CS 6375
ASSIGNMENT

Names of students in your group:

Yaswanth Adari - YXA220006
Venu Gopal Reddy Yendreddy - VXY210021

Number of free late days used: 1

Note: You are allowed a total of 4 free late days for the entire semester. You can use at most 2 for each assignment. After that, there will be a penalty of 10% for each late day.

Please list clearly all the sources/references that you have used in this assignment.

https://medium.com/data-folks-indonesia/step-by-step-to-understanding-k-means-clustering-and-implementation-with-sklearn-b55803f519d6
https://medium.com/swlh/tweets-classification-and-clustering-in-python-b107be1ba7c7
https://archive.ics.uci.edu/dataset/438/health+news+in+twitter

Part II (70 points)

Complete code for part2.py:
https://colab.research.google.com/drive/1yrKKR-E7scJ3KzAJ3AV7dlH1IUmt6kvl?usp=sharing

You will need to perform the following:

1. Choose a dataset suitable for regression from UCI ML Repository: - The data set used for this classifier model is : https://archive.ics.uci.edu/dataset/367/dota2+games+results

A dataset has been imported to a public GitHub repository, and Pandas has been used to read the dataset from the URL. A class named `Kmeans` has been created, and an object named `Kmeans_instnace` has been initialized to refer to the dataset. The URL has been passed to the class.

```
Kmeans_instance = KMEANS("https://raw.githubusercontent.com/YaswanthAd/KNN_Tweets/main/bbchealth.txt")
tweets, power = Kmeans_instance.dataPreprocessing()
```

2. Pre-process your dataset.

In the initial pre-processing phase, several steps were undertaken to refine the textual data. The tweet's unique identifier and timestamp were extracted, streamlining the content for further analysis. Any mentions of usernames, denoted by the '@' symbol, were meticulously eliminated to ensure that the focus remained solely on the text's core message. Similarly, hashtags, indicated by the '#' symbol, were transformed into plain words to facilitate a comprehensive understanding of the topics being discussed. Furthermore, all URLs were removed, reducing any extraneous elements. Finally, a uniform lowercase format was applied to every word, homogenizing the text and paving the way for consistent analysis.

3. Perform K-means clustering on the resulting tweets using at least 5 different values of K and report your results in the format below Note that the sum of squared error is defined as:

We have implemented the KMeans algorithm from scratch, the code presents an implementation of the k-means clustering algorithm, a widely-used technique for grouping data into distinct clusters based on their similarities. The algorithm takes an initial set of cluster points as input, represented by the variable "intial_cluster_points."

The core of the algorithm lies within the "algorithm" function, where the k-means process is executed. The function employs a loop that continues until the cluster points no longer change, indicating convergence. Within each iteration, the algorithm goes through three essential steps to refine the clusters.

Firstly, the "_create_kmeans_ctroid" function initializes an empty dictionary called "kmeans_ctroid," which serves as a data structure to hold the tweets associated with each cluster. Each initial cluster point is used as a key in this dictionary.

The next step is the "_cluster_points" function, responsible for assigning tweets to their nearest cluster points. It iterates through each tweet in the dataset (represented by "bbchealth_data_list") and calculates the Jaccard distance between the tweet and each cluster point. The Jaccard distance is a measure of dissimilarity, and the tweet is assigned to the cluster with the minimum distance. This process ensures that tweets with similar content are grouped together in the same cluster.

Lastly, the "_update_cluster_points" function recalculates the cluster points based on the newly assigned tweets. For each cluster, it finds the tweet that has the minimum average distance to all other tweets in that cluster. These tweets become the updated cluster points. This step is crucial as it helps to refine the position of the cluster points, ensuring that they are in the optimal positions to represent the respective clusters accurately.

Throughout the iterations, the cluster points are updated using the "_update_cluster_points" function, and tweets are assigned to clusters using the "_cluster_points" function. This iterative process continues until the cluster points stabilize, indicating that the algorithm has successfully converged.

The overall objective of this k-means algorithm is to effectively group tweets with similar content, allowing for meaningful and interpretable clusters to emerge. These clusters can provide valuable insights for data analysis, such as identifying trending topics, detecting patterns, or understanding the sentiment of tweets related to specific subjects.

It is worth noting that the Jaccard distance, employed in this algorithm, measures the dissimilarity between two sets and is a suitable metric for comparing the similarity of tweets in this context. Additionally, the code demonstrates a well-structured approach to implementing k-means, showcasing the practicality of the algorithm in real-world applications involving data clustering.

4. Hyperparameter Tuning:

We employed the Optuna optimization library to determine the optimal hyperparameters for our Kmeans model. The hyperparameters in question revolve around the K-values. Following each trial, we recorded the K-values, SSE, and cluster size, appending them to a list. This list was subsequently utilized for visualizing the results. After conducting 20 trials, Optuna adeptly traversed the hyperparameter space and pinpointed the optimal combination of parameters that led to the minimization of the SSE value.

The finest parameters were identified as follows: K: 100, SSE: 170.455

For the purpose of logging, we implemented a logging function to capture the K-values, SSE, and cluster sizes in a file named Kmeans.log. Below, you'll find a sample representation of the tabular data:

Value of K	SSE	Size of Cluster
5	235.15197572696468	1 : 2043 tweets 2 : 551 tweets 3 : 1702 tweets 4 : 119 tweets 5 : 304 tweets
10	203.51499363411392	1 : 136 tweets 2 : 364 tweets 3 : 250 tweets 4 : 1248 tweets 5 : 859 tweets 6 : 446 tweets 7 : 288 tweets 8 : 488 tweets 9 : 623 tweets 10 : 17 tweets
15	181.88338833260616	1 : 736 tweets 2 : 454 tweets 3 : 158 tweets 4 : 105 tweets 5 : 758 tweets 6 : 164 tweets 7 : 209 tweets 8 : 320 tweets 9 : 507 tweets 10 : 362 tweets 11 : 196 tweets 12 : 187 tweets 13 : 165 tweets 14 : 177 tweets 15 : 221 tweets
20	180.56068172433234	1 : 73 tweets 2 : 7 tweets 3 : 170 tweets 4 : 240 tweets 5 : 216 tweets 6 : 252 tweets 7 : 222 tweets 8 : 246 tweets 9 : 137 tweets 10 : 259 tweets 11 : 51 tweets 12 : 229 tweets 13 : 897 tweets

We have graphed the iterations for each 'K' value in the "Kmeans_results.csv" file, which has been submitted.

Conclusion:

This report details our exploration of hyperparameter tuning and the application of K-means clustering to a dataset of tweets, aiming to uncover underlying patterns and group similar tweets together. Our approach involved experimenting with varying 'K' values and evaluating the outcomes using the Sum of Squared Errors (SSE) metric. The results unveil an interesting trend: as we escalated the number of clusters (K), the SSE consistently decreased, signifying improved clustering effectiveness. However, it's crucial to strike a balance between achieving a lower SSE and maintaining the interpretability of the clusters.

One noteworthy limitation of our model is the absence of a definitive technique to determine the optimal 'K' value. While SSE serves as a valuable evaluation metric, it alone falls short. Metrics like the silhouette score or Davies-Bouldin index could offer deeper insights into the clustering quality. Furthermore, comprehending clusters, particularly with higher 'K' values, can prove intricate. To surmount this challenge, future endeavors might explore more advanced cluster evaluation and visualization methods, ensuring that clusters are more intuitive for users to grasp.

Furthermore, our strategy could benefit from incorporating supplementary features, such as user information or embeddings of tweet content. These enhancements could enrich the clustering process, extracting more profound patterns from the data. Yet, an obstacle we grappled with was the presence of outliers in the dataset. Outliers, diverging significantly from the bulk of data points, wield the potential to detrimentally influence clustering performance. Their influence on cluster centroids can distort cluster formation, introducing a level of complexity that demands careful consideration.