Route Optimization and Recommendation System

Kaleem Ullah
kaleemullah.ullah@studenti.unitn.it
University of Trento
Trento, Italy

Luca Zanolo
luca.zanolo@studenti.unitn.it
University of Trento
Trento, Italy

Yishak Tadele Nigatu
yishaktadele.nigatu@studenti.unitn.it
University of Trento
Trento, Italy

Hafiz Muhammad Ahmed
hafizmuhammad.ahmed@studenti.unitn.it
University of Trento
Trento, Italy

***Abstract*—The project "Route Optimization and Recommendation System" focuses on addressing discrepancies in logistics operations. Specifically, it deals with the issue of drivers diverging from planned routes and load specifications, and the project's objective is to minimize this discrepancy. The task involves creating a system to recommend optimal standard routes based on drivers' actual behavior patterns and preferences. This includes generating recommendations for the company on standard routes, creating a list of routes for each driver that minimizes diversion, and determining an ideal standard route for each driver. The project emphasizes the development of a large synthetic dataset to test the effectiveness and efficiency of the proposed solution, given the unavailability of real company data.**

***Index Terms*—data mining, recommendation system, route recommendation system, collaborative filtering, and unsupervised learning.**

I. Introduction

A. Introduction to Problem and its importance

In the logistics and transportation sector, efficient route planning and adherence are pivotal for operational success. This project confronts a prevalent issue in this domain: the divergence of truck drivers from planned routes and loads. This deviation not only inflates operational costs but also disrupts the efficiency of logistics management. To tackle this, the project aims to design a system that not only recommends optimal routes but also aligns with drivers' behavioral patterns, thereby bridging the gap between planned and actual routes.

Recognizing the challenges in obtaining real data for the task, the project has made an effort to generate a synthetic dataset encompassing various scenarios, which serves as a robust testing ground for the solution. This approach comprises data analysis and behavior analysis techniques.

This literature presents various techniques that were tried by the team to mitigate the divergence including the choice of effective embedding technique for routes. Analyzing the importance of cities,

B. Highlights of the solution presented

The project is methodically segmented into three distinct sub-tasks. Initially, Task One addresses the variation between standard and actual driver routes. This is accomplished through clustering techniques, which aid in discerning frequently traversed cities and thus facilitate the creation of ideal routes.

Subsequently, Task Two involves recommending standard routes to drivers. This is executed through multiple steps. First and foremost is the creation of a driver profile. First, it was analyzed the likeness of each trip within a route that the driver had performed then aggregated by the similarity of that driver with other drivers to make the driver profile more robust. The aggregation is performed via collaborative filtering. Finally, with the content-based recommendation top 5 routes were computed.

For the single most perfect route, an improvisation was implemented by combining the content-based recommendation system with the greedy search algorithm to find the optimal possible route for the driver.

C. Results Obtained

The initial task involves the exploration of diverse clustering algorithms, with a specific focus on density-based techniques such as **DBSCAN** and distance-based methods like **Kmeans**. After evaluating these approaches, the choice was finalized with **K-means** and **k-medoids**, which share similarities with Kmeans but offer the advantage of using clusteroids (an actual data point). Experimentation was conducted with both Euclidean and cosine distances as measuring tools.

In the second task, the results were assessed using Euclidean and cosine distances to compare driver profiles with standard routes. The presented results display an increase in similarity compared to the originally assigned routes to the drivers.

Finally, for the third task, the alignment of driver profiles with routes was carried out with the consideration of trips that could maximize the similarity between the driver profile and the final route, thereby optimizing route assignment based on driver preferences.

II. Related Work

E-commerce has experienced tremendous growth, but delivery issues remain a major challenge. Consumers now expect quick, convenient, and seamless delivery. To meet these demands, e-commerce companies are enhancing delivery processes using innovative technologies and strategies. Companies that can optimize their delivery operations and exceed customer expectations are well-positioned for future success [1]. Encoding categorical variables is necessary for machine learning algorithms to work effectively. One-hot encoding is a common method for encoding categorical variables, as it is simple to implement and can effectively capture the information contained in the categorical variable. Seger (2018) found that one-hot encoding was the most effective method for encoding categorical variables, outperforming other methods like binary encoding and feature hashing [2].

The research paper "Similarity Measures for Text Document Clustering" by Anna Huang (2008) compares the performance of Jaccard and cosine similarity on a variety of text document clustering tasks. The authors find that Jaccard similarity is generally more effective than Cosine similarity for small datasets, while Cosine similarity is generally more effective for large datasets [3]. Before applying the K-means clustering algorithm, the system utilizes PCA to reduce the dimensionality of the data. This technique improves the computational efficiency, accuracy, and noise reduction of the clustering process, particularly in high-dimensional datasets like taxi trajectory data. The authors validate the effectiveness of this method by applying it to a dataset of over 6 million taxi trajectories, resulting in enhanced clustering outcomes [4]. K-medoids clustering is a powerful and versatile clustering algorithm that is well-suited for a variety of data types and applications. K-medoids clustering selects k data points as cluster representatives, called medoids, and iteratively assigns each data point to the nearest medoid. In each iteration, the medoids are updated to minimize the overall intra-cluster distances. This process is repeated until the medoids no longer change [5]. Mini-batch K-Means (MBKM) is a clustering algorithm that processes data in small batches, making it more efficient and scalable for large datasets. While traditional K-Means processes the entire dataset at once, MBKM significantly reduces memory and time requirements by handling data in smaller chunks. The authors compare MBKM to traditional K-Means on various datasets and consistently find that MBKM outperforms traditional K-Means in terms of efficiency, scalability, and accuracy. MBKM offers promising results for large-scale clustering applications in data mining and machine learning [6].

Collaborative filtering (CF) works by identifying users who have similar preferences and then recommending items that similar users have liked in the past. Collaborative filtering is a great tool for recommending items, but it has some limitations. It can be less reliable if the data is not accurate or if the algorithm is not right for the task. Sometimes, other techniques like content-based filtering are more effective [7]. Content-based filtering is a recommendation system that recommends items to users based on the content of the items themselves, rather than simply relying on past user behavior. Content-based filtering has several advantages, including personalization, scalability, and interpretability. However, it also has some limitations, such as sparsity and difficulty recommending new and innovative items [8].

The paper introduces the silhouette and Davies-Bouldin index as metrics to evaluate the optimal number of clusters for clustering data. These metrics are employed to identify the number of clusters that best represent the underlying structure of the data. The authors apply these metrics to a dataset of mall customer data and determine the optimal number of clusters to be 5. This result aligns with findings from previous studies that have utilized these metrics to assess clustering performance. Silhouette and Davies-Bouldin index serve as valuable tools for evaluating the effectiveness of cluster number selection and enhancing clustering accuracy [9]. Sum of Squared Error (SSE) is also used to evaluate the quality of a clustering result. SSE is a measure of error, while DBI is a measure of compactness and separation of clusters.The method was tested on three datasets and the results showed that it can improve the clustering quality and show better clustering results compared to the method of conventional determining centroid [10].

III. Detailed Steps for solution

A. Data Generation

The problem necessitated the synthetic generation of data due to privacy issues. Consequently, a substantial effort was dedicated to dataset generation, involving the consideration of a combination of 65 different Italian cities and 48 types of merchandise.

- **Standard Dataset Generation**
  Standard routes represent the original data presented by the company for drivers to follow. Routes are generated by randomly selecting 2 to 5 cities, and for each consecutive city pair, a route segment (trip) is created with details on the starting and destination cities and the merchandise to be transported. For each trip in the route, a random subset of merchandise items of varying size between 2 to 4, is selected, with quantities ranging from 1 to 50 per item. The entire process is repeated for the specified number of routes, resulting in a diverse collection of synthetic standard routes. The generated shipments are then stored in a JSON file, providing a basis for subsequent analysis and research into distribution network dynamics. This methodology ensures variability and realism in the simulated routes.
- **Actual Dataset Generation**
  The process of actual dataset generation, focusing on variations of previously generated standard routes, involves a set of methods. Initially, a set of parameters including the list of drivers, the input file path containing standard routes data, the output file path for storing generated variations, the number of variations to be created, the maximum number of additional items to be added, and a set of general variations such as adding items, extending routes, varying merchandise, and removing trips.
  The cities and merchandise lists used for standard route generation are retained. Variations are generated through a series of methods that introduce random modifications to the existing routes:
  1) **Item Addition and Removal:** Merchandise items are randomly added to existing routes, with item type variations randomly chosen from 2 to 4. Items in the route may also be randomly removed, enhancing diversity in merchandise configurations.
  2) **Route Extension and Removal:** Routes are extended by adding extra trips, introducing new cities while excluding those already on the route. Random trips within a route can be removed, ensuring dynamic changes to the overall route structure.
  3) **Merchandise Variation:** Merchandise within a route is subjected to random variations, including the loss of items or changes in their quantities.
  4) **Random Variations Selection:** A random selection of these variations is applied to each standard route, creating diverse scenarios from the previously given variations.

  The resulting dataset of variations is stored in a JSON file, including details such as a unique identifier for each variation as an actual route ID, an assigned driver, an associated standard route, and the modified route itself.

B. Vocabulary

Efficient handling of categorical entities such as driver names, merchandise types, and city names is crucial in the data mining project. To this end, the `Vocabulary` class serves as a comprehensive mapping tool. This class is designed to convert categorical names into unique numerical identifiers (IDs) and vice versa, forming the basis for input to the encoder.

1) Encoding Schemes: The `Vocabulary` class implements three encoding schemes for routes:

1) **List of Encoded Trips**: Each route is encoded as a list of trips. For instance, a route `['from': 'Foggia', 'to': 'Brescia', 'merchandise': 'trash bags': 16, 'spaghetti': 26, 'toilet paper': 44, 'shampoo': 46]` is transformed into `[((56, 46), {6: 16, 34: 26, 42: 44, 29: 46})]`. This format is used in the encoder for constructing route matrices and vectors.
2) **Sequence of Cities**: Trips are represented as an ordered sequence of cities without repetition, e.g., the aforementioned route is encoded as `['Foggia', 'Brescia']`.
3) **Hashable Elements**: Each trip is encoded as a hashable element (`tuple`), enabling the creation of a set of trips considering all configurations of start city, end city, and merchandise. This format facilitates the mapping of trips to IDs for tracking identical trips. The example route would be encoded as `[('Foggia', 'Brescia', ('trash bags', 16), ('spaghetti', 26), ('toilet paper', 44), ('shampoo', 46))]`, and is used to construct the utility matrix.

2) Class Implementation: The `Vocabulary` class is implemented with the following key functions:

- `process_routes`: Processes routing data from JSON files, updating sets of unique entities like cities and merchandise.
- `process_trips`: Reads trip data from each route, updating the set of unique trips.
- `create_vocab`: Generates vocabularies (mappings) for encoding and decoding purposes.
- `encode_route`: Encodes routes in a DataFrame using established vocabularies, converting city and merchandise information into corresponding IDs.

This design allows for the efficient handling and transformation of route data, essential for the subsequent phases of the project.

C. Encoder

The Encoder is the component that transforms the routes encoded by the Vocabulary class into matrices and vectors. This

transformation is primarily handled by the `vectorize()` function.

1) Matrix and Vector Transformation: The `vectorize()` function converts a route into a matrix and a corresponding vector. The vector is essentially a flattened version of the matrix. The dimensions of this matrix depend on the maximum number of sequential trips in a route and the sum of the lengths of the cities and merchandise type vocabularies. For example, with 65 cities, 48 types of merchandise, and a maximum of 7 sequential trips, the matrix shape is 7×113 (65 cities + 48 merchandise types), resulting in a vector of length 791 (7×113).

Each row in the matrix represents a trip from the corresponding route. If a route comprises fewer trips than the matrix's fixed height, the remaining rows are set to zero. The encoding of a trip within a row is divided into two parts:

1) **City Encoding**: The first part, corresponding to the length of the city vocabulary, encodes the 'from' and 'to' cities while preserving order information. Using the respective city vocabulary IDs, this part of the row is indexed, and the value inserted corresponds to the order of the city in the route. For a route with trips from Rome to Milan followed by Milan to Turin, the first two rows in the matrix would be encoded as: 1,2 and 3,4, respectively.
2) **Merchandise Encoding**: The second part of the row encodes the merchandise data. The quantities of each merchandise type are normalized and concatenated to the city vector. This normalization is crucial as it aims to reduce the variance introduced by potentially higher quantity values relative to the ordering values in the first part of the array. By doing so, it ensures a balanced representation between the geographical and quantitative aspects of the routes.

After constructing the matrix and vector for each route, the Encoder calculates a corresponding signature using MinHash with 256 permutations. These signatures are utilized to compute the Jaccard similarity when needed. Additionally, the Encoder manages the application of Principal Component Analysis (PCA) to reduce the vector sizes in the clustering phase.

2) Implementation Details: The `Encoder` class is implemented with the following key methods:

- `vectorize`: Constructs a matrix and vector representation of a route, combining city and normalized merchandise information.
- `apply_PCA`: Applies Principal Component Analysis to reduce the dimensionality of data, adding the results as new columns in a data frame.
- `minhash_signature`: Computes the MinHash signature for a given vector, using the specified number of permutations.

This design enables the effective transformation of route data into a format suitable for analysis and clustering in the data mining project.

D. Routes Encoding Discussion

Various methods were experimented with to encode each route. Initially, a binary encoding scheme was adopted to represent the presence of cities and merchandise along the route. However, this approach proved inadequate as it failed to capture critical information related to the order of cities and the quantity of merchandise. Consequently, a One-Hot Encoding strategy was employed, as detailed in the section **??**. It is important to note that trips within the route were encoded, and then concatenated to form a route. This consideration was retained to preserve the significance of merchandise in each trip within a route.

Example:
Consider a scenario with three available cities (A, B, C) and two types of merchandise (M1, M2). For a trip from A to C carrying merchandise 15 M1 (with a maximum quantity registered of 50), the corresponding encoded versions are as follows:

Binary Encoding: $[1, 0, 1, 1, 0]$

One-Hot Encoding: $[1, 0, 2, 3, 0]$ In the above example, it is evident that the sequential information and the merchandise quantity information are lost in the binary encoding.

E. Summary of encoding Workflow

The route encoding process utilizes both the Vocabulary and the Encoder objects, as described in previous sections. The workflow comprises the following steps:

1) The raw route data is first translated using the Vocabulary class, employing the encoding schema specified in the *Encoding Schemes* subsection.
2) The encoded routes are then processed by the Encoder to produce their matrix representations, as outlined in the *Matrix and Vector Transformation* subsection.

This systematic approach ensures that each route is encoded in a manner that captures all essential data aspects, facilitating the downstream data analysis and machine learning tasks.

F. Utility Matrices

The utility matrix is used to analyze the preferences of each driver for various trips. This analysis is facilitated by the `UtilityMatrix` class, which implements a two-level system designed to transform and refine the matrix at each respective level. The matrices produced at both levels are structured with a row for each driver and a column for each trip. Thus, each row represents the profile of a specific driver, describing ratings for each available trip.

The utility matrix at each level is characterized as follows:

1) **Level 1:** At this level, the utility matrix is generated using the available data. Each row of the matrix represents a driver, and each column represents a trip. The matrix is constructed by comparing each actual route performed by a driver with the corresponding standard route that was requested. This comparison leverages the matrix versions of the routes, constructed as detailed in the section *Matrix and Vector Transformation*. The

process of building this matrix described by algorithm 1.

Algorithm 1 Utility Matrix Level 1 Calculation

$sumMatrix \leftarrow$ zero matrix of size Driver $\times$ Trips
$countMatrix \leftarrow$ zero matrix of size Driver $\times$ Trips
for each $route$ in $routes$ **do**
  for i, $actTrip$ in $actTrips$ **do**
    $cumulatedSimilarity \leftarrow 0$
    $comparisonCount \leftarrow 0$
    for k, $stdTrip$ in $stdTrips$ **do**
      $stdVectorSlice \leftarrow stdVector[k]$
      $actVectorSlice \leftarrow actVector[i]$
      $similarity \leftarrow$ cosineSimilarity$(actVectorSlice, stdVectorSlice)$
      if $similarity == 1$ and $i == k$ **then**
        $cumulatedSimilarity \leftarrow 1$
        $comparisonCount \leftarrow 1$
        break
      end if
      $cumulatedSimilarity \leftarrow cumulatedSimilarity + similarity$
      $comparisonCount \leftarrow comparisonCount + 1$
    end for
    $sumMatrix[driverIdx][actTripIndex] \leftarrow cumulatedSimilarity$
    $countMatrix[driverIdx][actTripIndex] \leftarrow comparisonCount$
  end for
end for
$utilityMatrixLv1 \leftarrow$ zero matrix of size Driver $\times$ Trips
for each $element$ in $utilityMatrix$ **do**
  if $countMatrix[element] \neq 0$ **then**
    $utilityMatrixLv1[element] \leftarrow \frac{sumMatrix[element]}{countMatrix[element]}$
  end if
end for
$normalizedCountMatrix \leftarrow \frac{countMatrix}{\max(countMatrix)}$
return $utilityMatrixLv1$

2) **Level 2:** This level introduces an elaborated version of the Level 1 utility matrix, which utilizes precomputed driver profile similarity matrices. These driver profiles, which will be discussed in detail in the following section, are fundamental to enhancing the predictions made by the utility matrix.
The refinement process at Level 2 involves updating the Level 1 matrix with predicted driver ratings for trips not previously taken by each driver. This prediction is based on the top-k most similar drivers for each driver, as identified in the driver profiles similarity matrix. The top-k similar drivers are precomputed and stored as a map from each driver to a set of k most similar drivers. The prediction algorithm works by iterating through each driver and trip. For trips not previously taken by a driver, the algorithm calculates a weighted average rating based on the ratings of the top-k similar drivers for that trip. This weighted average is computed using the similarity scores from the driver profiles and the ratings from the Level 1 matrix. The process is described in detail by algorithm 2.

Algorithm 2 Utility Matrix Level 2 Calculation

$topKMostSimilar \leftarrow$ Map(Driver: [K top similar Drivers])
$predictedMatrix \leftarrow utilityMatrixLv1$
for each $driverA$ **do**
  $similarDrivers \leftarrow topKMostSimilar[driverA]$
  for each $trip$ in $utilityMatrixLv1$ **do**
    if $predictedMatrix[driverA][trip] == 0$ **then**
      $totalRating \leftarrow 0$
      $totalSimilarity \leftarrow 0$
      for each $driverB$ in $similarDrivers$ **do**
        if $utilityMatrixLv1[driverB][trip] > 0$ **then**
          $totalRating \leftarrow totalRating + (driverProfiles[driverA, driverB] \times utilityMatrixLv1[driverB][trip])$
          $totalSimilarity \leftarrow totalSimilarity + driverProfiles[driverA, driverB]$
        end if
      end for
      if $totalSimilarity > 0$ **then**
        $predictedRating \leftarrow totalRating / totalSimilarity$
        $predictedMatrix[driverA][trip] \leftarrow$ round$(predictedRating,$ precision$)$
      end if
    end if
  end for
end for
$utilityMatrixLv2 \leftarrow predictedMatrix$
return $utilityMatrixLv2$

To summarize, the Level 1 utility matrix provides a foundational understanding of drivers' behaviors by quantifying their deviation from standard routes. The Level 2 utility matrix, on the other hand, extends this analysis by employing collaborative filtering to predict driver preferences for untried trips. This enhanced matrix leverages similarities between drivers' profiles to fill gaps in the dataset, offering a more nuanced and comprehensive view of potential driver behaviors.

G. Drivers profiles similarity

The `DriverProfile` class is used for evaluating the similarities between driver profiles defined by each row of the utility matrices described in the previous section. This class utilizes the Level 1 utility matrix to compute the similarity between each pair of drivers, essentially quantifying the extent to which their behaviors, in terms of trip execution, are similar.

Addressing Dimensionality: A significant challenge encountered during the development of driver profiles was the

high dimensionality of the data. Given the diversity and number of trips, driver profiles could potentially become exceedingly complex, with long vector representations. This complexity necessitates a method to balance the dimensionality of the profiles against the quantity of variance information represented for each element.

PCA for Dimensionality Reduction: To address this, Principal Component Analysis (PCA) is employed for dimensionality reduction. PCA is used to transform the driver profiles into a reduced space with 10 principal components. This reduction ensures that the profiles retain the most significant variance aspects while being compressed into a more manageable form to estimate similarities.

Similarity Computation: Post-dimensionality reduction, the similarity between drivers is calculated. In this reduced feature space, the `DriverProfile` class generates profiles using either cosine similarity metric. The resulting driver profile matrix encapsulates the similarities between drivers, reflecting how analogous two drivers are in terms of their trip preferences and patterns.

Code Implementation: In the implementation of the `DriverProfile` class, PCA is applied to the utility matrix, reducing its features to 10 principal components. Subsequently, the similarity between drivers is computed and stored, rounding off to a 4 decimal.

This profiles similarity matrices will be used to refine the level one utility matrix into the level two utility matrix, as anticipated in 2. This process now will be described more in detail in the next section.

H. Level 2 Matrix: Application of Collaborative Filtering

In the construction of the Level 2 utility matrix, a collaborative filtering approach is adopted to enhance the rating of the utility matrix, considering drivers that behave similarly. With this assumption, the rating for some of the trips that a driver has never performed is estimated.

Leveraging Driver Similarities: The foundation of the deployed collaborative filtering method lies in the use of precomputed driver profile similarity matrices. Using the function `getTopKIndices` and the profile similarity matrix is possible to identify the top-k most similar drivers for each driver. Then using only the ratings from the top-k most similar drivers, for each driver it can be inferred the preferences based on the collective experiences of the most similar ones.

Predicting Unknown Ratings: The core of the Level 2 matrix computation involves predicting ratings for trips that a driver has not previously undertaken. For each such trip, the algorithm computes a weighted average rating based on the ratings given by the top-k similar drivers for that trip. The weighting factor here is the degree of similarity between the drivers, as determined by the driver profile similarity matrix. This ensures that ratings from more similar drivers have a greater influence on the predicted rating.

Algorithmic Approach: The predictive algorithm iterates over each driver and assesses trips that they have not taken. For each of these trips, it aggregates the ratings from similar drivers, weighted by their respective similarity scores. The resulting aggregated rating forms the predicted rating for that trip for the driver in question. This process is described in algorithm 2.

I. New Feature for Clustering

In the pursuit of enhancing the representation of each trip in the data analysis, a new feature is introduced based on the average ratings given to each trip by drivers. This feature is derived from the level 2 utility matrix and is incorporated into the existing matrix representation of each route, consequently extending the corresponding vector representation. The addition of this feature effectively increases the dimensionality of the route matrix representation by one; a matrix with an original shape of $N \times M$ will, after this addition, have a shape of $N \times (M + 1)$.

Retrieving the Score: The new feature is composed by the mean of the non-zero rating in each column of the level 2 utility matrix multiplied by the sum of the total number of times that a trip has been executed. In this operation, the counts for each trip are normalized between 0 and 1 to have a more balanced impact on the trip's average ratings.

Incorporating the Feature into Matrices: The calculated score is then integrated into each route's matrix representation. This integration process involves appending the score as an additional column to the existing route's matrices.

Implementation: The implementation of this feature involves two primary functions:

1) `trip_evaluation`: This function evaluates trips based on the utility matrix and associated counts.
2) `add_trip_level_feature`: This function adds the trip-level feature, derived from the trip rankings, to the route matrices. It iterates through each route, updating the route's corresponding matrix and vector representations to include the new feature.

By integrating this new feature into the route representation based on aggregated driver preferences for each trip.

J. Task 1 - Approach for Generating New Standard Routes

Objective: The primary goal of Task 1 is to generate new standard routes based on actual ones. This is achieved by applying clustering models to the vector representation of routes, which is enhanced with the new trip rating feature.

Clustering Approaches: Two clustering methods, KMeans and KMedoids, were employed. The KMeans approach is a standard method for finding clusters in Euclidean spaces and serves as the baseline model. KMedoids, on the other hand, is used to ensure that clusters are built around already existing routes, thereby grounding the analysis in practical, real-world scenarios.

Data Preparation process reminder: The process begins with encoding the routes using the 'vocabulary' and 'encoder' objects. Following this, the Level 2 utility matrix and driver profiles are computed. Subsequently, the matrix and vector representations of each route are enhanced with the new trip

rating feature. All of these components are build as described in the previous sections.

Model Optimization:

Determining Optimal Number of Clusters: The best number of clusters was initially determined by assessing the model's performance across a range from 2 to 100 clusters. This was achieved by plotting metrics such as the sum of squared distances (SSE) and identifying the 'elbow point' in the graph. *Hyperparameter Tuning:* With the number of clusters established, the model parameters were fine-tuned using grid search. This process involved exploring various combinations of parameters to maximize the silhouette score while minimizing the Davis-Bouldin score.

Performance Evaluation: The evaluation of clustering performance involved the use of the silhouette score, the Davis-Bouldin score, and SSE. The SSE metric was particularly crucial in the first phase for selecting the best cluster number. The silhouette and Davis-Bouldin scores provided additional perspectives on cluster quality, focusing on entropy and separation, respectively.

Clustering Models: The KMeans model is sourced from 'sklearn', while the KMedoid model is from 'sklearn extra'.

The clustering settings are determined through a two-step process. Using the optimal settings, clustering is performed on the enhanced vector representation of each route. The generated cluster labels are then associated with each route. This association is key for tracing the correspondence between the original vector, the enhanced vector, and the assigned cluster label, an important aspect of the route generation processes.

1) *Generating New Standard Routes from medoids:* The new standard routes are derived from the medoids of each cluster identified by the KMedoid model. These routes correspond to actual routes already present in the dataset, providing a first part for the new standard routes.
2) *Generating New Standard Routes from Centroids:* In this approach, new standard routes are generated by considering the mean vector of each cluster formed by the KMeans model. The mean vector is calculated using the vector representations of routes within a cluster, excluding the trip rating feature. This ensures that the generated routes are centered around the core characteristics of the cluster's routes.

   The process for generating a route from a mean vector is as follows:

   a) For each cluster, calculate the mean vector of the routes' vector representations within that cluster. This is done by averaging the vectors (excluding the trip rating feature) of all routes in the cluster.
   b) Then for each mean vector for each cluster translate it to a readble route as described by algorithm 3.

Algorithm 3 is designed to generate new standard routes from cluster centroids. The first function, **TransformMerchandiseValues**, translate the values of the merchandise vector into their final quantities. This function operates under the assumption that merchandise quantities were initially normalized by dividing them by the maximum quantity; therefore, it multiplies these normalized values by the same maximum quantity to restore them to their original scale. In this process, values below 1 are set to zero, while all others are rounded to the nearest integer.

Then, the main algorithm focuses on identifying the highest values in the initial segment of the vector, which represent the starting and ending cities for each trip. It extracts the two maximum values from this segment: the first maximum value corresponds to the destination city, while the second maximum

Algorithm 3 Generation of New Standard Routes from Cluster Centroids

function TRANSFORMMERCHANDISEVALUES($values$)
  $values \leftarrow values \times$ merchandiseMaxQuantity
  for each $value$ in $values$ **do**
    $value \leftarrow \max(\text{int}(value), 0)$
  end for
  return $values$
end function

$tripVectorLength \leftarrow citiesCount + merchandiseTypeCount$
$newRoutes \leftarrow$ empty list

for each $cluster$ **do**
  $meanVector \leftarrow$ mean($clusterRoutesVector$)
  $trips \leftarrow$ empty list
  for i in range($maxTripNumberOfTrips$) **do**
    $[lowIndex, upperIndex] \leftarrow [i \times tripVectorLength, (i+1) \times tripVectorLength]$
    $[cityStartIndex, cityEndIndex] \leftarrow$ Determine Start and End Cities from $meanVector[lowIndex : upperIndex]$
    if $cityStartIndex == cityEndIndex$ **then**
      break
    end if
    $merchCountsVector \leftarrow$ TRANSFORMMERCHANDISEVALUES($meanVector[upperIndex - merchandiseTypeCount : upperIndex]$)
    $merchandise \leftarrow$ Non-zero items from $merchCountsVector$
    $trip \leftarrow \{'from' : cityStartIndex, 'to' : cityEndIndex, 'merchandise' : merchandise\}$
    Append $trip$ to $trips$
  end for
  $newRoute \leftarrow \{id : cluster, route : trips\}$
  Append $newRoute$ to $newRoutes$
end for

indicates the city of origin. This assignment is based on the ordinal values assigned to each city during the route encoding procedure, where the starting city is always given a lower ordinal value compared to the ending city. For instance, if the cities in a trip are A and B, in the city portion of the vector, city A would be represented by 1 and city B by 2.

By reconstructing information in this manner, the algorithm produces a new standard route for each cluster centroid. These routes, combined with the routes identified by Medoids, form the recommended routes for the company. They are stored in 'recStandard.json' in the output folder.

K. Task 2 - Approach

The goal is to create for each driver a list of standard routes in an order where the topmost route is the one with the least divergence from the driver's actual route.

The approach to retrieve the top 5 routes minimizing divergence for each driver is a two-step process:

1) Calculate the similarity between each actual route performed and the requested standard route using Jaccard and cosine similarities. The Jaccard similarity relies on the route signature (see Section **??**), and the cosine similarity uses vectors, not enhanced with the feature.
2) Compute a weighted preference score for each standard route for each driver by calculating the average similarity of each standard route with each actual route in which that route is requested. The weight is defined by the number of times a route has been executed.

This average similarity represents how closely a driver follows each requested standard route, on average.

Algorithm 4 Drivers' Favourite Routes Calculation

$uniqueDrivers \leftarrow$ List of unique driver identifiers from merged data
Initialize $driverProfiles$ as an empty list
for each $driver$ in $uniqueDrivers$ **do**
    Extract data for $driver$ from merged data
    Calculate average similarity for $driver$
    Aggregate and calculate statistics for each standard route assigned to $driver$
    Normalize counts and calculate preference scores for each standard route
    Select top N routes based on preference scores
    Append driver's profile including average similarity and top N routes to $driverProfiles$
end for
return $driverProfiles$ as a DataFrame indexed by driver

Algorithm 4 focuses on determining the most suitable routes for each driver by analyzing their route preferences. It works with *mergedData*, a dataset that merges standard routes with actual routes based on matching identifiers. This join ensures that each actual route is paired with the corresponding requested standard route. The algorithm calculates preference scores for the standard routes requested to each driver, using the average similarity to the actual routes taken. These scores help in identifying the top standard routes that best align with the driving patterns of each driver. The top 5 routes for each driver, as determined by these scores, are then saved in the driver.json file in the specified format.

L. Task 3 - Approach

The objective of Task 3 is to generate the most suitable standard route for each driver, minimizing divergence from their actual routes. In this literature, a collaborative filtering approach is employed, utilizing a level 2 utility matrix (refer to Section III-F) to identify efficient and preferred routes.

Methodology:

1) **Extract Trip Ratings:** Trip ratings for each driver are extracted from the level 2 utility matrix, which includes both actual and estimated values (refer to Section III-H). These ratings are fundamental in determining the sequence of trips in the drivers' routes.
2) **Initialize Route Construction:** For each driver, a route is initialized without any trips. The algorithm considers all available trips, taking into account the starting city and destination of each trip in relation to the current location (the city of the to attribute of the last trip considered).
3) **Greedy Trip Selection:** The algorithm iteratively selects the next trip to add to the driver's route. In each iteration, it chooses the trip with the highest rating that starts from the current city (or any city if it's the first trip). This step ensures that each added trip maximizes the immediate utility for the driver.
4) **Route Expansion:** After selecting a trip, the route is updated by adding this trip, and the current city is set to the destination of the chosen trip. This process continues, with the algorithm repeatedly choosing the next best trip based on the updated current city.
5) **Stopping Criteria:** The construction of the route stops when either no more trips can be added that start from the current city or when the maximum allowed number of trips for a route is reached, this ensures that the route is bounded by a predefined limit on trip sequence length.
6) **Final Route Assessment:** The final route for each driver is a sequence of trips selected based on individual trip ratings, ensuring a high overall rating in alignment with the preferences identified in the level 2 utility matrix.

Algorithm 5 details the greedy method for determining optimal driver routes. Each driver's route is built by iteratively selecting the highest-rated, city-compatible trip. This process continues until reaching the trip limit or exhausting available options.

The approach in Task 3 generates suitable standard routes for drivers, balancing their preferences with operational constraints. Utilizing a greedy algorithm ensures computational efficiency while adhering closely to driver preferences.

IV. Experimental Evaluation

In this literature three experiments have been conducted, with the described approach using three datasets of different sizes:

Algorithm 5 Generate Optimal Routes for Drivers (Greedy Approach)

$maxTripLength \leftarrow$ Max length of trip sequence
$numDrivers \leftarrow$ Number of rows in $matrix$
Initialize $optimalRoutes$ as an empty dictionary
for $driver \leftarrow 0$ to $numDrivers - 1$ **do**
    $currentCity \leftarrow$ None
    Initialize $route$ as an empty list
    Initialize $tripIndices$ as a list of integers from 0 to number of columns in $matrix$ minus 1
    while length of $route$ is less than $maxTripLength$ **do**
        $bestTrip \leftarrow$ None
        $bestRating \leftarrow -1$
        for each $tripIndex$ in $tripIndices$ **do**
            $tripValues \leftarrow vocabulary.id2trip.get(tripIndex)$
            $startCity, endCity \leftarrow tripValues[0], tripValues[1]$
            if $currentCity$ is None or $currentCity = startCity$ **then**
                $tripRating \leftarrow matrix[driver, tripIndex]$
                if $tripRating > bestRating$ **then**
                    $bestRating \leftarrow tripRating$
                    $bestTrip \leftarrow tripIndex$
                end if
            end if
        end for
        if $bestTrip$ is not None **then**
            Remove $bestTrip$ from $tripIndices$
            Append $vocabulary.id2trip.get(bestTrip)$ to $route$
            $currentCity \leftarrow vocabulary.id2trip.get(bestTrip)[1]$
        else
            break
        end if
    end while
    $optimalRoutes[driver] \leftarrow route$
end for
return $optimalRoutes$

1) **dataset A**: with 20 standard routes and 1000 variations of each (20,000 samples).
2) **dataset B**: with 50 standard routes and 1000 variations of each (50,000 samples).
3) **dataset C**: with 20 standard routes and 3000 variations of each (60,000 samples).

The following sections present the results achieved in the various phases.

A. Task 1 - Evaluation

1) Number of cluster analysis: This section reports the charts obtained from the cluster number analysis conducted on the three datasets. For occupy too much pages with all the charts generated here are presented only the three charts used to establish the number of cluster for the configuration that uses **dataset A** with clustering model. All the charts that present the metrics evaluated are available in folder "data/cluster_number_analysis2.

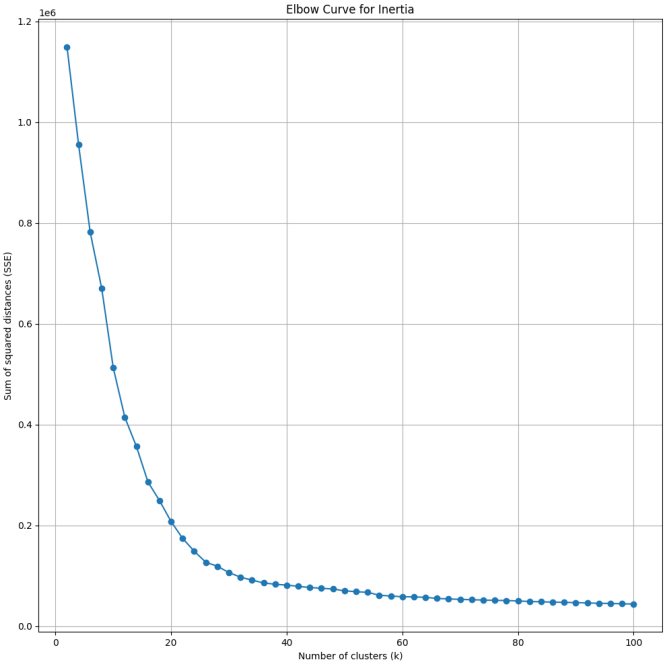

Fig. 1. SSE for different cluster sizes.

2) New standard routes evaluation: The new standard routes generated with the approach described in III-L are evaluated using two similarity matrices for each combination of actual and standard or new standard routes. The two matrices are calculated using cosine similarity and Jaccard similarity. For each dataset, the following similarity matrices are obtained:

1) Cosine similarity matrix between standard and actual routes
2) Jaccard similarity matrix between standard and actual routes
3) Cosine similarity matrix between the new standard and actual routes
4) Jaccard similarity matrix between the new standard and actual routes

Upon comparing the mean and standard deviation of these matrices, it is observed that the mean similarity between the new standard and actual routes is comparable to that of the previous standard. However, a notable difference emerges in the standard deviation values. With the new standard routes, the standard deviation is noticeably reduced. This behaviour is maybe due to the clustering approach, which aims to group similar routes and extract a representative route from each cluster. As these new standard routes are essentially the aver-

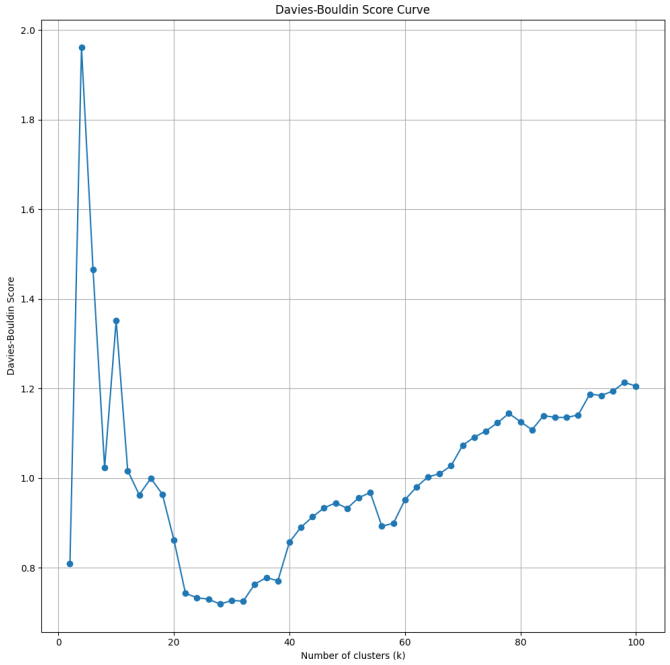

Fig. 2. Davis Putnam score for different cluster sizes.

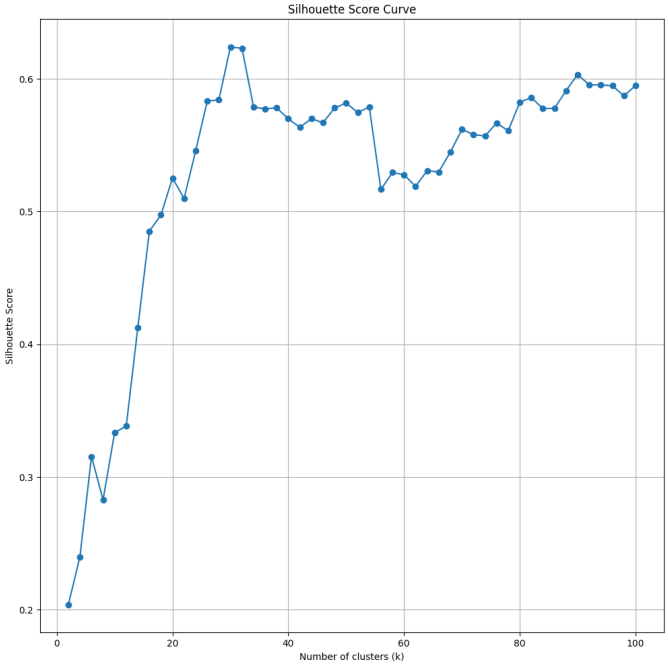

Fig. 3. Silhouette score for different cluster sizes

age of each group, their means remain similar to the previous standard. However, being more tailored to specific groups, they exhibit reduced variability, as indicated by the lower standard deviation values. This finding suggests that while the overall similarity remains consistent, the new standard routes offer a more precise representation of each group, enhancing the specificity and relevance of the route recommendations.

These matrices, showcasing the similarity analyses, are available for all three datasets in the ”data/matrices” folder.

B. Task 2 - Evaluation

The evaluation of this task involves the computation of aggregated similarity scores for the entire set of executed actual routes and the actual routes corresponding to the driver's preferred standard route (generated by task 2). For each driver, all data pertaining to the actual routes is considered, along with the identification of the top five preferred routes. Subsequently, all routes implemented by the driver, derived from both the preferred routes and routes not among the favorites, are sampled. The aggregated similarity score is then computed utilizing the Jaccard similarity metric for all drivers. The results are presented in the following.

1) Dataset A: According to the evaluation result shown in Fig. 4, there is a significant dominance of the estimated aggregated similarity score for the predicted top 5 favorite standard routes over total actual routes as well as the non-favorite standard routes implemented by that driver. However, it can be observed that there are some overlapping areas where the aggregated similarity of the top-ranked favorite routes is less than the aggregated similarities for the total and the non-favorite standard routes.

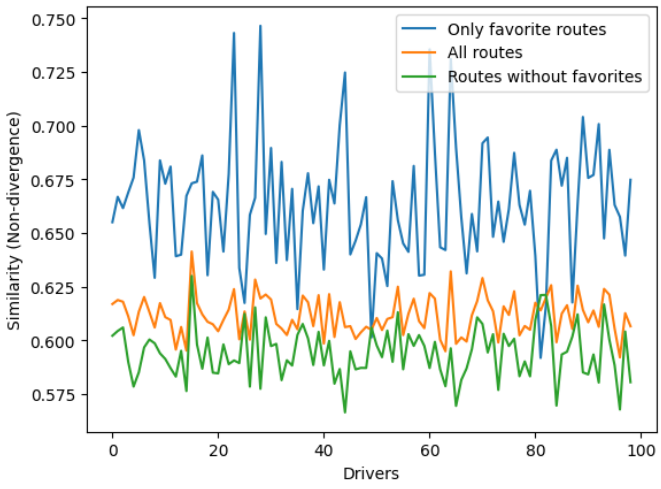

Fig. 4. Quality of Top-5 driver's routes for **Dataset A**.

2) Dataset B: For dataset B also, the total aggregated similarity for the top-ranked standard routes is significantly larger than that of the total and non-favorite standard routes. Based on this assessment, it is noteworthy that the standard route for Dataset B is 50, contrasting with the values of

20 observed for both Dataset A and C. Consequently, the effectiveness of the approach employed for addressing task 2 can be confirmed with this evaluation, demonstrating its capability to accommodate datasets of varying sizes.

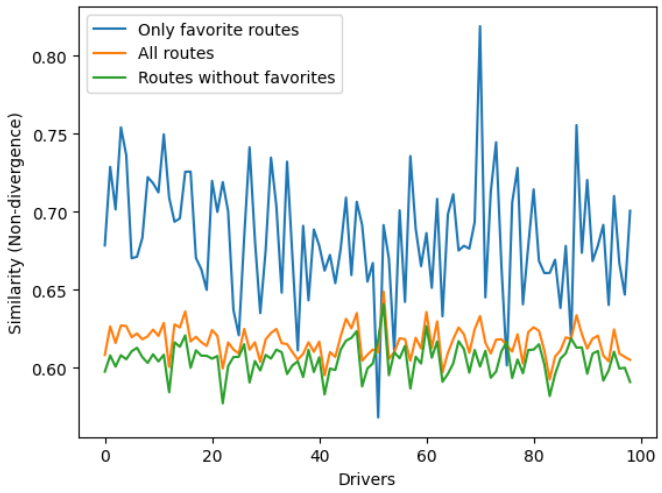

Fig. 5. Quality of Top-5 driver's routes for **Dataset B**.

3) Dataset C: For dataset C, the evaluation shows more spikes and multiple overlapping regions. However, the lines are clearly separated with the only-favorite-routes aggregated similarity score holding dominance.

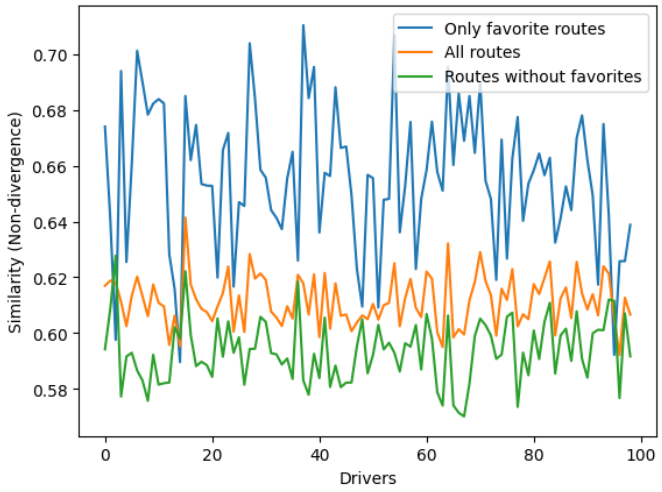

Fig. 6. Quality of Top-5 driver's routes for **Dataset C**.

V. Conclusion

In this study, a route optimization and recommendation system were developed. Given the standard and actual routes implemented by the drivers, the system aims to generate less divergent standard routes for a logistics company. The approach utilizes collaborative filtering and clustering techniques (KMeans and KMedoids) to find optimal routes that closely align with drivers' behavioral patterns and preferences.

The experimental evaluation, conducted across three distinct datasets, reveals the moderate effectiveness of the methodology. The clustering algorithms played a crucial role in creating new standard routes that more accurately reflect drivers' preferences from the implemented actual routes, while the collaborative filtering in the utility matrices refined these recommendations.

An important challenge encountered was the assessment of the quality of the newly generated routes, to understand how to fine-tune the process and interpret the results. The reduced standard deviation in the similarity matrices suggests that the recommendations are consistent with existing routes but also tailored to specific driver groups, thus enhancing the specificity and relevance of the new standard routes. However, the limited relevance of these evaluation results points to the need for more sophisticated techniques to better assess the quality of these routes for the company. This fact is also confirmed by the absence of an evaluation for the ideal standard routes generated for each driver, for task 3. We were not able to find a way to assert the quality of these routes.

In summary, the system presents a dynamic approach to problem-solving, offering flexibility in the main code to easily test different techniques and metrics beyond just Jaccard and cosine similarity. The clustering process aids in identifying the best cluster and hyperparameter settings for the current configuration, facilitating the straightforward generation of new routes. This adaptability underscores the potential of the system as a versatile baseline tool in route optimization and recommendation.

References

[1] A. S. M Mazar, H Belgherri and F. Salihou, "Integrating artificial intelligence and optimization techniques for efficient delivery," *2023 International Conference on Decision Aid Sciences and Applications (DASA)*, pp. 560–564, 2023.

[2] S. . Cedric., "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing," *arXiv preprint arXiv:1807.08507*, 2018.

[3] H. . Anna., "Similarity measures for text document clustering," *Proceedings of the sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008)*, vol. 4, pp. 9–56, 2008.

[4] Y. He, F. Zhang, Y. Li, J. Huang, L. Yin, and C. Xu, "Multiple routes recommendation system on massive taxi trajectories," *Tsinghua Science and Technology*, vol. 21, no. 5, pp. 510–520, 2016.

[5] U. K. Kaur, Noor Kamal and D. Singh., "K-medoid clustering algorithm-a review," *Int. J. Comput. Appl. Technol*, vol. 1, no. 1, pp. 42–45, 2014.

[6] L. D. . A. P. M Chavan, A Patil, "Mini batch k-means clustering on large dataset," *Int. J. Sci. Eng. Technol. Res*, vol. 4, no. 7, pp. 1356–1358, 2015.

[7] J. A. K. Herlocker, Jonathan L. and J. Riedl., *ACM Conference on Computer Supported Cooperative Work*, pp. 241–250, 2000.

[8] V. S. Van Meteren, R. and M., "Using content-based filtering for recommendation," vol. 30, pp. 47–56, 2000.

[9] H. F. H Mulyani, RA Setiawan, "Optimization of k value in clustering using silhouette score (case study: Mall customers data)," *Journal of Information Technology and Its Utilization*, vol. 6, no. 2, pp. 45–50, 2023.

[10] B. J. D. Sitompul, O. S. Sitompul, and P. Sihombing, "Enhancement clustering evaluation result of davies-bouldin index with determining initial centroid of k-means algorithm," *Journal of Physics: Conference Series*, vol. 1235, no. 1, p. 012015, 2019.