Introduction

TapToBuy, an online grocery store, has been facing stiff competition from personalized marketing strategies employed by its competitors. To bridge this gap, customer segmentation was performed using unsupervised machine learning models—K-Means, Hierarchical, and DBSCAN clustering—to identify distinct customer groups for targeted marketing campaigns.

Data Overview

The dataset for TapToBuy contains 4,832 rows and 9 columns. The dataset appears to represent customer demographic and behavioral data, for a retail or e-commerce context.These are the following customer attributes:

‘Gender’, ‘Ever Married’, ‘Age’, ‘Graduated’, ‘Profession’, ‘Work Experience’, ‘Spending Score’, ‘Family Size’.

While all the above features are self-explanatory, the Spending_Score feature is a categorical variable representing the customer's spending behavior or creditworthiness where the possible values include 'Low', 'Average', 'High'.

Data Preprocessing

After exploring the usual characteristics of the dataset, I checked for the missing values and there are indeed quite a few missing values in most of the columns-

```
In [8]: print("\nMissing Values:\n", df_gc.isnull().sum())

Missing Values:
 ID                 0
Gender              0
Ever_Married       85
Age                 0
Graduated          46
Profession         74
Work_Experience   483
Spending_Score      0
Family_Size       203
```

Missing values were handled by imputing the median for numerical variables such as ‘Work_Experience’ and ‘Family_size’ while for categorical variables, the mode was used. Categorical variables were then encoded using Label Encoding (it is important to first fill the missing values before using Label encoding as otherwise a separate category will be created for NaN types. After checking there were no more missing values, the features were scaled using StandardScaler for better model performance.

The value counts for each of the binary variables such as Gender, Marriage & Graduation status are shown below-

```
In [65]: df_gc['Gender'].value_counts()
Out[65]:
Gender
1    2633
0    2199
Name: count, dtype: int64

In [66]: df_gc['Ever_Married'].value_counts()
Out[66]:
Ever_Married
1    2840
0    1992
Name: count, dtype: int64

In [67]: df_gc['Graduated'].value_counts()
Out[67]:
Graduated
1    2985
0    1847
Name: count, dtype: int64
```

Similarly, for spending scores the split is shown here (where 2: 'Low', 1: 'High', 0: 'Average) -

```
In [69]: df_gc['Spending_Score'].value_counts()
Out[69]:
Spending_Score
2    2943
0    1175
1     714
Name: count, dtype: int64
```

Finally the family_size and profession categorical splits for count in each type are shown below:

```
In [68]: df_gc['Profession'].value_counts()
Out[68]:
Profession
0    1544
5     804
3     585
2     415
1     415
7     387
4     361
8     173
6     148
```

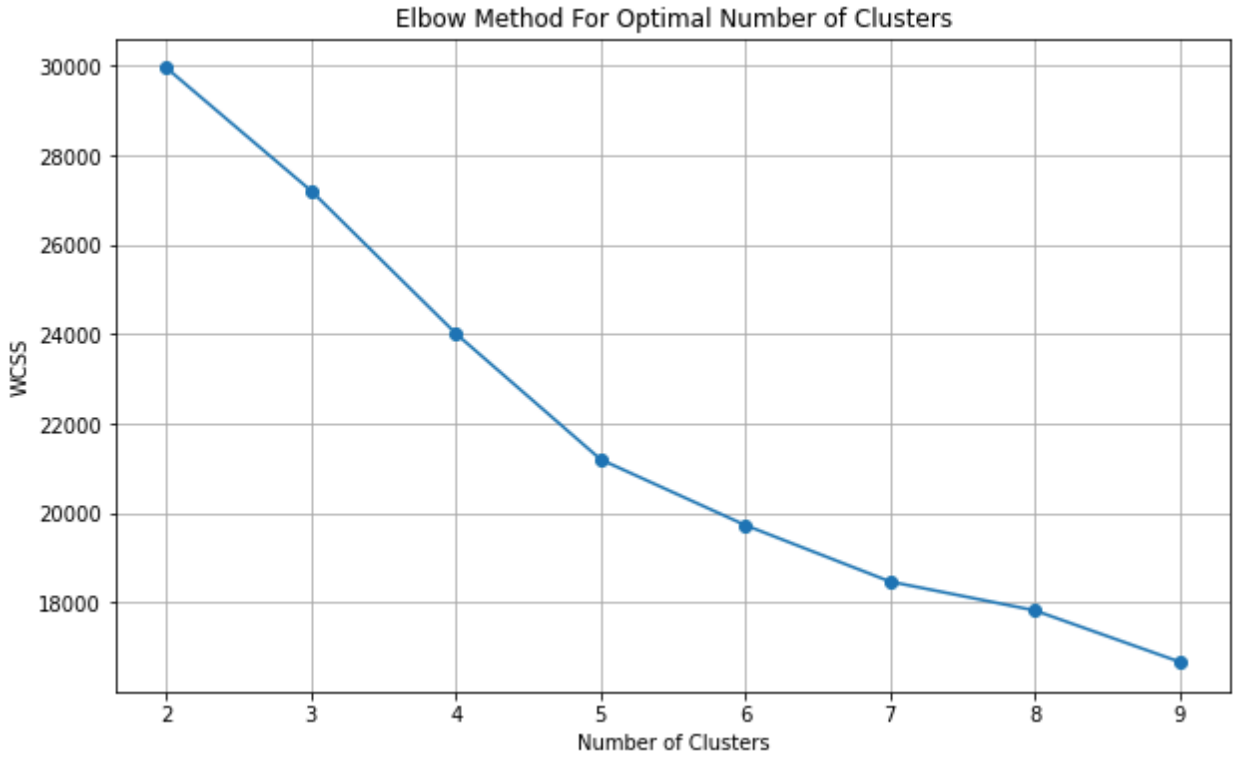
```
In [70]: df_gc['Family_Size'].value_counts()
Out[70]:
Family_Size
2.0    1422
3.0    1117
1.0     857
4.0     834
5.0     374
6.0     118
7.0      59
8.0      30
9.0      21
```

Before moving to the next step of different clustering models, I also used PCA to reduce the dimensions and visualise the clusters better.

Clustering Models

K-Means Clustering

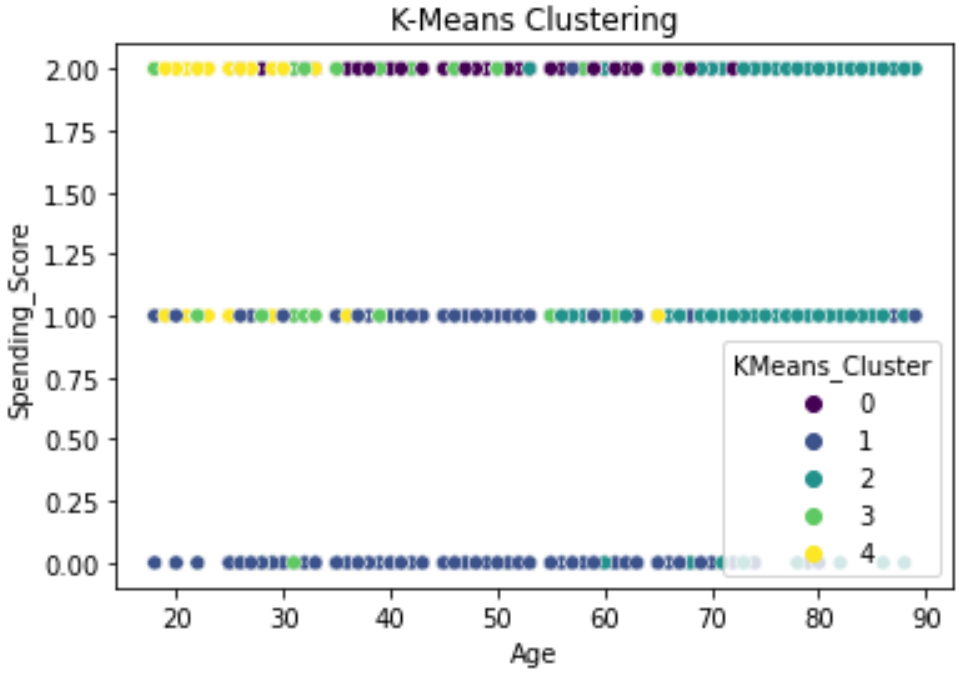
The optimal number of clusters (K) was determined using the Elbow Method. I also used the silhouette scores to check if it provides any better clarity on the number of clusters to be used, however, I went for 5 clusters as seen from the elbow method (although one could equally argue for 4 or 6 clusters).

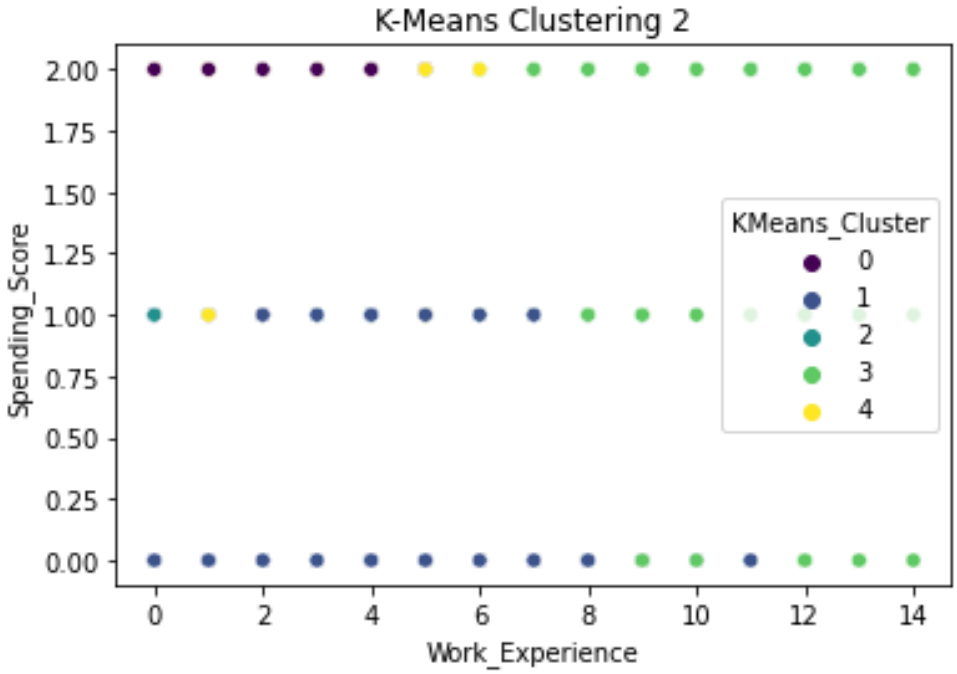


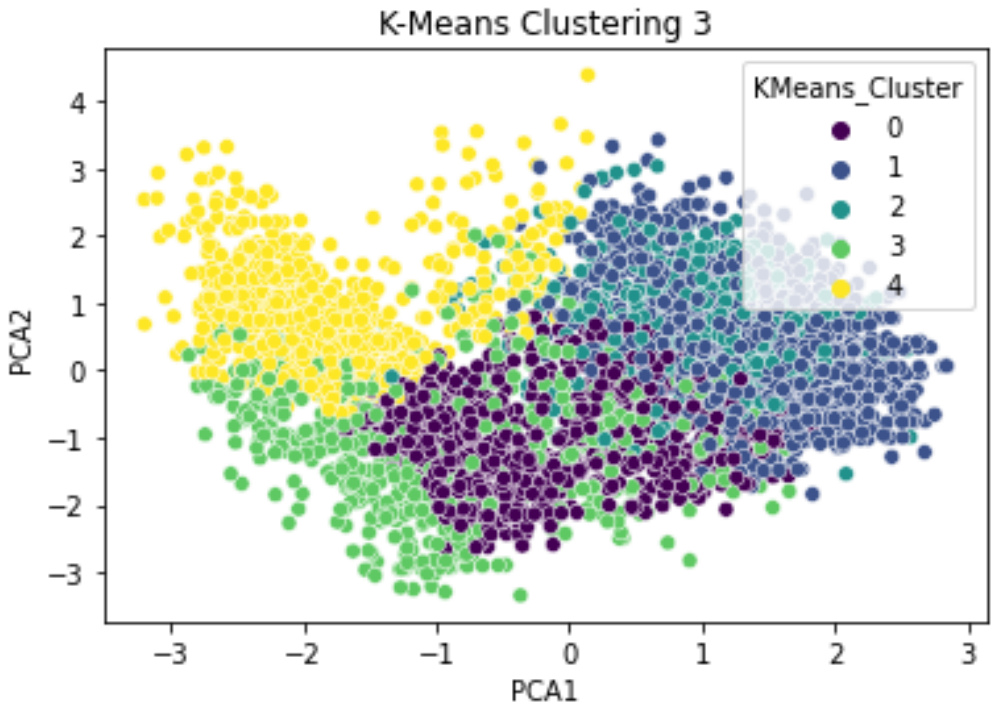
Using the silhouette scores below, one could have possibly gone for either 2 or 6 clusters-

```
In [40]: for i, score in zip(cluster_range, silhouette_scores):
    ...:     print(f'Number of Clusters: {i}, Silhouette Score: {score}')
Number of Clusters: 2, Silhouette Score: 0.22420234727021546
Number of Clusters: 3, Silhouette Score: 0.20207323009087177
Number of Clusters: 4, Silhouette Score: 0.2146741766651604
Number of Clusters: 5, Silhouette Score: 0.21490509313925624
Number of Clusters: 6, Silhouette Score: 0.21765923680380803
Number of Clusters: 7, Silhouette Score: 0.20934046745774726
Number of Clusters: 8, Silhouette Score: 0.20864178395418526
Number of Clusters: 9, Silhouette Score: 0.21367811235290968
```

After fitting the K-Means model to the standardized data using 5 clusters, I visualised the clusters first using ‘Spending_Score’ and ‘Age’, then ‘Spending_Score’ vs ‘Work_Experience’ and then finally clusters were visualized using PCA for dimensionality reduction.







The scatter plot above represents customer segmentation using K-Means clustering and PCA to reduce the high-dimensional data to two principal components (PCA1 & PCA2) for visualization. Each point represents a customer, color-coded based on their assigned K-Means cluster (0-4).

Cluster 0 (purple) is densely packed in the center, signifying a core group of customers with relatively homogeneous characteristics.

Cluster 1 (dark blue) is primarily concentrated in the middle-right of the plot, suggesting customers with moderate variation in attributes.

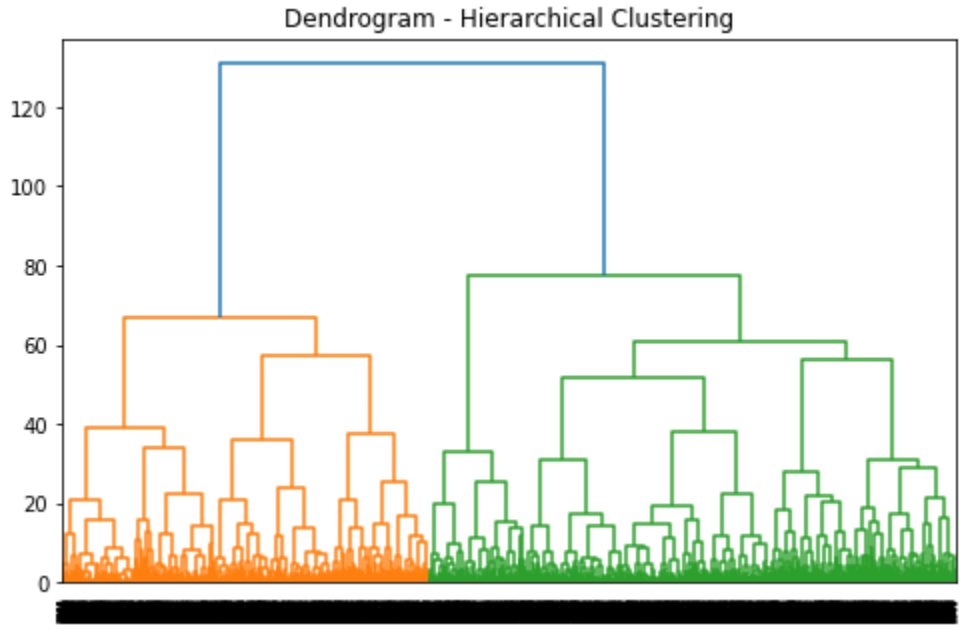
Cluster 2 (dark green) covers a large portion of the right-hand side, interspersed with Cluster 1, indicating similarity in customer behavior but slightly different segmentation.

Cluster 3 (light green) is primarily spread on the bottom left-hand side, showing distinct customer traits differing from the majority, although a few of them are also mixed with Cluster 0 in the centre.

Cluster 4 (yellow): Positioned at the top far-left, again suggesting a unique segment with significantly different customer attributes.

Hierarchical Clustering

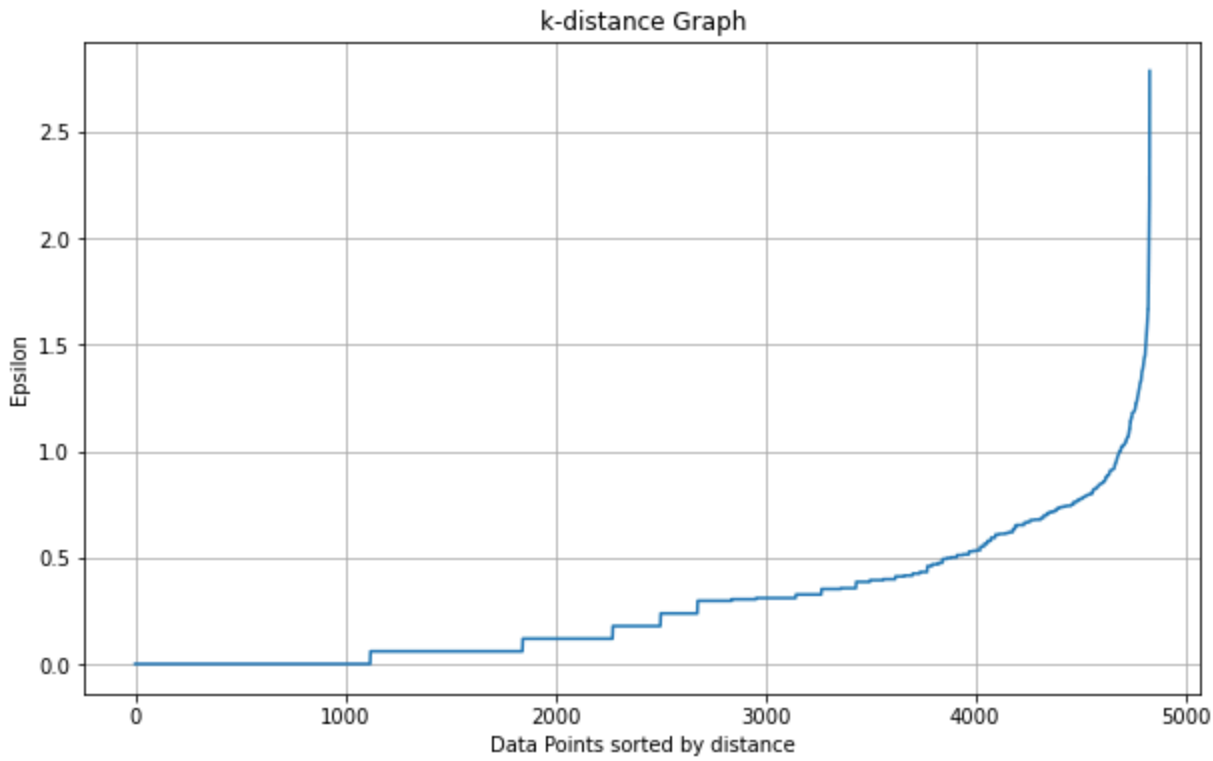
Agglomerative clustering was applied with Ward's linkage method. A dendrogram was used to determine the optimal number of clusters.



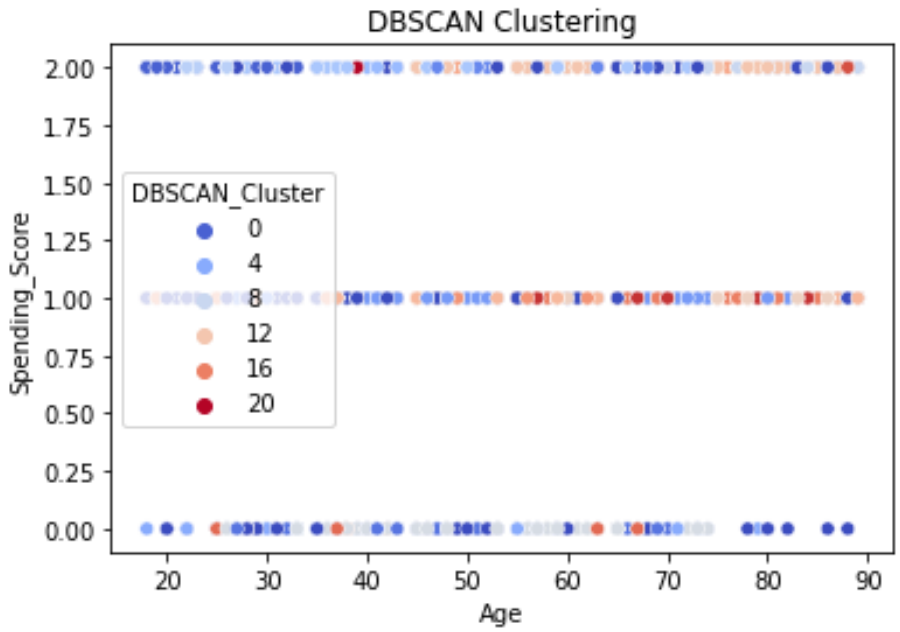
The y-axis represents the linkage distance (dissimilarity between clusters) while the x-axis represents individual data points (customers). The two primary branches (orange & green) indicate that the dataset naturally splits into two major clusters at a high level. This in a way agrees with what we saw in K-Means clustering as well where the silhouette scores suggested 2 clusters as the best approach. So based on the dendrogram, choosing 2-4 clusters seems reasonable for effective segmentation.

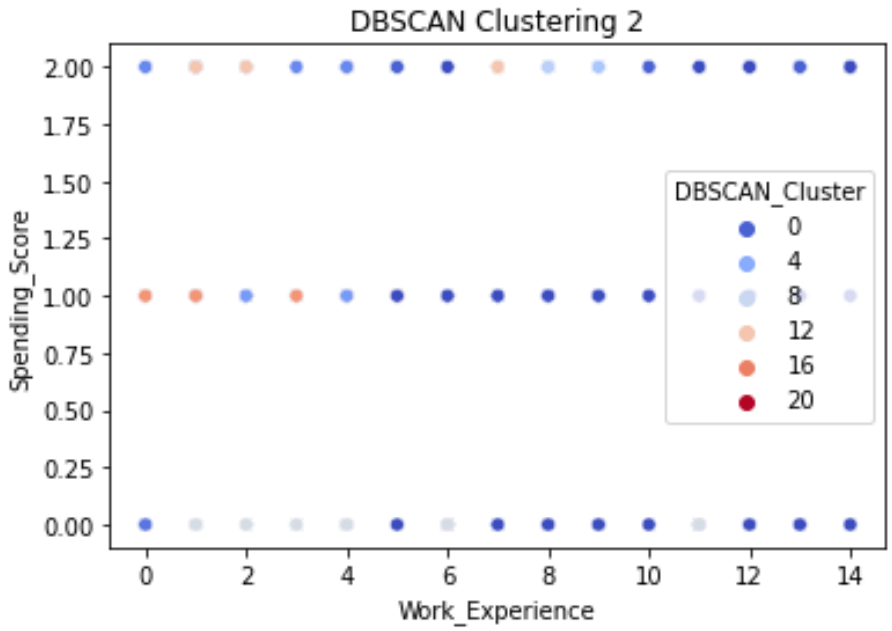
DBSCAN Clustering

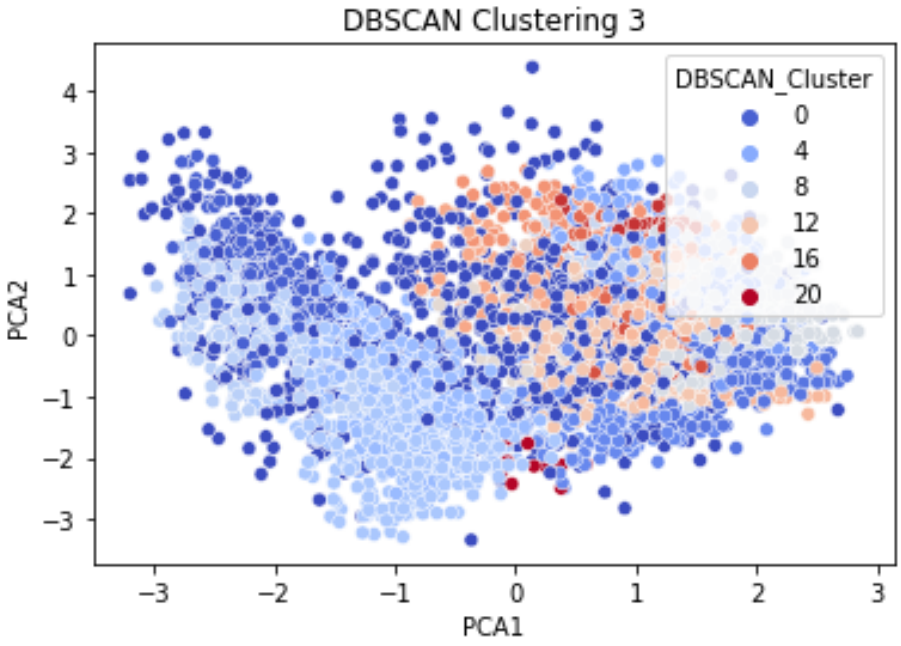
The k-distance graph is used to determine the optimal value of epsilon (eps). The "elbow" point in the graph (where the slope sharply increases) is often used to select the eps value. Based on the below graph, an eps value of approximately 1 seems reasonable, as distances increase significantly beyond this point.



Similar to K-means clustering, I visualised the DBSCAN clusters first using ‘Spending_Score’ and ‘Age’, then ‘Spending_Score’ vs ‘Work_Experience’ and then finally clusters were visualized using PCA for dimensionality reduction.







DBSCAN identified multiple clusters, with the majority of points assigned to a primary cluster (blue). The presence of several smaller clusters (red, orange) suggests that DBSCAN is capturing local density variations. Unlike K-Means & Hierarchical clustering, DBSCAN does not force all points into clusters and is better able to handle outliers.

The last DBSCAN plot using PCA1 and PCA2 shows that while some clusters are well-defined, others may be overlapping or fragmented.