Universidad Autónoma de Querétaro
DIVISIÓN DE INVESTIGACIÓN Y POSGRADO

Faculty of Engineering
MCIA

Activity 3

K-Means & Affinity Propagation

Student:
Juan Manuel
Aviña Muñoz

Professor:
Dr. Marco
Aceves Fernandez

August 6th, 2023

Contents

1 Introduction

In today's data-driven world, the ability to uncover meaningful patterns and structures within large datasets is of paramount importance. Data clustering, a fundamental technique in unsupervised machine learning, plays a crucial role in this endeavor. One of the most widely used clustering algorithms is K-Means, a method that partitions data points into distinct groups or clusters based on their similarity. K-Means is not only a foundational technique in machine learning but also finds applications in diverse fields such as image segmentation, customer segmentation, anomaly detection, and more.

Similarly, in the realm of machine learning, the concept of affinity holds significant importance in tasks such as clustering, recommendation systems, and anomaly detection. Affinity, at its core, quantifies the degree of similarity or closeness between data points or objects, enabling us to measure the relationships and shared characteristics between entities.

This report seeks to provide a comprehensive understanding of both the K-Means clustering algorithm and the broader concept of affinity in machine learning. We will begin by establishing the theoretical foundations of these concepts, exploring their core principles, mathematical underpinnings, and key components. Moreover, we will delve into the strengths, weaknesses, and real-world applications of K-Means and affinity-based techniques, highlighting their significance in various domains of data analysis and machine learning.

2 Theoretical Foundation

2.1 K-Means

K-Means is grounded in the principles of partitioning data into clusters by minimizing the within-cluster variance and maximizing the between-cluster variance. This fundamental concept is rooted in several key components:

- Data Points and Clusters: At its core, K-Means operates on a dataset comprising N data points, each represented as a multidimensional vector. The algorithm seeks to group these data points into K clusters, where K is a user-defined parameter.
- Centroid-Based Clustering: K-Means is a centroid-based clustering algorithm, meaning that it assigns data points to clusters based on the proximity to a central point called a centroid. Each cluster has its own centroid, and the goal is to minimize the distance between data points and their respective cluster centroids.
- Objective Function: The core objective of K-Means is to minimize the within-cluster sum of squares, often referred to as the ”inertia” or ”distortion” function. Mathematically, this is expressed as the sum of squared Euclidean distances between data points and their assigned cluster centroids. The objective function is defined as:

$$J = \sum_{i=1}^{K} \sum_{j=1}^{N_i} \|x_j - \mu_i\|^2 \tag{1}$$

where: J is the objective function (inertia). K is the number of clusters. N_i is the number of data points in cluster i. x_j is a data point in cluster i. μ_i is the centroid of cluster i.

K-Means follows a straightforward iterative process:

1. Initialize K centroids either randomly or using a specific initialization method.
2. Assign each data point to the nearest centroid, forming K clusters.
3. Recalculate the centroids as the mean of data points in each cluster.
4. Repeat the assignment and centroid update steps until convergence or a predefined stopping criterion is met.

In summary, K-Means is a versatile clustering algorithm rooted in the minimization of within-cluster variance. Understanding its theoretical foundation is pivotal for grasping its functionality and capabilities. In the following sections of this report, we will delve deeper into the practical implementation of K-Means, its variants, and its real-world applications. By doing so, we will gain a comprehensive view of how this algorithm contributes to knowledge discovery and data analysis in various fields [1, 2, 3].

2.2 Affinity Propagation

Affinity in machine learning is rooted in the concept of similarity or closeness between data points, and it serves as the basis for numerous algorithms and techniques. The theoretical foundation of affinity can be broken down into several key components:

- **Data Representation:** In machine learning, data is often represented as vectors or matrices, with each data point described by a set of features. These features can be numeric, categorical, or even unstructured data such as text or images.
- **Distance Metrics:** Affinity is typically measured using distance or similarity metrics. A distance metric quantifies the dissimilarity between two data points, while a similarity metric quantifies their likeness. Common distance metrics include Euclidean distance, Manhattan distance, and cosine similarity. These metrics help determine the proximity or similarity between data points in a feature space.
- **Affinity Matrices:** Affinity between data points is often represented as an affinity matrix, also known as a similarity matrix or distance matrix. In this matrix, each element (i, j) corresponds to the affinity or distance between data points i and j. The matrix is symmetric, and its diagonal elements represent the self-affinity of each data point.

- **Affinity-Based Algorithms:** Affinity-based algorithms leverage the affinity matrix to solve various machine learning tasks. Clustering algorithms, such as spectral clustering and hierarchical clustering, use affinity to group similar data points together. In recommendation systems, affinity is employed to identify user-item relationships and make personalized recommendations. Anomaly detection algorithms identify data points with low affinity to the rest of the dataset, signaling potential anomalies or outliers.

- **Kernel Methods:** In some cases, affinity can be computed implicitly using kernel methods. These methods project data points into a higher-dimensional feature space, where the dot product between data points approximates their affinity. The kernel trick, commonly used in Support Vector Machines (*SVMs*) and kernel *PCA*, is a prime example of this concept.

- **Graph Theory:** Affinity is closely related to graph theory, where data points can be represented as nodes in a graph, and edges between nodes reflect their affinity or similarity. Graph-based techniques, such as spectral clustering and PageRank, use graph structures to capture and exploit affinity relationships.

Affinity within the context of machine learning serves as a fundamental concept that allows us to measure the similarity or proximity between data points, ultimately facilitating the creation of diverse data-driven algorithms and methodologies. Grasping the theoretical underpinnings of affinity is of paramount importance for the adept application of these approaches to practical real-world challenges. In the forthcoming sections of this report, we will delve into the pragmatic implementations of affinity-based techniques and assess their pivotal roles across various domains within the realms of machine learning and data analysis. [4, 5].

2.3 Elbow Method

The elbow method is a fundamental technique used in unsupervised machine learning, particularly in the context of clustering algorithms, to determine the optimal number of clusters (K) for a given dataset. The primary objective is to find an optimal value for K, which represents the number of clusters into which the dataset should be partitioned. This determination is crucial for the effective application of clustering algorithms like K-Means.

The core concept behind the elbow method is the Sum of Squared Distances (SSD), which measures the sum of the squared distances between data points and their respective cluster centroids. It quantifies how well data points within a cluster are grouped around their centroid.

To apply the elbow method, we perform the following steps:

1. We run the K-Means algorithm for a range of K values, typically starting from a minimum value (e.g., 1) to a maximum value (which can be determined based on the problem).

2. For each K value, we calculate the SSD.

3. We then plot the K values against their corresponding SSD values. This results in an ”elbow curve.”

The elbow point represents the optimal K value, it is the point at which adding more clusters does not significantly reduce the SSD. In other words, it's the K value where the rate of decrease in SSD slows down, forming an ”elbow” shape in the curve.

Once the optimal K is determined using the elbow method, it can be used to run the K-Means clustering algorithm with the chosen K value, resulting in well-defined clusters for further analysis.

3 Methods and Materials

For this project, Python 3.9 was chosen as the primary programming language. A dedicated script was crafted, and a Jupyter notebook was constructed to facilitate the achievement of our project's objectives. This choice of programming environment allowed for a seamless workflow, enabling efficient code development and interactive exploration of the data and algorithms.

3.1 Data Set

We have a set of 240 Stars with 5 Types:

- Red Dwarf - 0
- Brown Dwarf - 1
- White Dwarf - 2
- Main Sequence - 3
- Super Giants - 4
- Hyper Giants - 5

Temperature: Temperature in Kelvin.
R: Radius of star relative to the sun.
L: Luminosity of the star relative to the sun.
Absolute Magnitude: Magnitude of the star relative to the sun.
Color: Color of the star.
Spectral Class: An asteroid spectral type is assigned to asteroids based on their emission spectrum, color, and sometimes albedo. These types are thought to correspond to an asteroid's surface composition.

Database obtained from the Kaggle platform [6] sourced from NASA observations.

3.2 Resources

Equipment:

- Intel i7-9750H @ 2.60 GHz processor.
- 16 GB DDR4 @ 2666 MHz.
- 1 Tb HDD + 1 Tb SSD.
- GeForce GTX 1660Ti 6 GB VRAM GDDR6.

3.3 Libraries

The following libraries were utilized for this activity:

- **NumPy** is a fundamental package for scientific computing in Python as it provides support for large, multi-dimensional arrays and matrices, along with a variety of mathematical functions to operate on these arrays. NumPy is a core library in the data science ecosystem and is used for tasks like numerical computations, linear algebra operations, random number generation, and more.
- **Pandas** is a powerful library for data manipulation and analysis in Python. It provides data structures (primarily Series and DataFrame) to efficiently handle and manipulate structured data. Pandas enables tasks such as loading data from various file formats (CSV, Excel, SQL databases), cleaning and preprocessing data, filtering, merging, transforming, and summarizing data. It's widely used for data wrangling and exploration.
- **Seaborn** is a statistical data visualization library built on top of Matplotlib. It provides a high-level interface for creating attractive and informative statistical graphics. Seaborn simplifies the process of creating complex visualizations like scatter plots, bar plots, histograms, box plots, and more. It also supports color palettes, themes, and additional statistical plotting capabilities.
- **Matplotlib** is a versatile 2D plotting library in Python. It provides a wide range of functions to create static, interactive, and animated visualizations. While it can be used to create basic plots, it can also be customized to create complex visualizations. Matplotlib is the foundation for many other visualization libraries and tools.
- **TQDM** is a Python library used for adding progress bars to the code, particularly in loops or functions where we want to track the progress of an operation.
- **Utils** these are external modules used to organize and encapsulate various functions and utilities.
  - The **eda** module contains functions and utilities related to exploratory data analysis.

- The **clustering** module contains functions for implementing clustering algorithms such as K-Means and Affinity Propagation.
- The **visualization** module, includes functions for creating various types of plots and visualizations to help present and understand the results.
- The **dimensionality_reduction** library contains functions related to dimensionality reduction techniques such as PCA.

These libraries are essential for various data analysis, data preprocessing, and data visualization tasks in Python.

4 Results

The first step is to import the libraries for this project, as seen on Figure 1.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from utils import eda, clustering, visualization, dimensionality_reduction
```

Figure 1: *Python libraries.*

Next, we import the dataset to be used and display our data, as shown in Figure 2.

```
Total instances: 240
Total features: 7
     Temperature             L          R    A_M  Color Spectral_Class  Type
0       0.183600      0.002400     0.1700  16.12    Red              M   0.0
1       0.035627      0.000500     0.1542  16.60    Red              M   0.0
2       0.017367      0.000300     0.1020  18.70    Red              M   0.0
3       0.022622      0.000200     0.1600  16.65    Red              M   0.0
4       0.000000      0.000138     0.1030  20.06    Red              M   0.0
..           ...           ...        ...    ...    ...            ...   ...
235     0.972150  374830.000000  1356.0000  -9.93   Blue              O   1.0
236     0.759307  834042.000000  1194.0000 -10.63   Blue              O   1.0
237     0.181025  537493.000000  1423.0000 -10.73  white              A   1.0
238     0.191692  404940.000000  1112.0000 -11.23  white              A   1.0
239     0.944352  294903.000000  1783.0000  -7.80   Blue              O   1.0

[240 rows x 7 columns]
```

Figure 2: *Values from the dataset.*

Next, we obtain the type of data on our dataset, Figure 3.

```
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Temperature     240 non-null    float64
 1   L               240 non-null    float64
 2   R               240 non-null    float64
 3   A_M             240 non-null    float64
 4   Color           240 non-null    object
 5   Spectral_Class  240 non-null    object
 6   Type            240 non-null    float64
dtypes: float64(5), object(2)
```

Figure 3: *Types of data.*

The next step involves checking our dataset for any missing values, Figure 4.

```
Temperature       0
L                 0
R                 0
A_M               0
Color             0
Spectral_Class    0
Type              0
dtype: int64
```

Figure 4: *Missing values.*

Now, we proceed to acknowledge the distribution for our dataset, Figures 5, 6, 7, 8, 9, 10 & 11.

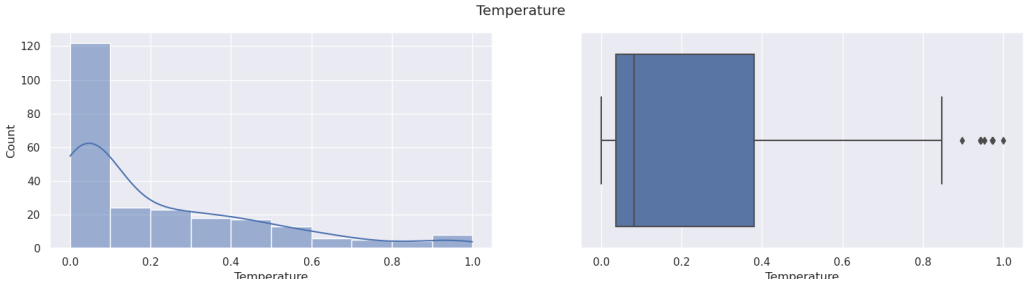

Figure 5: *"Temperature" distribution.*

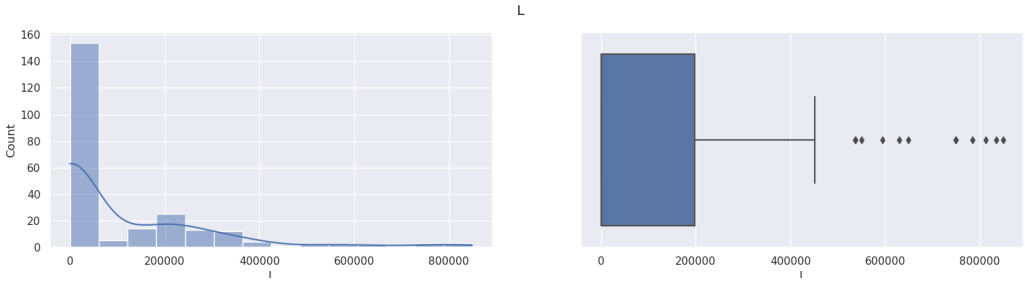

Figure 6: *"L" Distribution.*

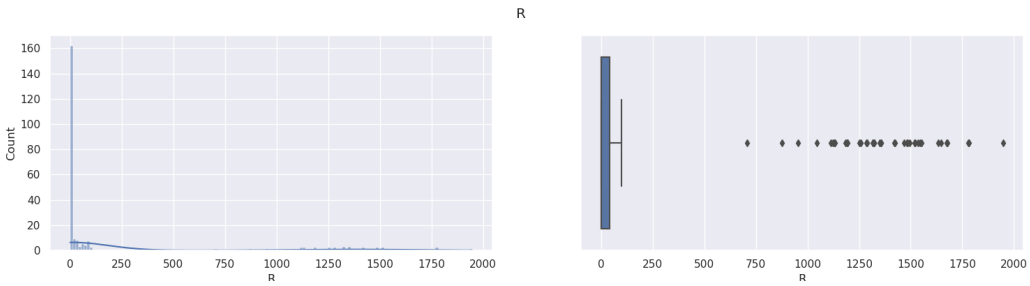

Figure 7: *"R" Distribution.*

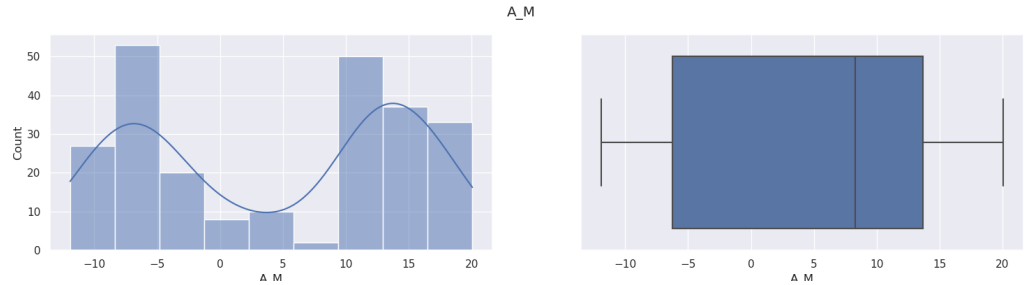

Figure 8: *"A_M" Distribution.*

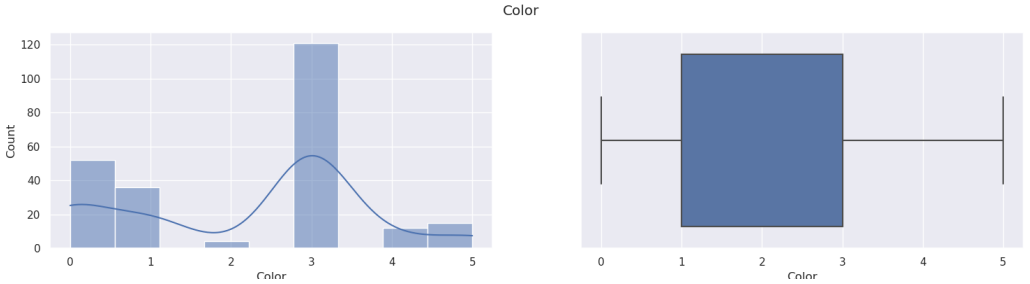

Figure 9: *"Color" Distribution.*

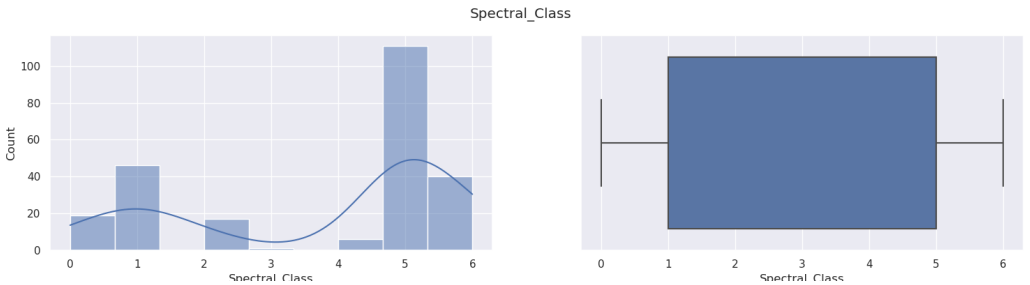

Figure 10: *"Spectral_Class" Distribution.*

The next step, is to acknowledge the balance in our ”Spectral-Class” feature, shown Figure 12.

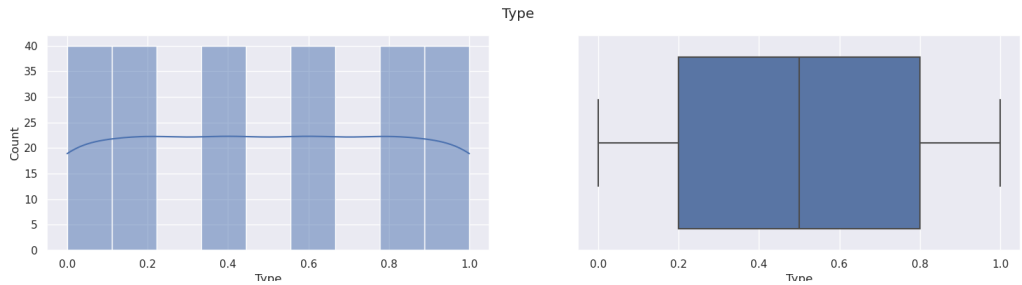

Figure 11: *"Type" Distribution.*

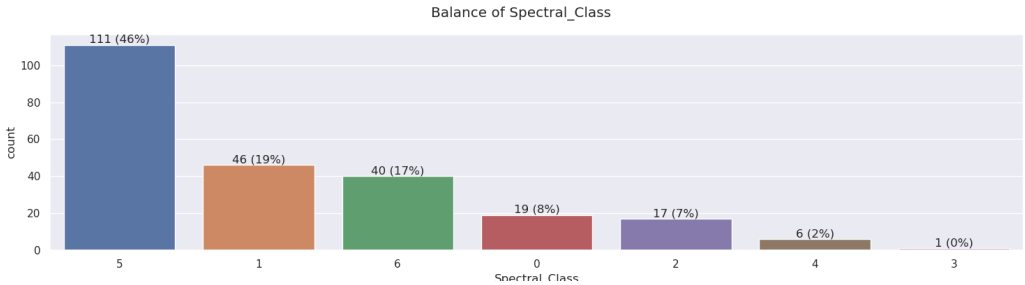

Figure 12: *"Spectral_Class" Balance.*

We will reduce dimensionality using *PCA*, in order to do this, first we need to obtain the relevance of our attributes, Figure 13 & 14.

	relevance
Temperature	9.999940e+01
L	6.004332e-04
R	1.513458e-07
A_M	5.818875e-09
Color	1.249800e-10
Type	2.766176e-11

Figure 13: *Relevant features.*

	Temperature	L	R	A_M
0	0.183600	0.002400	0.1700	16.12
1	0.035627	0.000500	0.1542	16.60
2	0.017367	0.000300	0.1020	18.70
3	0.022622	0.000200	0.1600	16.65
4	0.000000	0.000138	0.1030	20.06
5	0.023673	0.000650	0.1100	16.98
6	0.299545	0.000730	0.1270	17.22
7	0.017367	0.000400	0.0960	17.40
8	0.018681	0.000690	0.1100	17.45
9	0.019994	0.000180	0.1300	16.05

Figure 14: *Relevant features.*

Next, we obtain the subspace of our principal components, as shown on Figure 15.

	PC1	PC2
0	0.001995	-0.060812
1	0.000051	-0.041719
2	-0.000313	0.024923
3	-0.000242	-0.047176
4	-0.000529	0.033277
5	0.000119	0.005094
6	0.000215	-0.010245
7	-0.000169	0.021982
8	0.000140	0.008326
9	-0.000283	-0.021304

Figure 15: *Relevant features.*

On Figure 16, we can visualize our dataset after reducing dimensionality with *PCA*.

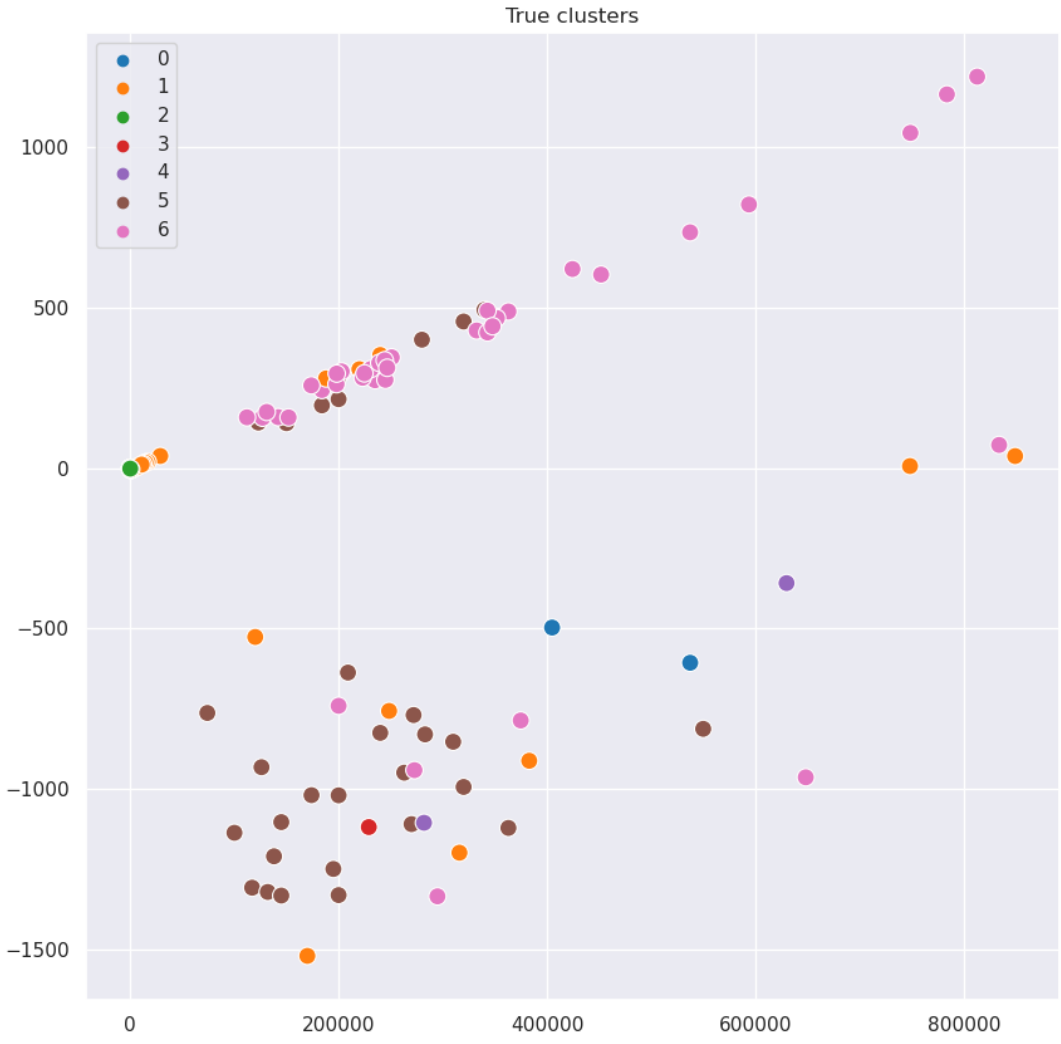

Figure 16: *Dimensionality Reduction with "PCA".*

The next step, it to apply the *K-Means* method for clustering, Figure 17.

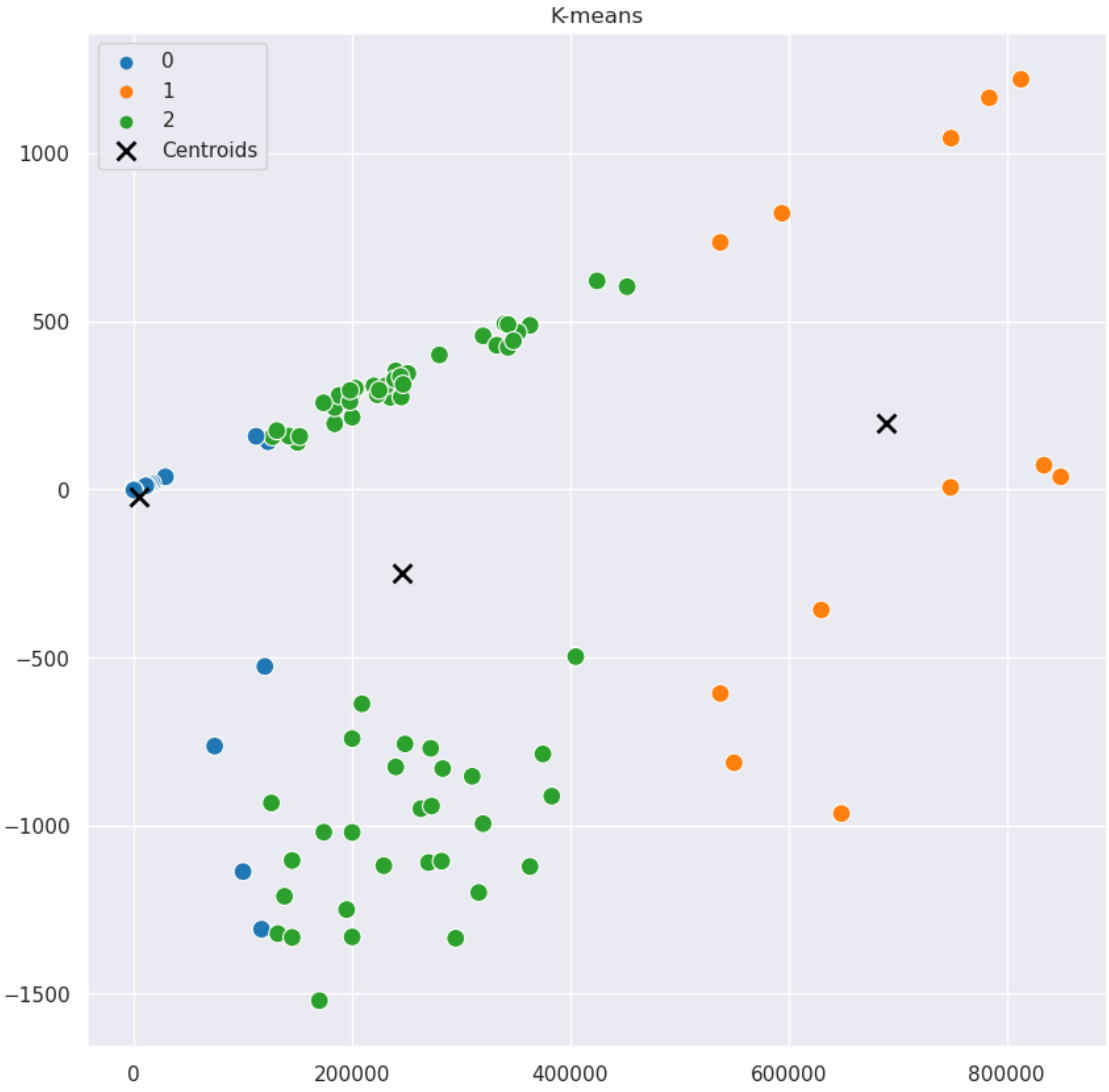

Figure 17: *"K-Means" clustering method.*

For the purpose of comparison, we will employ the *Affinity Propagation* method, as illustrated in Figure 18.

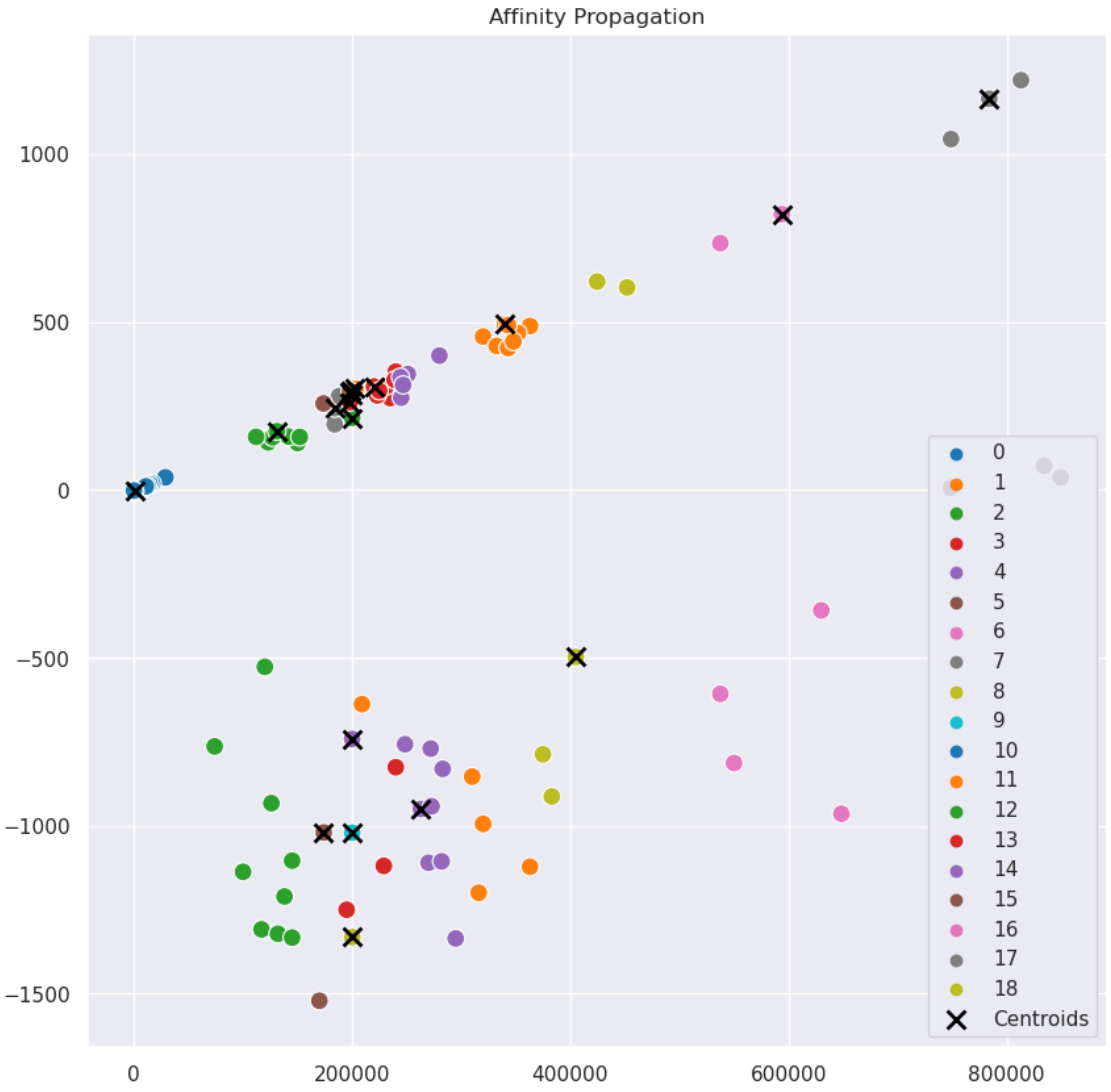

Figure 18: *"Affinity Propagation" clustering method.*

Lastly, we turn to the elbow method as a crucial step in our analysis to ascertain the optimal number of clusters (K) for our dataset, as depicted in Figure 19.

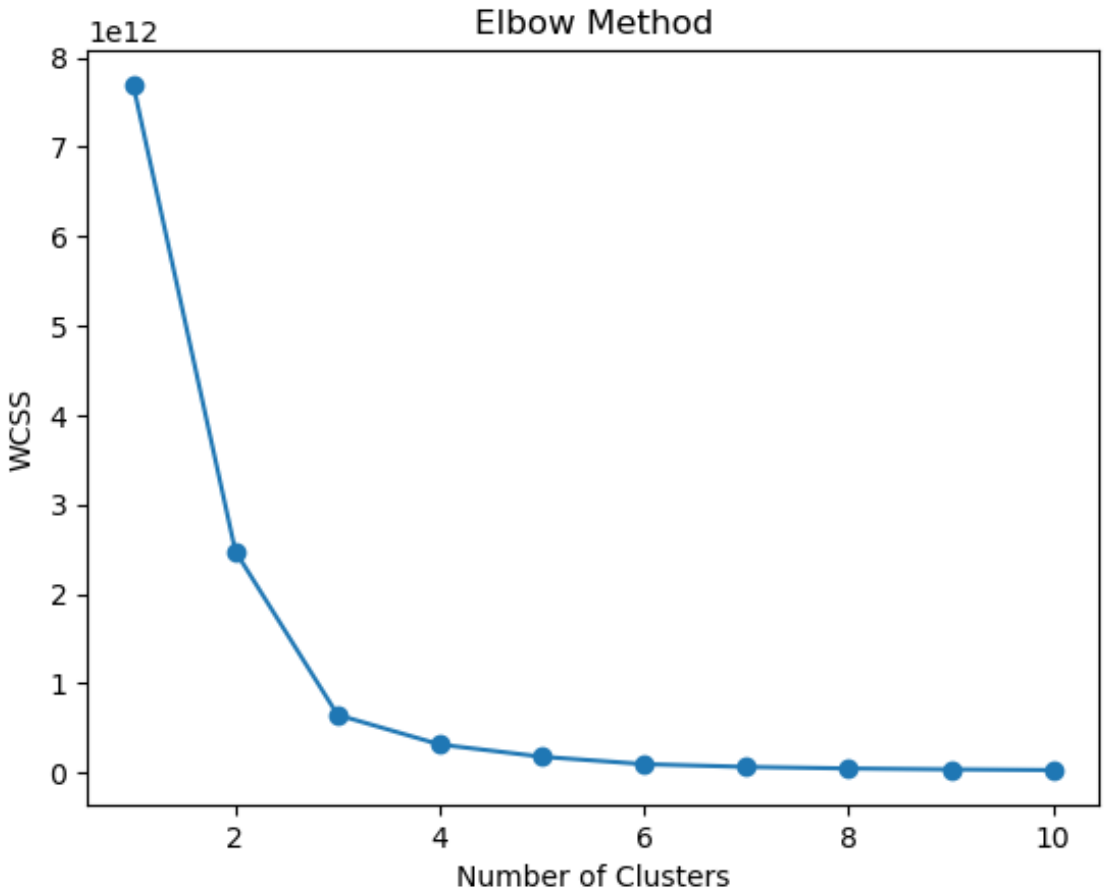

Figure 19: *"Elbow" method.*

5 Discussion

Affinity Propagation and K-Means are both clustering algorithms, but they work in fundamentally different ways and are suitable for different types of data and clustering objectives. Here's a comparison of Affinity Propagation and K-Means:

1. **Clustering Approach**
    - **K-Means** is a partitional clustering algorithm that aims to partition data points into clusters, where each data point belongs to the cluster with the nearest centroid. It minimizes the sum of squared distances between data points and their assigned centroids.
    - **Affinity Propagation** is a clustering algorithm that does not require specifying the number of clusters in advance, it identifies exemplars within the data, which are representative data points that best summarize each cluster. It considers all data points as potential exemplars and identifies the exemplars that best fit the data.
2. **Number of Clusters**
    - **K-Means** requires the user to specify the number of clusters in advance.
    - **Affinity Propagation** automatically determines the number of clusters based on the data and the similarity matrix.
3. **Centroids or Exemplars**
    - **K-Means** assigns data points to the nearest cluster centroid, which serves as the center of the cluster.
    - **Affinity Propagation** identifies exemplar data points that represent each cluster, the exemplars are actual data points from the dataset.
4. **Data Types**
    - **K-Means** works well with numeric data and Euclidean distance-based clustering; it may not perform well with categorical or non-Euclidean data without the appropriate preprocessing.
    - **Affinity Propagation** can be applied to a wide range of data types, including non-numeric data, and can use different similarity measures.
5. **Scalability**
    - **K-Means** is generally more scalable and efficient for large datasets because it involves calculating centroids and updating assignments iteratively.
    - **Affinity Propagation** can be computationally expensive and less scalable, especially for large datasets, due to the need to compute pairwise similarities.

6. **Sensitivity to Initial Conditions**

    - **K-Means** is sensitive to initial centroid positions. Different initializations can lead to different clustering results.
    - **Affinity Propagation** is less sensitive to initial conditions since it considers all data points as potential exemplars.

7. **Use Cases**

    - **K-Means** commonly used for data compression, customer segmentation, image compression, and other tasks where we want to partition data into distinct groups.
    - **Affinity Propagation** is useful when we want to discover the most representative data points in our dataset, which can be valuable in tasks like document summarization, image segmentation, or selecting a subset of data points for further analysis.

The choice between K-Means and Affinity Propagation depends on our specific clustering objectives, the nature of the data, and whether we want to specify the number of clusters in advance. K-Means is a good choice for traditional clustering tasks with a predefined number of clusters, while Affinity Propagation is more suitable when we want to automatically determine clusters and find exemplars that represent them.

The elbow method is a vital component for clustering activities, since it provides a systematic and data-driven approach to deciding the number of clusters, enhancing the effectiveness of clustering algorithms like K-Means in uncovering meaningful patterns within datasets.

In the context of our analysis, opting for the K-Means clustering approach proves to be more suitable. This choice is primarily driven by the observation that the number of clusters generated by the Affinity Propagation method does not align with the natural groupings represented by the classes within our target feature. Additionally, the insights gained from the elbow method deserve due consideration in our decision-making process.

By contrast, K-Means provides a more controlled and congruent clustering solution, harmonizing with the specific objectives of our analysis.

6 Conclusions

This project provided us with a solid foundation in clustering methodologies, equipping us with the skills to uncover meaningful patterns in data. The journey through K-Means and the Affinity propagation method not only enhanced our understanding of unsupervised learning but also highlighted the importance of data preparation and visualization in the data analysis process.

As we move forward in the world of data-driven decision-making, the knowledge gained from this project will prove invaluable in tackling a wide range of data clustering challenges across various domains.

References

[1] W. Shahzad, Q. Rehman, and E. Ahmed, “Missing Data Imputation using Genetic Algorithm for Supervised Learning,” *International Journal of Advanced Computer Science and Applications*, vol. 8, 2017.

[2] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: A review,” *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] M. A. Thafar, M. Alshahrani, S. Albaradei, T. Gojobori, M. Essack, and X. Gao, “Affinity2Vec: drug-target binding affinity prediction through representation learning, graph mining, and machine learning,” *Scientific Reports*, vol. 12, no. 1, pp. 1–18, 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-08787-9

[5] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[6] B. Dincer, “Star Type Classification/NASA,” 2021, Kaggle. [Online]. Available: https://www.kaggle.com/datasets/brsdincer/star-type-classification

[7] L. Pierson, *Data Science for Dummies*. Wiley, 2015.

[8] M. Aceves, *Inteligencia Artificial Para Programadores Con Prisa*. Universo de Letras, 2021.

[9] D. Little, R. Rubin, *Statistical Analysis with Missing Data*, 3rd ed. Wiley, 2019.

[10] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets*. Cambridge University Press, 2014.