Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης
Τμήμα Πληροφορικής
Μπαλακτσής Χρήστος (**3865**), Παναγιωτόπουλος Νέρων Μιχαήλ (**3990**)

Εργασία στις **Αποθήκες Δεδομένων & Εξόρυξη Δεδομένων**

Ερωτήματα 1-5

Ο πηγαίος κώδικας που απαντά στα εν λόγω ερωτήματα αναπτύχθηκε σε γλώσσα **Python** (v3.10) και περιέχεται στο αρχείο main.py, το οποίο απαιτεί ως παράμετρο εισόδου το όνομα/διαδρομή του αρχείου δεδομένων.

Εκτέλεση από γραμμή εντολών σε περιβάλλον Linux:

$ **python3 main.py** <**dataset_file.csv**>

όπου <dataset_file.csv> το όνομα του αρχείου εισόδου.

Ερώτημα 6

Σε πρώτη φάση, ζητήθηκαν από το ChatGPT οδηγίες για τον εντοπισμό συστάδων σε ένα σύνολο δισδιάστατων δεδομένων, **δεδομένου εκ των προτέρων του πλήθους των συστάδων** και λαμβάνοντας υπόψη την ανάγκη χαμηλής πολυπλοκότητας και τη δυνατότητα παραλληλισμού της διαδικασίας.

[user]: *Given a dataset of 2-d points, knowing a priori they are clustered in 5 groups, suggest instructions for an efficient way to detect these clusters, considering low time and space complexity and the ability to parallelize the procedure.*

[gpt]: To efficiently detect clusters in a dataset of 2D points with low time and space complexity, and the ability to parallelize the procedure, you can use the k-means clustering algorithm. Here's a step-by-step guide:

1. **Initialize Cluster Centers:**
   - Randomly select five points from the dataset as initial cluster centers.

2. **Assign Points to Clusters:**
   - For each point in the dataset, assign it to the cluster whose center is nearest using Euclidean distance. This step is computationally simple and can be easily parallelized.

3. **Update Cluster Centers:**
   - Recalculate the cluster centers by taking the mean of all points assigned to each cluster.

4. **Repeat Assign and Update Steps:**
   - Repeat the assignment and update steps until convergence. Convergence occurs when the cluster assignments no longer change significantly or after a certain number of iterations.

5. **Parallelization:**
   - To parallelize the assignment step, you can divide the dataset into chunks and assign points to clusters independently within each chunk. This can be done using parallel computing frameworks like Dask, Spark, or by using parallel processing libraries in programming languages such as Python's multiprocessing.

6. **Optimization Techniques:**
   - Use k-means++ initialization for more robust convergence.
   - Implement early stopping to halt the algorithm if little change occurs between iterations.
   - Consider using a smart convergence criterion, such as monitoring the change in the sum of squared distances between cluster centers.

7. **Choose the Right k:**

- Since you know there are 5 clusters, set k=5. If the number of clusters is unknown, you may need to experiment with different values of k or use techniques like the elbow method.

8. **Handle Outliers:**
   - Consider preprocessing the data to handle outliers, as k-means is sensitive to them. You can use techniques like removing outliers or transforming the data.

9. **Evaluate Results:**
   - After convergence, evaluate the quality of the clusters using metrics like silhouette score, Davies-Bouldin index, or within-cluster sum of squares.

10. **Implement in a Programming Language:**
    - Implement the k-means algorithm in a programming language that supports parallel processing, such as Python with libraries like scikit-learn or PySpark.

By following these steps, you can efficiently detect clusters in your dataset while considering low time and space complexity, and the ability to parallelize the procedure.

Το ChatGPT ξεκινά την απάντησή του προτείνοντάς καταρχήν την τεχνική του αλγορίθμου **k-means**. Πρόκειται για **μια επιστημονικά αναμενόμενη προσέγγιση**, μιας και τόσο για δεδομένο αριθμό συστάδων όσο και για τη δυνατότητα παραλληλισμού ενδείκνυται γενικά η χρήση του. (Jin & Han, 2011)

Δεδομένου του πλήθους *k*, παραθέτει τη γενική μεθοδολογία *k μέσων*, προτείνοντας την τυχαία επιλογή 5 σημείων ως κέντρα των 5 συστάδων, τη σειριακή προσπέλαση κάθε σημείου και την ένταξή του στην κοντινότερη ομάδα και, τελικά, το αναδρομικό επαναϋπολογισμό των κέντρων των συστάδων και τις νέες ενσωματώσεις. Υπολογιστικά, πρόκειται για **χαμηλού κόστους πράξη** και **πράγματι** εύκολα **παραλληλοποιήσιμη**. Μάλιστα, προτείνει -μεταξύ άλλων- τη χρήση του Spark για την κατανομή και παραλληλοποίηση των δεδομένων και διαδικασιών (βλ. και Παράρτημα 1) (Han, 2023).

Περιγράφει ως τερματική συνθήκη του υπολογισμού των συστάδων τη **σύγκλιση**, δηλαδή όταν σταματήσουν να μεταβάλλεται η κατανομή των σημείων, **όπως** δηλαδή **διατείνεται η βιβλιογραφία**. Αυτό, ωστόσο, δεν είναι πάντοτε πραγματοποιήσιμο ή ικανοποιητικά αποτελεσματικό, μιας και εξαρτάται από τη φύση των δεδομένων (πυκνότητα, διακριτότητα κατανομών) και τον αριθμό *k*. Στην απάντησή του, **σωστά προτείνει ως τρόπο περιορισμού του προβλήματος σύγκλισης** τον ορισμό κατάλληλου κατωφλίου στη μέτρηση της μεταβολής της κατανομής, ενώ **σχετικά με την επιλογή του κατάλληλου *k* απέτυχε** να αντιπροτείνει τεχνικές που βελτιώνουν το αποτέλεσμα του k-means, όταν το *k* είναι γνωστό (όπως η λύση που προτάθηκε στο ερώτημα 4 της παρούσας εργασίας), καταδεικνύοντας ότι πρέπει να οριστεί με τη δεδομένη του τιμή (λ.χ. *k=5*) σε κάθε τέτοια περίπτωση. Για τις περιπτώσεις που το *k* δεν είναι γνωστό, **ορθά συνιστά** τον πειραματισμό με διάφορες τιμές σε συνδυασμό με τεχνικές (πχ. elbow) καθορισμού προϋποθέσεων τερματισμού της διαδικασίας.

Σχετικά τον εντοπισμό των οριακών και ανώμαλων τιμών (outliers), προτείνει την εξέταση τεχνικών **προεπεξεργασίας** των δεδομένων για την αφαίρεσή τους. **Σωστά** αναφέρει ότι η ευαισθησία του εν λόγω αλγορίθμου διαστρεβλώνει τα αποτελέσματα κατανομής στις συστάδες, όταν υπάρχουν στο σύνολο δεδομένων τέτοιες τιμές. Διερευνώντας περισσότερο το βήμα αυτό (βλ. Παράρτημα 2), προτείνει μια σειρά από τεχνικές που μπορούν να εφαρμοστούν (όπως Z-score) για την αφαίρεση των οριακών τιμών, αλλά και αλγορίθμους, όπως ο **DBSCAN**, προς χρήση **αντί του k-means**, εκμεταλλευόμενοι τον σαφή εντοπισμό και τη διατήρηση των τιμών αυτών σε ξεχωριστή ομάδα, **αγνοώντας** όμως την αύξηση της πολυπλοκότητας (Shalmoli Gupta, 2017).

Τέλος, το ChatGPT **προτείνει ορθά τη χρήση και αξιοποίηση εμπειρικών τεχνικών και μετρικών**, όπως τον προσδιορισμό συντελεστή σιλουέτας κ.α. για την εκτίμηση του μέτρου επιτυχίας της συσταδοποίησης, προκειμένου να διαπιστωθεί αν χρειάζεται επανάληψη της διαδικασίας ή όχι.

Άξιο αναφοράς είναι το γεγονός ότι η προτεινόμενη προσέγγιση **δε λαμβάνει υπόψιν τη γνώση για τη φύση των δεδομένων** από την πλευρά της πυκνότητας, συνέχειας και διακριτότητας των συστάδων που αυτά κατανέμονται. Δηλαδή, αποτελεί μια αφαιρετική πρόταση του γλωσσικού μοντέλου, η οποία συμπλέει με τη βιβλιογραφία στη

γενική περίπτωση άγνωστων στοιχείων των δεδομένων (*k* και φυσικά χαρακτηριστικά), αλλά δεν καλύπτει -σε πρώτη τουλάχιστον φάση- τις περιπτώσεις που τα στοιχεία αυτά είναι a priori γνωστά, όπως στην περίπτωση της παρούσας εργασίας.

Επεκτείνοντας τη συζήτηση με το ChatGPT, παρατέθηκε λεκτικά/περιγραφικά η οπτική πληροφορία που γνωρίζουμε για τα δεδομένα στην εξεταζόμενη περίπτωση (**διακριτές συστάδες** και **ύπαρξη ανωμαλιών στο χωρικό ενδιάμεσο τους**) (βλ. Παράρτημα 3). Ωστόσο, αγνοεί και πάλι τη γνώση των χαρακτηριστικών των δεδομένων εισόδου στο βαθμό που **αποτυγχάνει να προτείνει ένα συνδυασμό τεχνικών** που να καλύπτει τόσο την -παραλληλοποιήσιμη- συσταδοποίηση όσο και την ανίχνευση των οριακών τιμών. Μάλιστα, χρησιμοποιώντας τη μηχανή του ChatGPT 4 (μέσω του Bing Chat) για την παράθεση screenshot της κατανομής των δεδομένων στον χώρο, η απόκρισή του παρέμενε ίδια στις βασικές γραμμές, μόνο που πρότεινε ως τιμή του *k* το 6, θεωρώντας το σύνολο των outliers ξεχωριστή συστάδα (βλ. Παράρτημα 4). Αυτή η τεχνική **δεν επιβεβαιώνεται από τη βιβλιογραφία**, μιας και η ύπαρξη outliers κατά την εκτέλεση του k-means προκαλεί συνήθως αλλοίωση των συστάδων και στη γενική περίπτωση δε θα ενώσει όλες τις οριακές τιμές σε ξεχωριστή συστάδα, αλλά μάλλον θα ανακαλύψει νέα (Barai & Dey, 2017).

Συγκεντρωτικά, η προτεινόμενη λύση του ChatGPT με βάση τις οδηγίες και επεξηγήσεις που δόθηκαν ακολουθεί στα περισσότερα σημεία της την επιστημονική πρακτική, αγνοώντας όμως σημαντικά την ιδιαιτερότητα του εκάστοτε προβλήματος και υποπίπτοντας σε λογικά σφάλματα. Για γνωστά χαρακτηριστικά δεδομένων, δεν είναι σε θέση να παράξει καινοτόμα (σ.σ. μη προφανή) μεθοδολογία για τον εντοπισμό των συστάδων. Σημειώνει, βέβαια, ότι πρέπει να δίνεται έμφαση στη *γνώση πεδίου* και στην *μορφή των δεδομένων*, αλλά πρακτικά αδυνατεί να εκμεταλλευτεί τις επιπλέον πληροφορίες πάνω σε αυτά. Από την άλλη, σχετικά με την υλοποίηση του k-means απαντάει λανθασμένα σχετικά με τον αριθμό του *k* όταν παρατηρεί τις συστάδες απεικονιστικά, ενώ η θεώρηση του συνόλου των outliers ως ξεχωριστή συστάδα απομακρύνεται από την επιστημονική ορθότητα και δε μοιάζει να αγνοεί πλήρως το πώς ο αλγόριθμος σχηματίζει σταδιακά τις συστάδες.

Παράρτημα 1

Ζητήθηκε από το ChatGPT η πρόταση ενός τρόπου συσταδοποίησης **χωρίς τη χρήση του Spark** για δεδομένο πλήθος συστάδων.

[user]: *Suggest instructions for an efficient way to perform clustering, considering low time and space complexity and distribution techniques without using SPARK, knowing the number of clusters.*

[gpt]: Performing efficient clustering with low time and space complexity, especially in a distributed environment without using Spark, can be achieved through various techniques. Here are instructions for an efficient clustering approach:

1. Choose a Fixed-Number Clustering Algorithm:
   - Opt for algorithms designed for a fixed number of clusters, such as K-means, K-medoids, or Gaussian Mixture Models (GMM).

   1. **K-means Clustering:** If you choose K-means, initialize the algorithm with a predefined number of clusters (k). You may need to run the algorithm multiple times with different initializations to avoid local minima.
   2. **K-medoids Clustering:** If your data has noise and outliers, consider using K-medoids (PAM - Partitioning Around Medoids) instead of K-means. The **pyclustering** library in Python provides an implementation.
   3. **Gaussian Mixture Models (GMM):** GMM is a probabilistic model that assumes data points are generated from a mixture of several Gaussian distributions. It is implemented in the **sklearn** library.

2. Data Preprocessing:
   - Remove irrelevant features and handle missing values.
   - Standardize or normalize the data to ensure that all features contribute equally to the distance metric.

3. Sampling for Large Datasets:

- For large datasets, consider using a representative sample instead of the entire dataset for clustering. This can significantly reduce the computational load.

4. Parallelization Techniques:
   - Implement parallelization to distribute the workload across multiple processors or nodes. You can use parallel programming libraries such as `multiprocessing` in Python for parallelizing tasks.

5. Distributed Computing (without Spark):
   - Utilize frameworks like Dask or Joblib in Python for distributed computing. Dask provides a parallel computing framework that integrates seamlessly with popular machine learning libraries.

6. Optimize Distance Calculations:
   - Use efficient distance metrics and algorithms. For example, if using K-means, utilize the triangle inequality to skip unnecessary distance calculations.

7. Incremental Clustering:
   - If your dataset is dynamic and grows over time, consider using incremental clustering algorithms that can efficiently update clusters without reprocessing the entire dataset.

8. Memory Efficiency:
   - Implement memory-efficient data structures and algorithms. For example, use sparse representations for high-dimensional data.

9. Reduce Dimensionality:
   - Apply dimensionality reduction techniques such as PCA (Principal Component Analysis) or t-SNE (t-distributed Stochastic Neighbor Embedding) to reduce the number of features, which can improve clustering performance.

10. Optimize Hyperparameters:
    - Tune the hyperparameters of your clustering algorithm to achieve better results. This might involve experimenting with the number of clusters, convergence criteria, or other algorithm-specific parameters.

11. Use Approximate Algorithms:
    - Consider using approximate clustering algorithms that provide a good solution with reduced computational cost. Algorithms like Mini-Batch K-means can be faster than the traditional K-means.

12. Implement Data Sharding:
    - If working with distributed computing, shard your data into smaller partitions to distribute the load evenly across nodes.

Remember to thoroughly evaluate the performance of your clustering algorithm using appropriate metrics to ensure that the results meet your expectations. Additionally, the specific requirements of your dataset and the problem at hand may influence the choice of techniques.

Αναφορικά με την αποφυγή του Spark ως εργαλείο υλοποίησης των απαιτητικών αλγορίθμων συσταδοποίησης, προτείνει τη χρήση συστημάτων/βιβλιοθηκών όπως το Dask για την κατανομή των μεγάλης τάξης μεγέθους δεδομένων και την παραλληλοποίηση των διαδικασιών.

Πέρα από τα συστήματα, προτείνει εφαρμογή τεχνικών που **εναρμονίζονται με τη βιβλιογραφία**, όπως η χρήση αποδοτικών δομών δεδομένων (πχ. B-Tree), τεχνικών αυξητικής συσταδοποίησης (για αυξανόμενα δεδομένα) και τεχνικών μείωσης χώρου και χρόνου, όπως PCA και αξιοποίηση γεωμετρικών συσχετισμών (λ.χ. τριγωνική ανισότητα για τις ευκλείδειες αποστάσεις).

Παράρτημα 2

[user]: *What technique can I use to handle the outliers?*

[gpt]: Handling outliers in a dataset is crucial, as they can significantly impact the performance of clustering algorithms like k-means. Here are several techniques you can use to handle outliers:

1. **Remove Outliers:**
   - Identify outliers using statistical methods such as the Z-score or IQR (Interquartile Range). Points that fall outside a certain threshold are considered outliers and can be removed from the dataset. Be cautious not to remove too many points, as it may affect the representativeness of your data.

2. **Transform the Data:**
   - Apply mathematical transformations to the data, such as logarithmic or Box-Cox transformations, to reduce the impact of outliers. This can make the data more suitable for clustering algorithms.

3. **Winsorizing:**
   - Winsorizing involves capping the extreme values of the dataset to a specified percentile (e.g., 95th or 99th percentile). This helps in reducing the influence of extreme values without completely removing them.

4. **Use Robust Clustering Algorithms:**
   - Consider using clustering algorithms that are inherently robust to outliers. For example, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is less sensitive to outliers compared to k-means.

5. **Modify Distance Metrics:**
   - Choose distance metrics that are less sensitive to outliers. For instance, instead of Euclidean distance, you can use Manhattan distance (L1 norm) or Minkowski distance with a lower exponent.

6. **Data Scaling:**
   - Scale your data using robust scalers like RobustScaler in scikit-learn. Robust scaling is less affected by outliers compared to standard scaling methods.

7. **Isolation Forest:**
   - Implement isolation forest algorithms, which are specifically designed for outlier detection. These algorithms build an ensemble of decision trees to isolate outliers efficiently.

8. **Use Ensemble Methods:**
   - Utilize ensemble methods such as Random Forests for clustering. Ensemble methods can be more robust to outliers as they consider multiple models.

9. **Manual Inspection:**
   - Examine the nature of outliers in your dataset. If they represent genuine data points and are not errors, you may choose to keep them and adjust your clustering approach accordingly.

10. **Implement Robust Initialization:**
    - If using k-means, consider using a robust initialization method like k-means++ to reduce sensitivity to outliers in the initial cluster center selection.

The choice of method depends on the nature of your data and the characteristics of the outliers. It's often beneficial to experiment with different techniques and evaluate their impact on the clustering results. Keep in mind that the goal is not necessarily to eliminate all outliers but to mitigate their impact on the clustering process.

Παράρτημα 3

[user]: *Having a visualization of the clusters, knowing they have discreet boundaries and the same capacity and the existence of outliers in a small capacity in the between space is known, what could be done about the k?*

[gpt]: If there are outliers between the clusters in the 2D space, it adds an additional challenge to cluster detection. In such cases, handling outliers becomes crucial to ensure that they don't unduly influence the clustering process. Here's how you might adapt your approach:

1. **Outlier Handling Techniques:**
   - Apply the outlier handling techniques, such as removing outliers, transforming data, using robust clustering algorithms, or modifying distance metrics to make the clustering process less sensitive to outliers.

2. **Visual Inspection with Outliers:**
   - When visually inspecting the image, pay attention to regions where outliers are present. Consider whether these outliers should be part of a cluster or if they should be treated separately.

3. **Adjust k According to Outliers:**
   - If outliers are present and should be considered as part of the clusters, you might need to adjust the value of k accordingly. The outliers may represent additional clusters or subclusters.

4. **Density-Based Clustering:**
   - Consider using density-based clustering algorithms like DBSCAN. DBSCAN is less sensitive to outliers and can identify clusters based on the density of points, making it more suitable for datasets with varying cluster densities.

5. **Cluster Validation Metrics:**
   - Use cluster validation metrics, such as silhouette score or Davies-Bouldin index, to evaluate the quality of clusters. These metrics can help you assess how well the clusters capture the underlying structure of the data, accounting for outliers.

6. **Outlier Detection Algorithms:**
   - Use dedicated outlier detection algorithms, like Isolation Forest or One-Class SVM, to identify and handle outliers before applying k-means clustering.

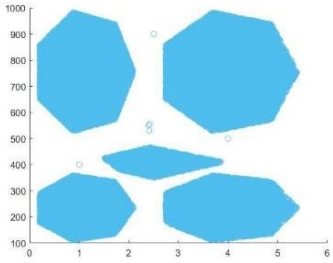
7. **Domain Knowledge:**
   - Leverage domain knowledge to understand whether outliers are meaningful data points or noise. Depending on the context, outliers might represent important information, anomalies, or errors that need special attention.

8. **Experiment and Validate:**
   - Experiment with different approaches, including various outlier handling techniques and different values of k. Validate your results against your understanding of the data and its underlying structure.

In summary, adapt your clustering approach to handle outliers between clusters. Utilize techniques that are robust to outliers, and be open to adjusting the number of clusters (k) based on the characteristics of the outliers and the domain knowledge about your data.

Παράρτημα 4

[user]: *I have this clusters that come from a not labeled dataset. How can I perform k-means clustering knowing the data characteristics shown in the picture.*



[Bing]: Sure, I can guide you on how to perform k-means clustering on your dataset. Here are the steps:

1. **Data Preparation**: Ensure your data is in a suitable format. Each data point should be a vector of numerical values. In your case, each point could be a 2D vector (x, y) representing its position in the scatter plot.
2. **Choose the Number of Clusters (k)**: From your image, it appears there are six distinct clusters. So, you can start with k=6.
3. **Initialize Centroids**: Randomly assign each data point to a cluster and compute the centroid of each cluster. Alternatively, you can select k data points from your dataset at random and use these as the initial centroids.
4. **Iterative Process**: Assign each data point to the cluster that has the closest centroid base on Euclidean distance. After the reassignment, recalculate the centroids. Repeat this process until the centroids stabilize (i.e., they do not change after an iteration) or after a set number of iterations.
5. **Evaluation**: Evaluate the quality of the clustering, you can use metrics like WCSS (Within-Cluster-Sum-of-Squares). Remember, a good clustering has tight clusters (low WCSS) and the data points in the same cluster are very similar to each other.

Remember, k-means clustering is sensitive to the initial placement of centroids. Therefore, it might be beneficial to run the algorithm multiple times with different initial centroid placements and choose the clustering with the best evaluation metric.

Βιβλιογραφικές Αναφορές

Barai, A., & Dey, L. (2017). Outlier Detection and Removal Algorithm in K-Means and Hierarchical Clustering. *World Journal of Computer Application and Technology*, 24.

Han, M. (2023). Research on optimization of K-means Algorithm Based on Spark. *2023 IEEE 6th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC)* (σσ. 1829-1836). Chongqing, China: IEEE.

Jin, X., & Han, J. (2011). K-Meabs Clustering. Στο *Encyclopedia of Machine Learning* (σσ. 563-564). Boston, MA: Springer.

Shalmoli Gupta, R. K. (2017). *Local Search Methods for k-Means with Outliers.* VLDB Endowment.