Python for Computer Science and Data Science 2 (CSE 3652)

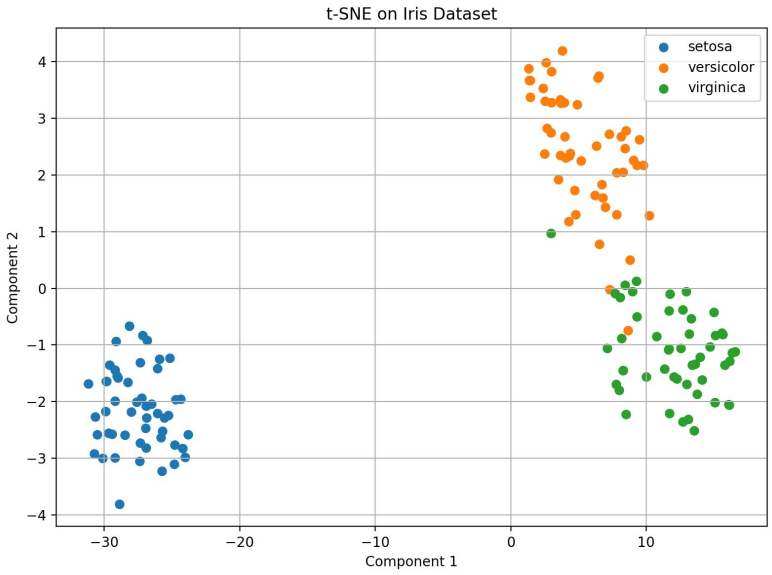
MINOR ASSIGNMENT-4: MACHINE LEARNING- CLASSIFICATION, REGRESSION AND CLUSTERING

1. Perform dimensionality reduction using scikit-learn's TSNE estimator on the Iris dataset, then graph the results.

Ans:-

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.manifold import TSNE
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X)
plt.figure(figsize=(8, 6))
for i, target_name in enumerate(target_names):
    plt.scatter(X_tsne[y == i, 0], X_tsne[y == i, 1], label=target_name)
plt.title("t-SNE on Iris Dataset")
plt.xlabel("Component 1")
plt.ylabel("Component 2")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output: -

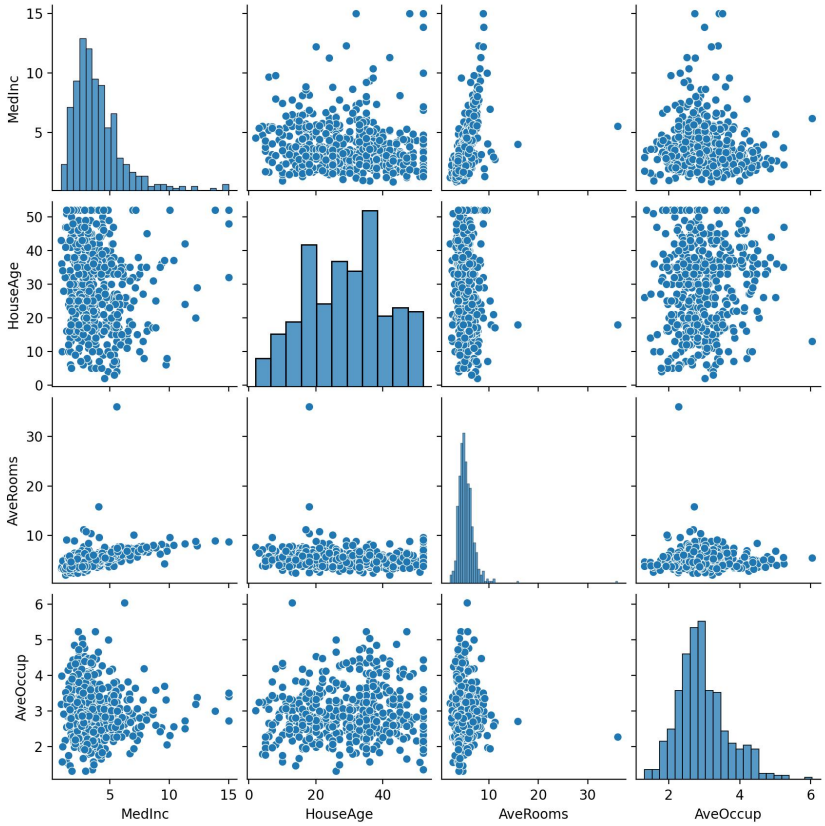


2. Create a Seaborn pairplot graph for the California Housing dataset. Try the Matplotlib features to panning and zoom in on the diagram. These are accessible via the icons in the Matplotlib window.

Ans:-

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
import pandas as pd
housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df_sampled = df.sample(500, random_state=42)
sns.pairplot(df_sampled[['MedInc', 'HouseAge', 'AveRooms', 'AveOccup']])
plt.suptitle("Pairplot of California Housing Features", y=1.02)
plt.show()
```

Output: -



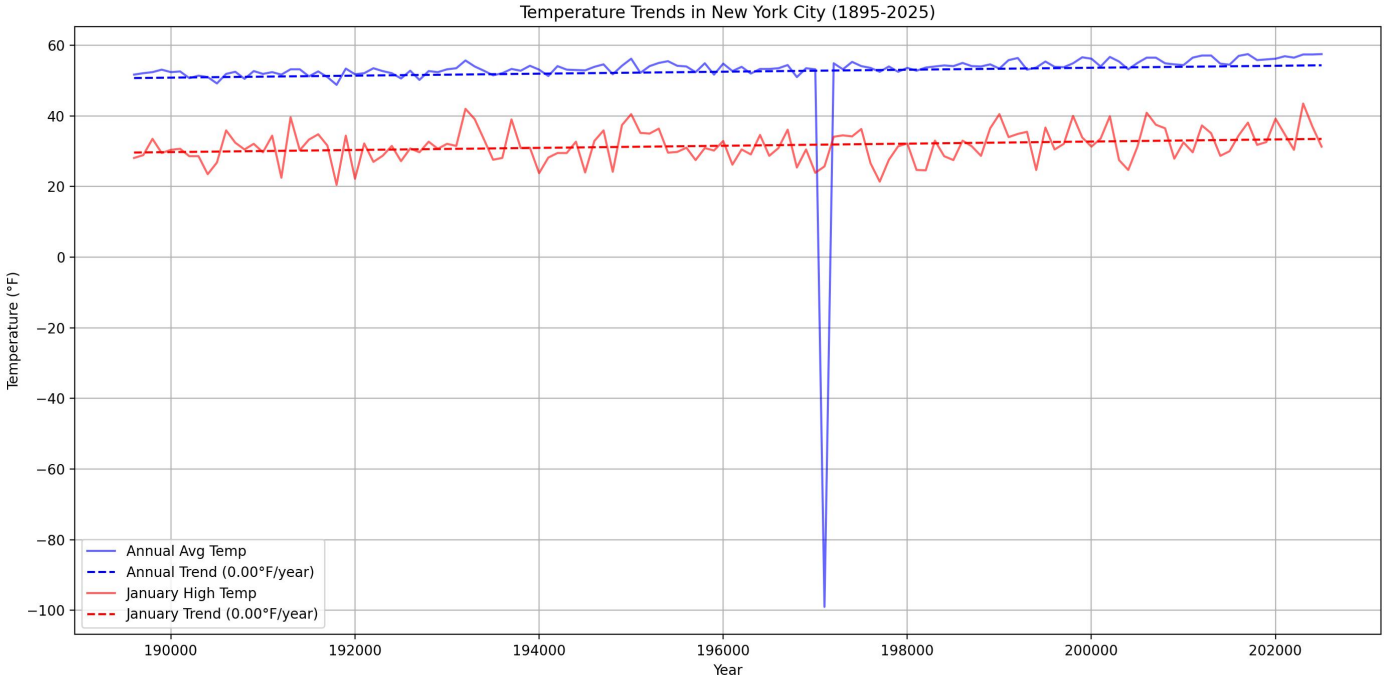
3. Go to NOAA's Climate at a Glance page (Link) and download the available time series data for the average annual temperatures of New York City from 1895 to today (1895-2025). Implement simple linear regression using average annual temperature data. Also, show how does the temperature trend compare to the average January high temperatures?.

Ans:-

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
annual_df = pd.read_csv('annual_avg_temp.csv', skiprows=4)
january_df = pd.read_csv('january_high_temp.csv', skiprows=4)
annual_df.columns = ['Year', 'AnnualAvgTemp']
january_df.columns = ['Year', 'JanuaryHighTemp']
merged_df = pd.merge(annual_df, january_df, on='Year')
X = merged_df['Year'].values.reshape(-1, 1)
y_annual = merged_df['AnnualAvgTemp'].values
y_january = merged_df['JanuaryHighTemp'].values
model_annual = LinearRegression().fit(X, y_annual)
model_january = LinearRegression().fit(X, y_january)
annual_pred = model_annual.predict(X)
january_pred = model_january.predict(X)
annual_trend = model_annual.coef_[0]
january_trend = model_january.coef_[0]
plt.figure(figsize=(14, 7))
plt.plot(merged_df['Year'], y_annual, label='Annual Avg Temp', color='blue', alpha=0.6)
plt.plot(merged_df['Year'], annual_pred, label=f'Annual Trend ({annual_trend:.2f}°F/year)',
        color='blue', linestyle='--')
plt.plot(merged_df['Year'], y_january, label='January High Temp', color='red', alpha=0.6)
plt.plot(merged_df['Year'], january_pred, label=f'January Trend ({january_trend:.2f}°F/year)',
        color='red', linestyle='--')
plt.title('Temperature Trends in New York City (1895-2025)')
plt.xlabel('Year')
```

```
plt.ylabel('Temperature (°F)')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output: -



4. Load the Iris dataset from the scikit-learn library and perform classification on it with the k-nearest neighbors algorithm. Use a KNeighborsClassifier with the default k value. What is the prediction accuracy?

Ans:-

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
iris = load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Prediction Accuracy: {accuracy:.2f}")
```

Output:- `Prediction Accuracy: 1.00`

5. You are given a dataset of 2D points with their corresponding class labels. The dataset is as follows:

Point ID	*x*	*y*	Class
A	2.0	3.0	0
B	1.0	1.0	0
C	4.0	4.0	1
D	5.0	2.0	1

A new point *P* with coordinates (3.0, 3.0) needs to be classified using the KNN algorithm. Use the Euclidean distance to calculate the distance between points.

Ans:-

```
import numpy as np
from collections import Counter
data = [
    [2.0, 3.0, 0],
```

```
    [1.0, 1.0, 0],
    [4.0, 4.0, 1],
    [5.0, 2.0, 1],
]
P = np.array([3.0, 3.0])
distances = []
for point in data:
    coord = np.array(point[:2])
    dist = np.linalg.norm(P - coord)
    distances.append((dist, point[2]))
distances.sort(key=lambda x: x[0])
k = 3
k_neighbors = distances[:k]
classes = [cls for _, cls in k_neighbors]
most_common = Counter(classes).most_common(1)[0][0]
print(f"Predicted class for point {P}: {most_common}")
```

Output:- `Predicted class for point [3. 3.]: 1`

6. A teacher wants to classify students as ”Pass” or ”Fail” based on their performance in three exams. The dataset includes three features:

Exam 1 Score	Exam 2 Score	Exam 3 Score	Class (Pass/Fail)
85	90	88	Pass
70	75	80	Pass
60	65	70	Fail
50	55	58	Fail
95	92	96	Pass
45	50	48	Fail

A new student has the following scores:
- Exam 1 Score: 72
- Exam 2 Score: 78
- Exam 3 Score: 75

Classify this student using the K-Nearest Neighbors (KNN) algorithm with k = 3.

Ans: -

```
import numpy as np
from collections import Counter
data = [
    [85, 90, 88, 'Pass'],
    [70, 75, 80, 'Pass'],
    [60, 65, 70, 'Fail'],
    [50, 55, 58, 'Fail'],
    [95, 92, 96, 'Pass'],
    [45, 50, 48, 'Fail'],
]
new_student = np.array([72, 78, 75])
distances = []
for row in data:
    scores = np.array(row[:3])
    label = row[3]
    dist = np.linalg.norm(new_student - scores)
    distances.append((dist, label))
k = 3
distances.sort(key=lambda x: x[0])
k_neighbors = distances[:k]
labels = [label for _, label in k_neighbors]
most_common = Counter(labels).most_common(1)[0][0]
print(f"Predicted class for the new student: {most_common}")
```

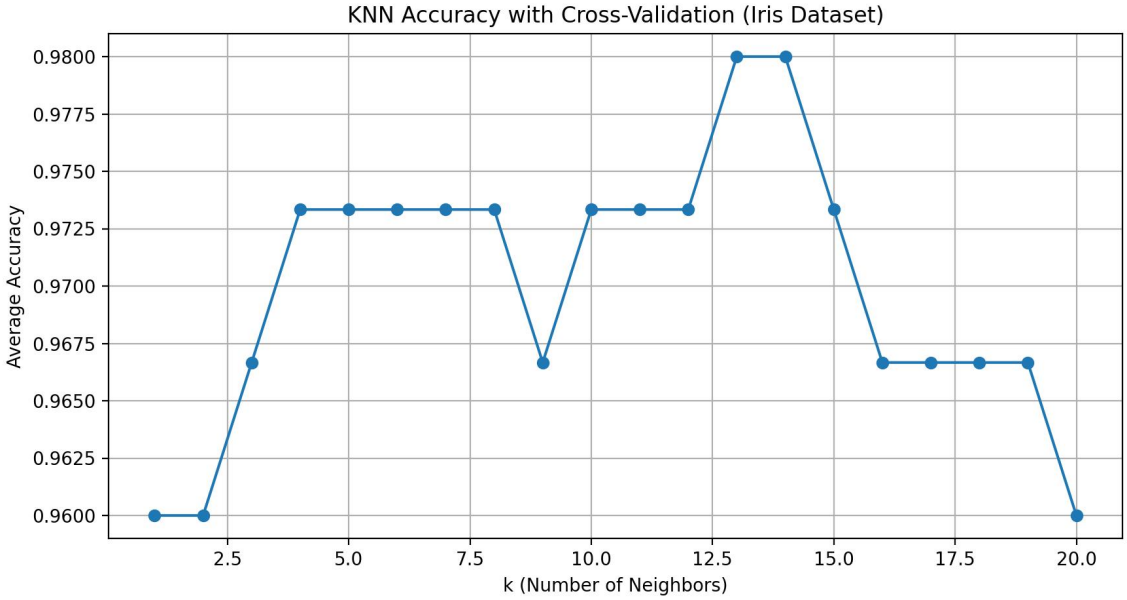
Output - `Predicted class for the new student: Pass`

7. Using scikit-learn's KFold class and the cross val score function, determine the optimal value for k to classify the Iris dataset using a KNeighborsClassifier.

Ans:-

```
from sklearn.datasets import load_iris
from sklearn.model_selection import KFold, cross_val_score
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
import numpy as np
iris = load_iris()
X, y = iris.data, iris.target
kf = KFold(n_splits=5, shuffle=True, random_state=42)
k_range = range(1, 21)
mean_scores = []
for k in k_range:
    model = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(model, X, y, cv=kf)
    mean_scores.append(scores.mean())
best_k = k_range[np.argmax(mean_scores)]
print(f"Optimal k: {best_k}")
print(f"Accuracy: {max(mean_scores):.4f}")
plt.figure(figsize=(10, 5))
plt.plot(k_range, mean_scores, marker='o')
plt.xlabel('k (Number of Neighbors)')
plt.ylabel('Average Accuracy')
plt.title('KNN Accuracy with Cross-Validation (Iris Dataset)')
plt.grid(True)
plt.show()
```

Output:-



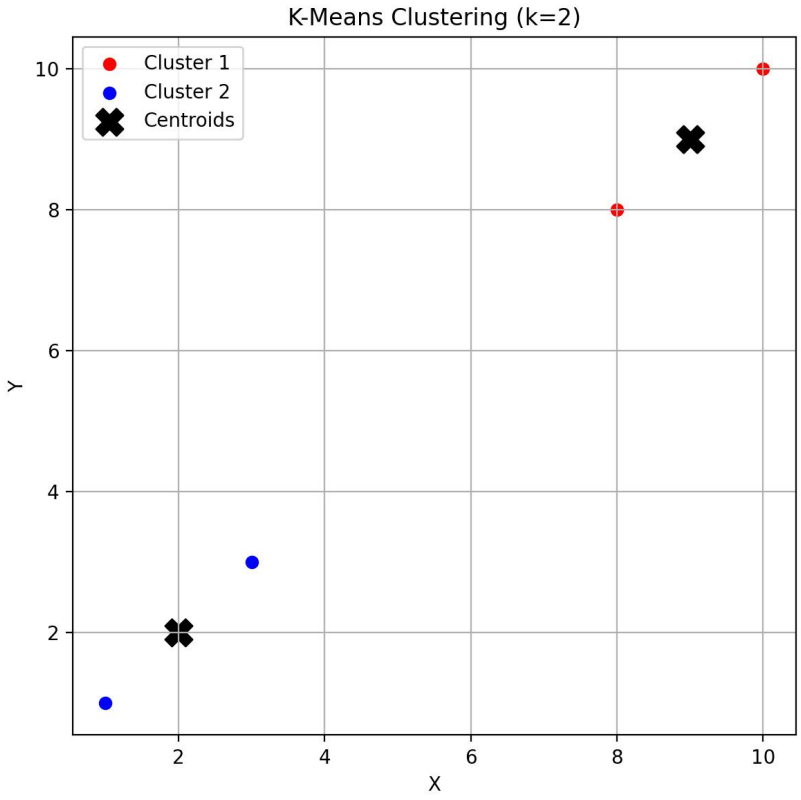
8. Write a Python script to perform K-Means clustering on the following dataset:
Dataset: {(1, 1),(2, 2),(3, 3),(8, 8),(9, 9),(10, 10)}
Use k=2 and visualize the clusters.

Ans:-

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
data = np.array([
    [1, 1],
    [2, 2],
    [3, 3],
    [8, 8],
    [9, 9],
```

```
    [10, 10]
])
kmeans = KMeans(n_clusters=2, random_state=0, n_init='auto')
kmeans.fit(data)
labels = kmeans.labels_
centers = kmeans.cluster_centers_
plt.figure(figsize=(6, 6))
colors = ['red', 'blue']
for i in range(2):
    cluster_points = data[labels == i]
    plt.scatter(cluster_points[:, 0], cluster_points[:, 1], color=colors[i], label=f'Cluster {i+1}')
plt.scatter(centers[:, 0], centers[:, 1], s=200, c='black', marker='X', label='Centroids')
plt.title("K-Means Clustering (k=2)")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```

Output:-



9. Write a Python script to perform K-Means clustering on the following dataset: Mall Customer Segmentation. Use k = 5 (also, determine optimal k via the Elbow Method) and visualize the clusters to identify customer segments.

Expected Output:

• Scatter plot showing clusters (e.g., "High Income-Low Spenders," "Moderate Income-Moderate Spenders").

• Insights for targeted marketing strategies.

Ans:-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
df = pd.read_csv('Mall_Customers.csv')
```

```
print(df.head())
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
inertia = []
K = range(1, 11)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X)
    inertia.append(kmeans.inertia_)
plt.figure(figsize=(8, 4))
plt.plot(K, inertia, 'bo-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.grid(True)
plt.show()
k_optimal = 5
kmeans = KMeans(n_clusters=k_optimal, random_state=42)
df['Cluster'] = kmeans.fit_predict(X)
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='Annual Income (k$)', y='Spending Score (1-100)',
                hue='Cluster', palette='Set1')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=200,
            c='black', marker='X', label='Centroids')
plt.title('Customer Segments')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.grid(True)
plt.show()
```

Output:-

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

Insights: -

Figure 1: -

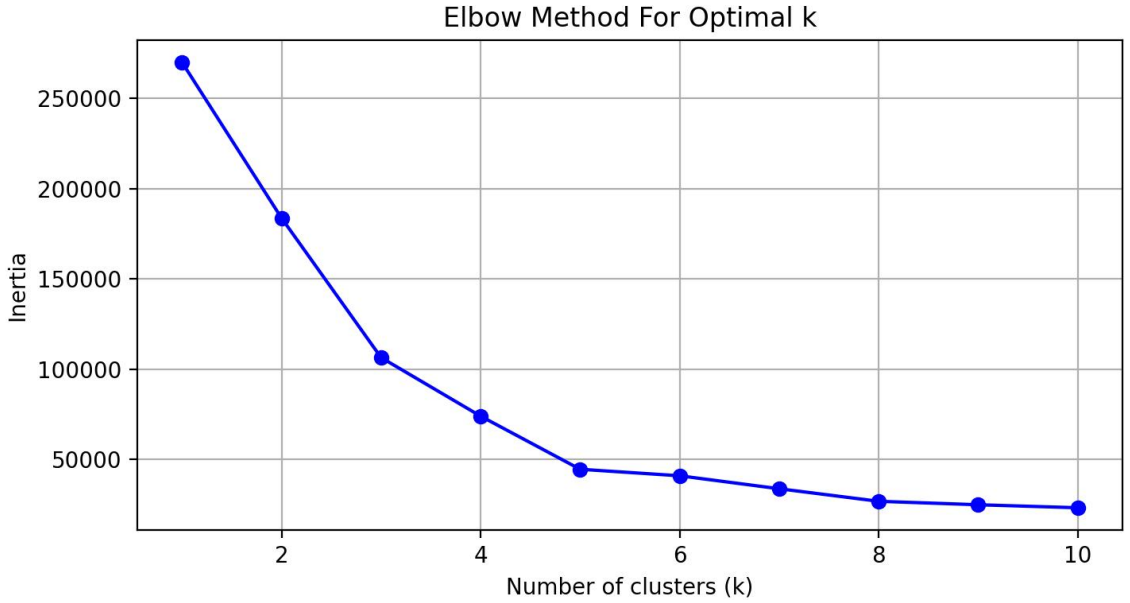
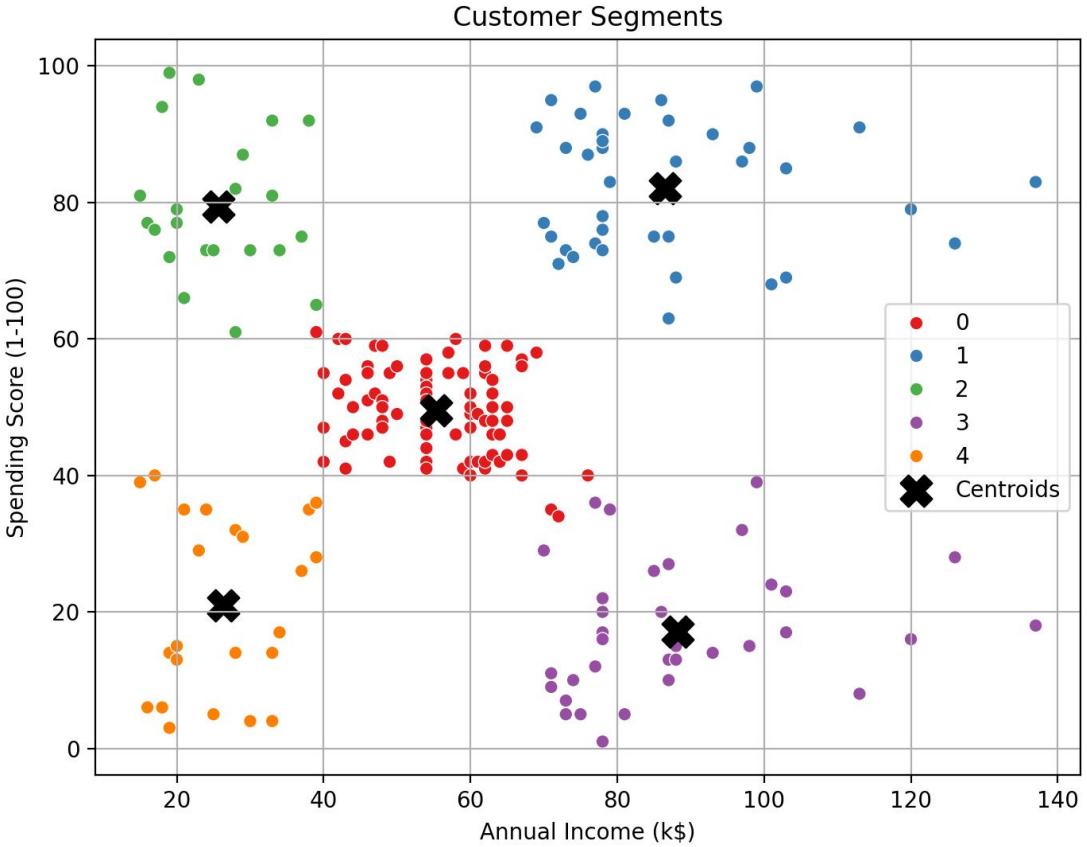


Figure 2: -



10. Perform the following tasks using the pandas Series object:
(a) Create a Series from the list [7, 11, 13, 17].
(b) Create a Series with five elements where each element is 100.0.
(c) Create a Series with 20 elements that are all random numbers in the range 0 to 100. Use the describe method to produce the Series' basic descriptive statistics.
(d) Create a Series called temperatures with the following floating-point values: 98.6, 98.9, 100.2, and 97.9. Use the index keyword argument to specify the custom indices 'Julie', 'Charlie', 'Sam', and 'Andrea'.
(e) Form a dictionary from the names and values in Part (d), then use it to initialize a Series.

Ans:-

```
import pandas as pd
import numpy as np
# (a)
series_a = pd.Series([7, 11, 13, 17])
print("Series a:\n", series_a, "\n")
# (b)
series_b = pd.Series([100.0] * 5)
print("Series b:\n", series_b, "\n")
# (c)
series_c = pd.Series(np.random.randint(0, 101, size=20))
print("Series c:\n", series_c, "\n")
print("Descriptive statistics:\n", series_c.describe(), "\n")
# (d)
temperatures = pd.Series(
    [98.6, 98.9, 100.2, 97.9],
    index=['Julie', 'Charlie', 'Sam', 'Andrea']
)
```

```
print("Series d:\nTemperatures Series:\n", temperatures, "\n")
# (e)
temp_dict = temperatures.to_dict()
series_from_dict = pd.Series(temp_dict)
print("Series e:\nSeries from dictionary:\n", series_from_dict)
```

Output:-

```
Series a:
 0     7
1    11
2    13
3    17
dtype: int64

Series b:
 0    100.0
1    100.0
2    100.0
3    100.0
4    100.0
dtype: float64

Series d:
Temperatures Series:
 Julie       98.6
Charlie     98.9
Sam        100.2
Andrea      97.9
dtype: float64

Series e:
Series from dictionary:
 Julie       98.6
Charlie     98.9
Sam        100.2
Andrea      97.9
dtype: float64
```

```
Series c:
 0     90
1     93
2     88
3     87
4     40
5     44
6     19
7     22
8     12
9      3
10     5
11    31
12    63
13    27
14    44
15    29
16    45
17    29
18    21
19     6
dtype: int64

Descriptive statistics:
 count    20.000000
mean     39.900000
std      29.522338
min       3.000000
25%      20.500000
50%      30.000000
75%      49.500000
max      93.000000
dtype: float64
```