Name: Zubayer Hassan

Roll:23081554

Github: https://github.com/zbrhsn/Ads-Clustering

K-Means Clustering Tutorial: A Practical Guide

Introduction

In unsupervised learning K means clustering is one of the most used machine learning algorithms to find out distinct group from a dataset. It helps in discovering hidden patterns, segmenting data, and enabling better decision-making in various fields like marketing, biology, and image processing. This tutorial aims to provide a detailed understanding of K-Means Clustering, using a cereal dataset as an example. We will explore the mathematical foundations, step-by-step implementation, and visual interpretation using Python.

K-Means clustering intends to partition n objects into k clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly k different clusters of greatest possible distinction. The best number of clusters k leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of K-Means clustering is to minimize total intra-cluster variance, or, the squared error function:

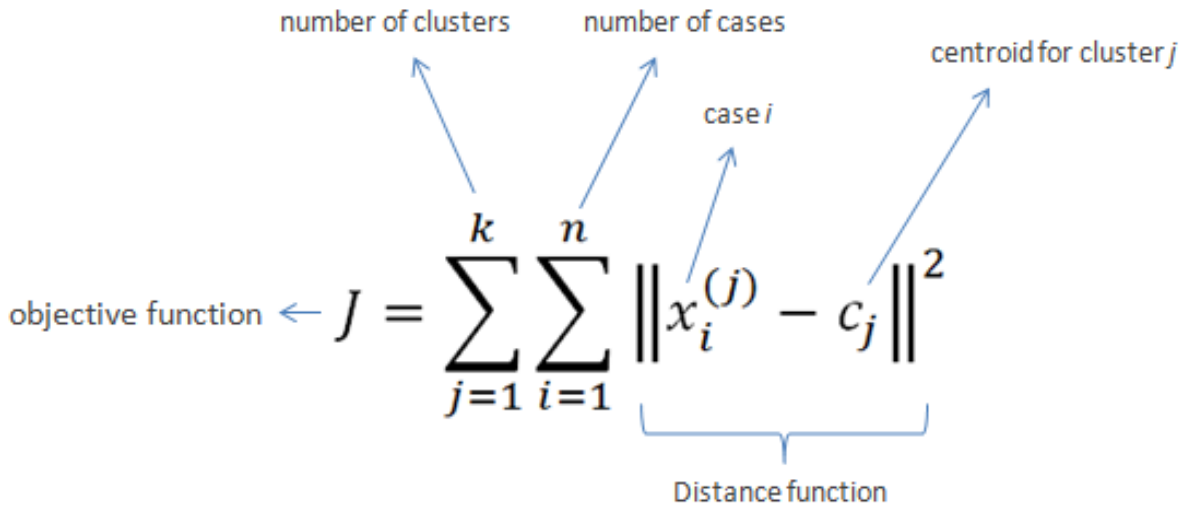

Figure 1: Formula of KMeans Clustering

Algorithm of K Means Clustering:

1. Clusters the data into *k* groups where *k* is predefined.
2. Select *k* points at random as cluster centers.
3. Assign objects to their closest cluster center according to the *Euclidean distance* function.
4. Calculate the centroid or mean of all objects in each cluster.
5. Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds

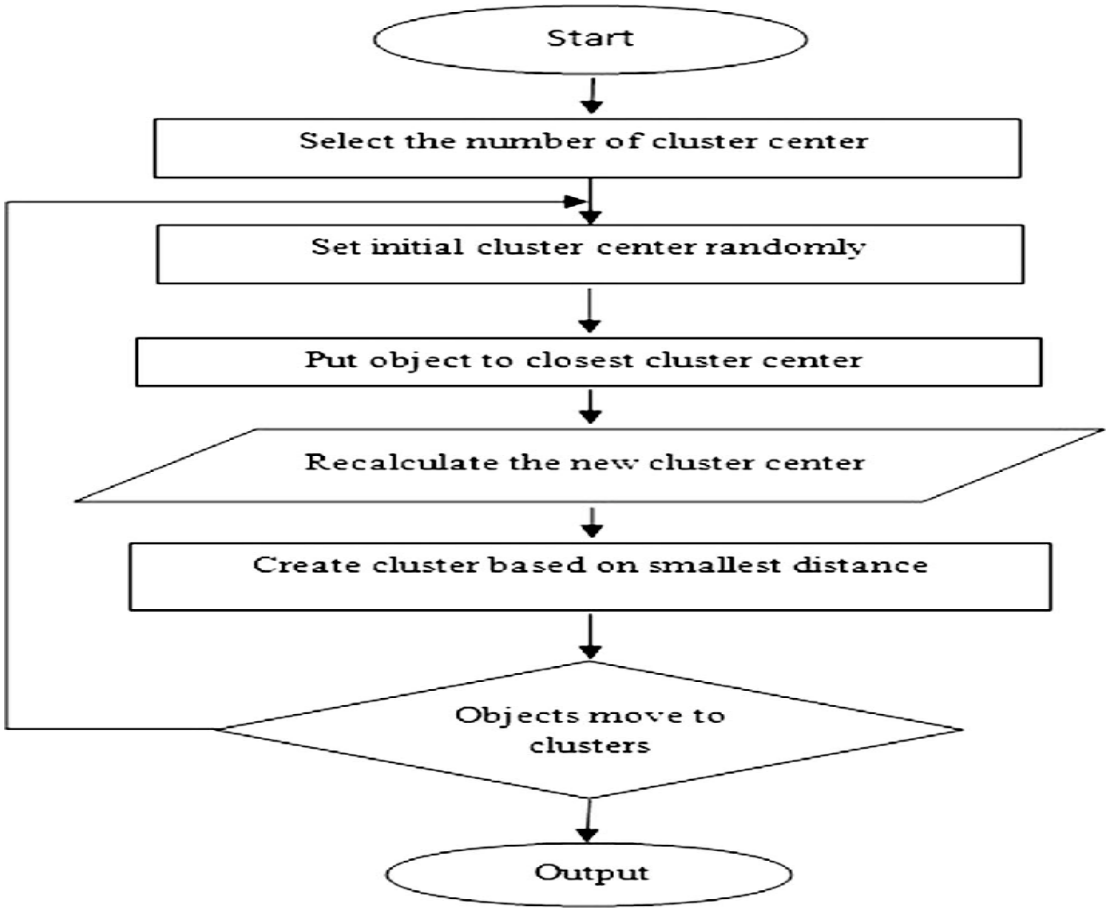

Figure 2: Flow Chart of KMeans Clustering

Data Overview and Objectives:

The dataset contains **150 rows and 4 columns**, representing individuals and their engagement with different entertainment categories. The columns include:

- **name**: Name of People
- **books**: Indicating the individual's engagement with books.
- **tv_shows**: Indicating the individual's engagement with TV shows.
- **video_games**: Individual's engagement with video games.

There are **no missing values** in any column.The numerical columns (books, tv_shows, and video_games) are stored as float.
From the given values it is needed to find how many ads it need run for a agency or organization.

Pre Processing of Data:
In the preprocessing of data column named is removed as the data set is used for k means clustering which is unsupervised learning.

```python
# check 5: select features? exclude the name column for modeling.
data = df.drop(columns='name')
data.head()
```

Figure 3: Removing name column

Step 1: Choose the Number of Clusters (K)

Selecting the optimal number of clusters (K) is crucial to ensure meaningful grouping. A common approach is the **Elbow Method**, which involves:

- Running K-Means for different K values
- Plotting inertia (sum of squared distances of data points from centroids)
- Identifying the "elbow" point where inertia stops decreasing sharply

```python
# create an empty list to hold many inertia values
inertia_values = []

# create 2 - 15 clusters, and add the intertia scores to the list
for k in range(2, 16):
    kmeans = KMeans(n_clusters=k, n_init='auto', random_state=42)
    kmeans.fit(data)
    inertia_values.append(kmeans.inertia_)
```

```python
# plot the inertia values
import matplotlib.pyplot as plt

# turn the list into a series for plotting
inertia_series = pd.Series(inertia_values, index=range(2, 16))

# plot the data
inertia_series.plot(marker='o')
plt.xlabel("Number of Clusters (k)")
plt.ylabel("Inertia")
plt.title("Number of Clusters vs. Inertia");
```

Figure 4: Showing k = 2 to 16 inertia vs clusters using python

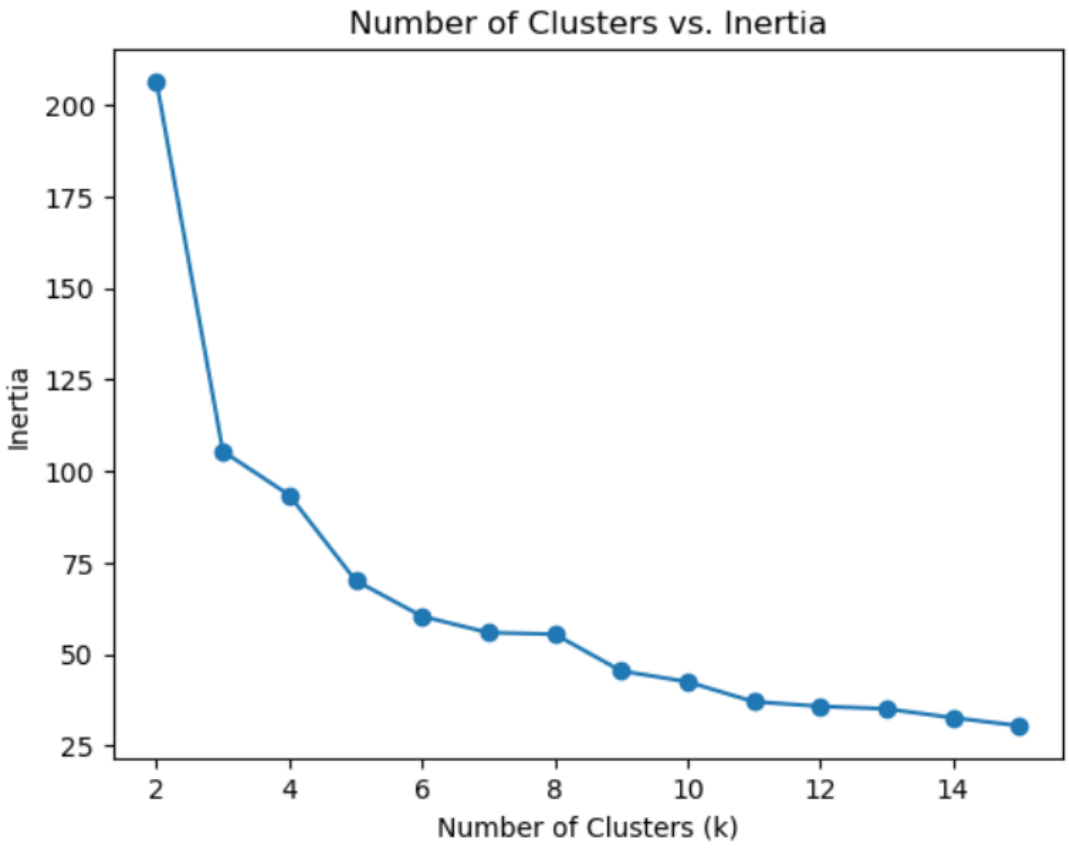

Figure 5: Inertia vs Clusters Graph

Step 2: Initialize Cluster Centroids

After selecting K, the algorithm initializes K centroids. The **K-Means++ initialization** is preferred as it speeds up convergence.

```
# import plotting libraries
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D

# combine the data and cluster labels
cluster_labels = pd.Series(kmeans2.labels_, name='cluster')

# create a clean dataframe
df_clean = pd.concat([data, cluster_labels], axis=1)

# create a 3d scatter plot
fig = plt.figure(figsize=(8, 6))
ax = Axes3D(fig)
fig.add_axes(ax)

# specify the data and labels
sc = ax.scatter(df_clean['books'], df_clean['tv_shows'], df_clean['video_games'],
                c=df_clean['cluster'], cmap='tab10')
ax.set_xlabel('books')
ax.set_ylabel('tv_shows')
ax.set_zlabel('video_games')

# add a legend
plt.legend(*sc.legend_elements(), title='clusters',
           bbox_to_anchor=(1.05, 1));
```

Figure 6 :Plotting k=2 cluster using Python.

Step 3: Assign Data Points to the Nearest Centroid

Each data point is assigned to the closest centroid based on **Euclidean distance**:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Figure 7: Euclidean Distance method.

The above code step 2 will cover the assign data point to the nearest centroid after assigning the data points to the nearest centroid the image will be like.

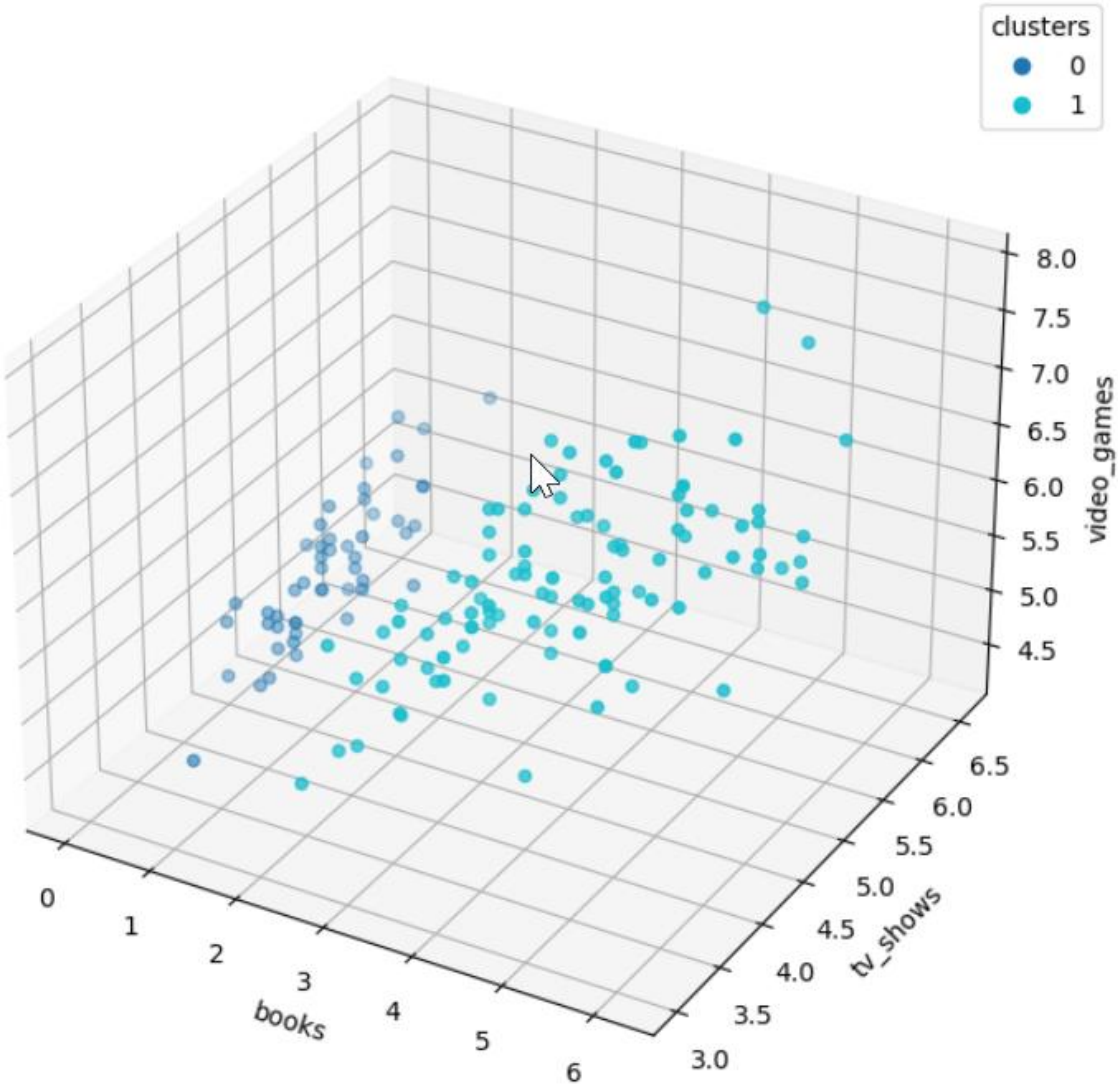

Figure 8: Plotting value for k=2

Step 4: Update Centroids

For cluster 2 analyse the results:

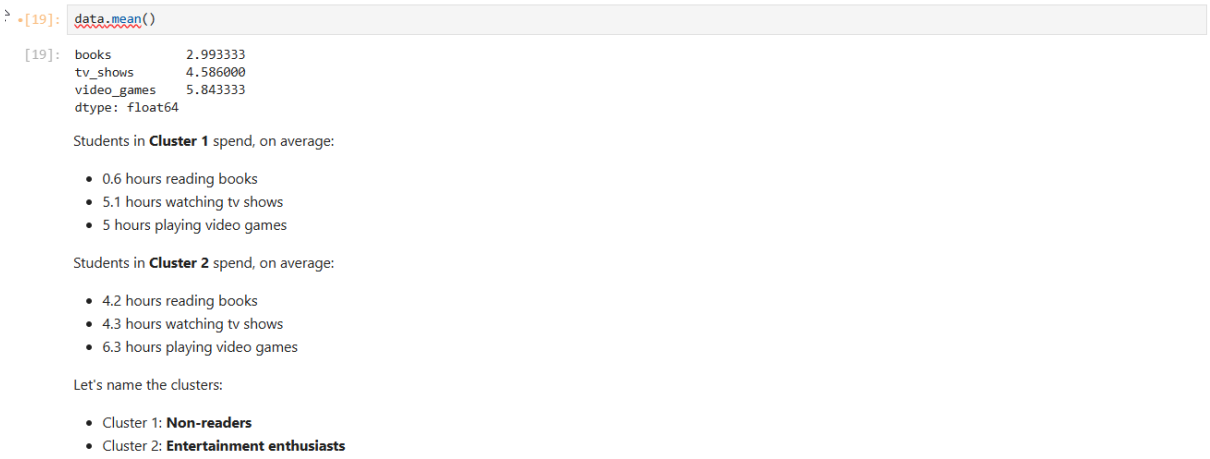

Figure 9: Cluster 2 result comparison.

It shows Students in **Cluster 1** spend, on average:

- 0.6 hours reading books
- 5.1 hours watching tv shows
- 5 hours playing video games

Students in **Cluster 2** spend, on average:

- 4.2 hours reading books
- 4.3 hours watching tv shows
- 6.3 hours playing video games

Let's name the clusters:

- Cluster 1: **Non-readers**
- Cluster 2: **Entertainment enthusiasts**

New centroid positions are computed by taking the mean of all points in each cluster

Step 5: Repeat Until Convergence

Steps 3 and 4 repeat until centroids stabilize or the maximum iterations are reached.

```
# fit a kmeans model with 3 clusters
kmeans3 = KMeans(n_clusters=3, n_init='auto', random_state=42)
kmeans3.fit(data)
```

```
C:\Users\zbrhs\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
KMeans(n_clusters=3, n_init='auto', random_state=42)
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.
KMeans
KMeans(n_clusters=3, n_init='auto', random_state=42)
```

```
# view the cluster labels
kmeans3.labels_
```

```
array([0, 0, 0, 2, 2, 1, 1, 2, 0, 2, 1, 0, 0, 0, 0, 0, 2, 0, 1, 1, 2, 2,
       2, 1, 2, 1, 0, 0, 0, 1, 2, 1, 1, 1, 1, 2, 2, 0, 0, 1, 2, 2, 0, 0,
       1, 2, 0, 1, 0, 1, 1, 1, 2, 2, 2, 1, 0, 1, 2, 2, 0, 0, 0, 2, 1, 1,
       0, 2, 1, 2, 1, 1, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 0, 0, 0, 2, 2, 0,
       0, 2, 0, 2, 2, 2, 2, 1, 2, 1, 2, 0, 1, 0, 2, 2, 1, 2, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 1, 2, 0, 1, 2, 2, 1, 0, 0, 0, 1, 2, 2, 2, 0, 0, 0,
       1, 0, 2, 1, 1, 1, 1, 2, 0, 0, 2, 1, 2, 1, 1, 1, 0, 0])
```

```
# view the cluster centers
kmeans3.cluster_centers_
```

```
array([[0.596     , 5.13      , 5.006     ],
       [5.14375   , 4.52708333, 6.63958333],
       [3.31346154, 4.11730769, 5.91346154]])
```

Figure 10 : Calculating values for k=3

Then repeat step 3 and 4 for cluster 3 and continue increase cluster till maximum iteration is reached.

From the step 2 k=3 and k=5 are the elbow but for k=5 there are some less categorial value in number. So, k=3 and k=4 is picked for final result
There are total 150 entry and for k=3 the clusters are named like below and quantity is also given below:

- Cluster 0: Non-readers (50)
- Cluster 1: Entertainment enthusiasts (48)
- Cluster 2: Prefer video games to books (52)

For k=4 the name is and quantity is given below:

- Cluster 0: Less entertainment, especially books (50)
- Cluster 1: Less screens (36)
- Cluster 2: Entertainment enthusiasts, especially video games (12)
- Cluster 3: Typical students (52)

Conclusion: In this situation, looking at a combination of the results of both models, the recommendation could be to create:

1. A general ad for the typical teen
2. An ad targeted towards teens who don't read very much
3. An ad targeted towards teens who love video games