5
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++', max_iter=300, n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X)
print(y_kmeans)
```

Assignment4_Q2.ipynb - Colab
colab.research.google.com/drive/16olZBp80Uf0Y4xnG9GkmQoo7qYgKGfbp#scrollTo=HTCSGq1Uo5Gs
Assignment4_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Share
Gemini
Elbow Method For Optimal Clusters
WCSS (Within-Cluster Sum of Squares)
2500
2000
1500
1000
500
0
2
4
6
8
10
Number of Clusters
1s completed at 3:00 PM
Search
3:04 PM
12/5/2024

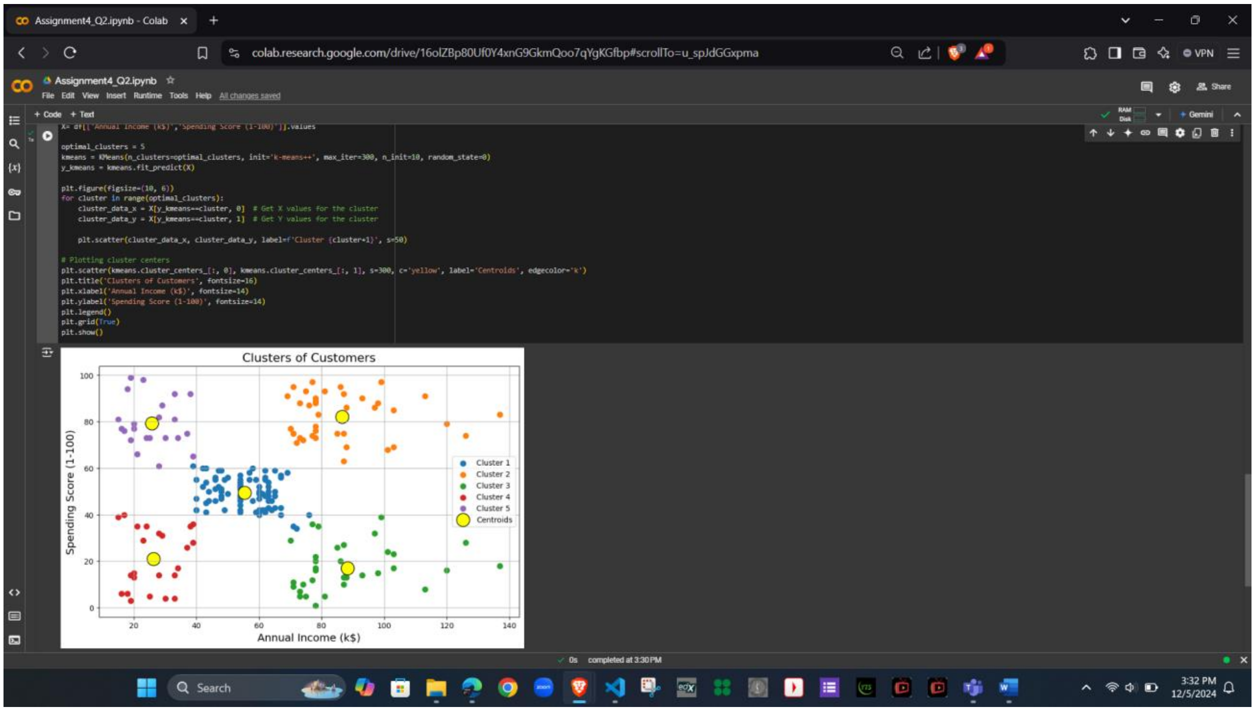
- K-mean output



```
[1 0 3 4 1 1 2 3 4 4 2 0 3 0 1 3 3 1 2 2 1 1 3 2 2 3 1 3 2 3 4 0 3 0 0 0 0
 0 2 1 3 2 3 3 2 2 4 2 0 1 2 1 4 1 1 2 4 2 0 1 0 3 4 2 2 2 0 1 0 2 3 2 4 2
 2 4 2 3 1 4 1 3 1 1 4 3 1 3 0 0 3 1 4 2 2 3 1 1 3 2 4 1 4 1 3 1 1 3 0 3 2
 2 1 0 1 3 0 1 1 3 2 1 2 1 1 1 1 2 1 2 4 2 2 1 0 2 2 0 3 4 0 2 3 2 3 2 0 3
 0 0 0 3 0 3 1 2 4 2 1 3 0 3 3 1 3 2 2 3 1 3 3 4 1 3 2 4 1 1 3 2 1 3 2 2 3
 3 3 3 1 0 3 2 3 3 2 2 2 3 2 0 3 2 1 2 3 0 2 4 3 0 3 2 3 3 0 2 2 1 1 3 0 1
 1 2 1 2 3 4 4 3 3 4 3 1 2 3 1 2 4 2 1 3 1 0 0 0 0 2 2 0 3 2 1 3 2 2 2 1 1
 0 3 3 2 1 4 2 3 4 3 1 1 2 2 3 1 1 1 3 4 0 1 1 3 1 1 1 0 2 4 3 1 1 0 4 4 1
 1 3 4 2]
```

Assignment4_Q2.ipynb - Colab
colab.research.google.com/drive/16olZBp80Uf0Y4xnG9GkmQoo7qYgKGfbp#scrollTo=HTCSGq1Uo5Gs
Assignment4_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Share
Gemini
WCSS (Withi
1000
500
0
2
4
6
8
10
Number of Clusters
[ ] Start coding or generate with AI.
1s completed at 3:00 PM
Search
3:04 PM
12/5/2024

Step 4: Visualize Data After Clustering



```
X= df[['Annual Income (k$)','Spending Score (1-100)']].values

optimal_clusters = 5
kmeans = KMeans(n_clusters=optimal_clusters, init='k-means++', max_iter=300, n_init=10, random_state=0)
y_kmeans = kmeans.fit_predict(X)

plt.figure(figsize=(10, 6))
for cluster in range(optimal_clusters):
    cluster_data_x = X[y_kmeans==cluster, 0]  # Get X values for the cluster
    cluster_data_y = X[y_kmeans==cluster, 1]  # Get Y values for the cluster

    plt.scatter(cluster_data_x, cluster_data_y, label=f'Cluster {cluster+1}', s=50)

# Plotting cluster centers
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='yellow', label='Centroids', edgecolor='k')
plt.title('Clusters of Customers', fontsize=16)
plt.xlabel('Annual Income (k$)', fontsize=14)
plt.ylabel('Spending Score (1-100)', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

Step 5: Interpret Your Results

The clustering results provide insights into customer segmentation based on Annual Income (k$) and Spending Score (1-100). Here’s a detailed breakdown of each cluster:

Cluster Sizes:

- Cluster 0 (81 customers): Largest group with moderate income and spending behavior.
- Cluster 1 (39 customers): High-income, high-spending customers (premium segment).
- Cluster 2 (35 customers): High-income, low-spending customers (potential for engagement).
- Cluster 3 (23 customers): Low-income, low-spending customers (limited purchasing power).
- Cluster 4 (22 customers): Low-income, high-spending customers (price-sensitive but engaged).

Cluster Statistics Analysis:

1. Cluster 0 (Moderate Income, Moderate Spending):
    - Average Annual Income: ~$55.3k
    - Average Spending Score: ~49.5
    - Represents customers with balanced spending habits, likely mid-tier or average-value customers.
    - Opportunity: Maintain their satisfaction with general promotions or loyalty programs.
2. Cluster 1 (High Income, High Spending):

   - Average Annual Income: ~$86.5k
   - Average Spending Score: ~82.1
   - High-value customers who spend a lot and likely purchase premium products/services.
   - Opportunity: Strengthen loyalty through exclusive rewards, VIP programs, and personalized experiences.

3. Cluster 2 (High Income, Low Spending):
   - Average Annual Income: ~$88.2k
   - Average Spending Score: ~17.1
   - High-income customers with low engagement or spending behavior.
   - Opportunity: Explore why these customers are not spending much (e.g., mismatched offerings, lack of interest). Use targeted marketing to improve engagement.
4. Cluster 3 (Low Income, Low Spending):
   - Average Annual Income: ~$26.3k
   - Average Spending Score: ~20.9
   - Budget-conscious customers with low purchasing power.
   - Opportunity: Offer affordable options, discounts, or budget-friendly product lines to attract and retain these customers.
5. Cluster 4 (Low Income, High Spending):

- Average Annual Income: ~$25.7k
- Average Spending Score: ~79.4
- Price-sensitive customers who are highly engaged despite having limited income.
- Opportunity: Provide value-based products and loyalty programs to encourage frequent purchases.

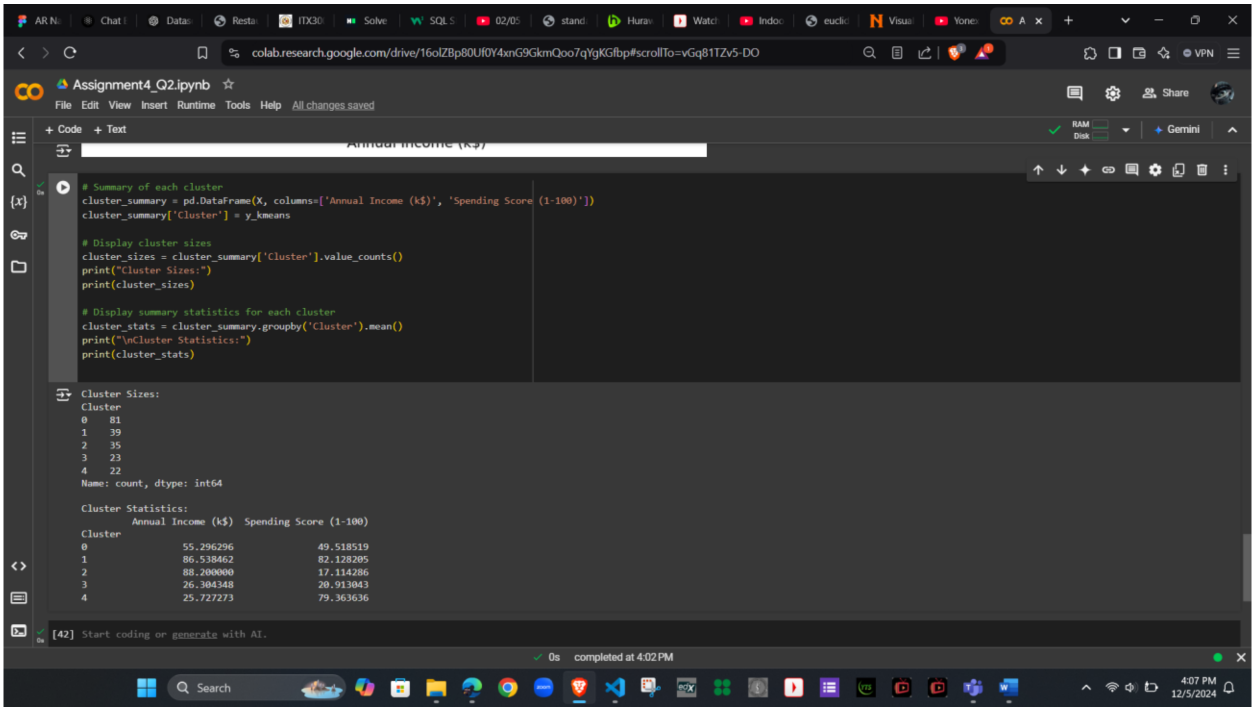
How This Information Helps Identify Specific Groups:

1. Targeted Marketing Strategies:
    - Cluster 1: Focus on retaining high-value customers through personalized and exclusive offers.
    - Cluster 2: Design campaigns to convert high-income, low-spending customers into active spenders.
    - Cluster 4: Develop affordable product bundles or discounts to maximize the engagement of price-sensitive customers.
2. Customer Retention:
    - Enhance customer retention in Cluster 0 through loyalty programs that reward consistent spending.
    - Identify disengagement reasons in Cluster 2 and address them with surveys or tailored incentives.
3. Product and Service Design:
    - Offer premium products and services to Cluster 1.

   - Introduce budget-friendly options for Cluster 3 and Cluster 4.

4. Resource Allocation:
   - Allocate more resources to engage and retain high-potential customers in Clusters 1, 2, and 4.
   - Focus less on Cluster 3, as they represent a low-spending group with limited income.
5. Long-Term Strategies:
   - Monitor spending trends in Cluster 2 to identify signs of increased engagement.
   - Encourage upselling and cross-selling to moderate spenders in Cluster 0.

Additional code


colab.research.google.com/drive/16oIZBp80Uf0Y4xnG9GkmQoo7qYgKGfbp#scrollTo=vGq81TZv5-DO
Assignment4_Q2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Share
Gemini
completed at 4:02 PM
4:07 PM
12/5/2024

```
# Summary of each cluster
cluster_summary = pd.DataFrame(X, columns=['Annual Income (k$)', 'Spending Score (1-100)'])
cluster_summary['Cluster'] = y_kmeans

# Display cluster sizes
cluster_sizes = cluster_summary['Cluster'].value_counts()
print("Cluster Sizes:")
print(cluster_sizes)

# Display summary statistics for each cluster
cluster_stats = cluster_summary.groupby('Cluster').mean()
print("\nCluster Statistics:")
print(cluster_stats)
```

```
Cluster Sizes:
Cluster
0    81
1    39
2    35
3    23
4    22
Name: count, dtype: int64

Cluster Statistics:
         Annual Income (k$)  Spending Score (1-100)
Cluster
0                 55.296296               49.518519
1                 86.538462               82.128205
2                 88.200000               17.114286
3                 26.304348               20.913043
4                 25.727273               79.363636
```

```
[42] Start coding or generate with AI.
```