Customer Segmentation using K-Means Clustering

Siddhartha Chintalapalli

Supervisor - Maged Elkashlan

210807390Big data science, school of EE & CS  
QMUL,London

*Abstract*—One of the most successful strategies for companies in the Business to Business (B2B) or Business to Consumer (B2C) is by targeting customers by segmenting or clustering them into groups. This helps to find a relationship between the customers so that efficient customer targeting can be done and hence more sales. In this project I have developed a project incorporating similar concepts where I segment the customers based on their persona and personal data to find a relationship between the customers which helps the businesses for efficient campaigning and customer targeting which would lead to higher customer acquisition. I have developed a Web app which uses Angular.js for frontend, Flask as backend and MongoDB as the NoSQL database. The entire project and all of these components have been hosted individually on Heroku Cloud platform while MongoDB is hosted on its own cloud server which is MongoDB Atlas.

Keywords—K-Means clustering, MongoDB, Flask, Heroku, Angular.js, Web Application

# Introduction

Customer segmentation is the process by which we divide the customers into several groups based on characteristics, such as socioeconomic status or behavioral patterns, in order to market to those customers for their particular needs. A group of customers with similar wants and behaviors is referred to as a customer segment. Customers can fall into multiple clusters.

These client division gatherings can likewise be utilized to start conversations of building a showcasing persona. This is on the grounds that client division is regularly used to illuminate a brand's informing, situating and to further develop how a business sells, so showcasing personas should be firmly adjusted to those client sections to be powerful.

This process is widely used globally and in fact is one of the biggest success strategies used and found by Amazon and made Jeff Bezos one of the richest people in the world. Customer segmentation helps to group customers having similar tendencies which can help the business to predict what product a customer is probable to buy and when. This strategy helps to segment customers based on their interest and helps to find out a relationship between the customers. Stating that if a customer has a certain buying pattern and if there is another customer in the same cluster, then his buying pattern might be the same as the other customers in the same cluster.

In this project I have created a Web Application that can be used by the regular users as well as the admin where they can see the clusters that have been formed using the K-Means clustering algorithm. The insights page has been created for such visualizations where the clusters can be seen in the form of graphs. There are certain graphs that can be visualized based on the dataset that contains details like country, gender, birth date, etc. These visualizations have been corelated with the clusters that have been formed to give a clear understanding of how the clustering has happened and what forms the basis for the K-Means clustering that has been performed. Finally, a separate page has been created for the admin only where he/she would be able to add, delete and modify the data and give him the general admin rights. The angular frontend and the flask backend is hosted on Heroku cloud, and the MongoDB database is hosted on its own free server that is the MongoDB Atlas.

Section II describes the related works, section III describes the Methodology, section IV discusses the results and inference and section V talks about the Conclusion.

# Related Works

For the K-means algorithm one of the major drawbacks is the specification of K to be defined before the algorithm is used. This paper [1] audits the existing methods of determining the number of clusters. Factors that influence this determination are then examined and another action to help the choice is proposed. Another article clearly [2] explores the typical k-means clustering algorithm and examines its drawbacks, such as the fact that each iteration of the k-means clustering process requires calculating the distance between each data object and each cluster centre. This lowers the clustering efficiency. In order to answer this question, the enhanced k-means algorithm proposed in this study uses a straightforward data structure to store data in each iteration that would be needed in the subsequent iteration. The literature [3] has a number of k-means extensions that have been proposed.

The k-means technique and its extensions are always influenced by initializations with a necessary number of clusters a priori, despite the fact that clustering in pattern recognition and machine learning is an unsupervised learning process. To put it another way, the k-means algorithm isn't really an unsupervised clustering technique. In this study, we provide an unsupervised learning schema for the k-means method that simultaneously finds the ideal number of clusters and is free of initializations without parameter selection. In other words, we suggest a brand-new unsupervised k-means (U-k-means) clustering algorithm that automatically determines the ideal number of clusters without requiring initialization or parameter choice. The proposed k-means clustering algorithm's computational complexity is also examined. The proposed U-k-means are contrasted with other existing techniques. These positive features of the suggested U-k-means clustering algorithm are actually shown by experimental findings and comparisons. The availability of data has increased, which has raised concerns about clustering algorithms to integrate them coherently and find patterns in large data.

Consequently, there are some situations where Meta-Heuristic algorithms are superior than traditional optimization techniques. Prior to this, optimization problems were thought to be serious limitations in the K-means algorithm, one of the most straightforward clustering techniques. Thus the optimization problem can be readily solved with less extra information. The clustering k-means algorithm and meta-heuristics algorithms are reviewed in this study [4]. Customer mapping is a collection of customer profiling that makes it easier for SMEs to analyse and make decisions regarding the manufacturing of goods, particularly batik sales. To increase the efficiency and effectiveness of the K-Means method in processing vast volumes of data, researchers [5] will combine it with elbow. A localised optimization technique called K-Means Clustering is sensitive to the choice of the beginning position from the middle of the cluster. Therefore, using the middle of a problematic cluster as the starting position for the K-Means Clustering algorithm will provide significant error rates and subpar cluster outputs.

The optimal number of clusters cannot be determined by the K-means method. Elbow thus uses the K-means approach to find the optimal amount of clusters. The elbow approach can yield the same number of clusters K on the amount of varied data based on the outcomes of the process used to determine the best number of clusters. The outcome of figuring out the ideal number of clusters using the elbow technique will be the default for the case study-based characteristic procedure. Machine learning is expanding as quickly as ideas like Big Data and the overall field of data science. The systematic review's [6] goal was to examine academic works dealing with or applying supervised and unsupervised machine learning algorithms in various problem-solving paradigms that were released between 2015 and 2018.

The evaluation method discovered 84 scientific publications that have been published in various journals using the PRISMA components. Despite their metadata stating otherwise, 6 of the 84 articles were published earlier than 2015. The six articles that were included in the final papers were present due to indexing mistakes. In contrast, principal component analysis, hierarchical clustering, and k-means also emerged as the most widely used unsupervised learners. The research also identified additional frequently used algorithms, such as ensembles and reinforcers, and because machine learning and data science are now undergoing improvements, future systematic studies can concentrate on these. The fundamental tenet of unsupervised learning of disentangled representations is that real-world data is produced by a small number of explanatory elements of variation that can be retrieved by unsupervised learning methods.

This article [7] examine recent advancements in the subject realistically and raise several accepted notions. We first demonstrate theoretically that inductive biases on both the models and the data make unsupervised learning of disentangled representations fundamentally impossible. This article [8] intends to investigate the viability of applying the microservice architecture style for frontend development, or micro frontend, in light of the growing relevance of the microservice architecture style in software development. This paper document the definitions along with various advantages and disadvantages of micro frontend against monolithic architecture style, thus trying to establish a best practise concerning when to adopt micro frontend architecture and when to migrate a monolithic to micro frontend. This is accomplished by reviewing a variety of literature concerning frontend development and microservice architecture. Moreover, this paper is also accompanied by an example project, which is a pseudo social media page, with the aim of demonstrating the migration process using the single-spa package, migrating from React framework to micro frontend system utilizing both Angular and Framework for its multiple micro applications.

Another article [9] investigates the prospect of bridging the gap between frontend development and web user experience design using WYSIWYG (What You See Is What You Get) technologies. In this scenario, WYSIWYG tools would produce markup code from UX designer prototypes that could be passed straight to front-end developers, saving time and reducing misunderstandings between the two jobs. Although these instruments have been around for more than twenty years, they are not actually frequently employed for this purpose. Therefore, the purpose of this essay is to ascertain why and investigate what is required of these technologies in order to close the gap between frontend development and online UX design. All across the world, software is always changing and expanding. With these adjustments, the groups responsible for the creation and upkeep of these world-powering systems have made advancements as well.

This [10] project put several features to the test in order to comprehend the impact of the existing state of "best practises," in which industries employ these frameworks. The project made the best use of recently popular frameworks, languages, platforms, and tools for increasing developer productivity, from CI/CD to containerization. The results showed how much the development team's confidence altered as a result of incorporating these features and examining its morale. Numerous team members reported feeling overwhelmed by the amount of technology, according to the research. A well-liked framework for single page web apps is AngularJS. The layout is defined separately in HTML files while the programming logic is implemented in Javascript in AngularJS apps. This division makes it difficult to track data and manage flow in most cases. Authors [11] suggest a technique to distinguish interactions from one another while visualising data and control flow in single page web apps built with AngularJS.

The article provides an approach that makes it easier to comprehend how the application works, to recognise the boundaries of interactions, and to grasp what is updated during interactions and what isn't. The data and control flow of AngularJS apps are typically challenging to follow because of the separation of programming logic and GUI. The paper [12] suggest using a white-box approach that first unifies several concerns into a single interaction diagram that depicts the program's overall data and control flow before segregating user interactions from one another. Our approach provides innovative test coverage requirements for AgnularJS-based single page web applications and aids in better understanding them through the use of interactions. Hundreds of millions of people browse the most popular websites daily thanks to how widely used the internet has grown. The conventional relational database management systems that made up monolithic designs, which were utilised often in the past, quickly lost their ability to support the massive data flow that is so typical these days.

In the meantime, NoSQL databases have evolved to fill in some relational databases' missing features, including as horizontal scaling, eventual consistency, and schema-less design. This study [13] examines and contrasts the implementation of the consistency model on five well-known NoSQL databases: Redis, Cassandra, MongoDB, Neo4j, and OrientDB. All of them offer consistency, at least sporadically, and several even give the choice to encourage high consistency. Because NoSQL databases are so effective at managing the diverse information acquired from IoT contexts, relational databases based on SQL are making way for them. Within the IoT network's devices or in the cloud, such data can be distributedly stored. Due to the need for access control, authentication, and authorisation, security and privacy issues always arise.

In particular, Redis, Cassandra, MongoDB, and Neo4j stores are examined in this study [14] as the state-of-the-art in security and privacy solutions for NoSQL databases. The study also seeks to highlight current issues and potential future research initiatives in the area of database security in the context of the Internet of Things. The goal of the thesis [15] was to examine the various characteristics, benefits, and limitations of two Python web development frameworks. It compares how Django and Flask are used from the perspective of a beginner. The many categories of programming languages and web technologies are presented in the theoretical section of the thesis. However, the study is split into two portions in the application section, each of which examines a different web application framework

# Methodology

## Frontend – Angular.js

For creating the frontend pages, Angular is used which is a TypeScript-based free and open-source web application framework led by the Angular Team at Google and by a community of individuals and corporations. It is widely popular for creating single page apps. It divides the whole frontend html page being served on the browser to several components and provides separation of concern for each of these components. Simply meaning that each component renders independently than the rest of the page and hence its logic and data can be maintained separately from the other parts of the website. Hence, Angular brings a modular and a very robust approach of building frontend applications. Also, in order to subscribe to different APIs, Angular provides the concept of services to have different providers and use dependency injection. So, to sum up the core concepts of Angular include Modules, Components and Services. This may seem a lot but on a whole scheme of things it makes the development of an application very simple. For creating the frontend Angular application, several independent components have been defined as elaborated below:

1. Signup Component- This component handles the functionality of registering a user into the database. In this page the user needs to provide all the required details in order to proceed to the homepage. For providing the login and signup functionality Angular’s reactive forms have been used providing all the necessary validations and helper functions.
2. Login Component- As the name suggests this component handles the login functionality wherein the user enters his/her “CustID” in order to login to the homepage.
3. Home Component- This component provides the navigation functionality which enables the user to navigate throughout the homepage to different tabs.
4. Insights Component- This component opens up the Insight’s tab wherein different graphical visualizations are used to show the K-means clustering data. In order to display graphs, ChartJs is used for plotting different type of graphical representations (Bar, line, Scatter, etc). Also every graphical representation is a Component of its own making the code less redundant and providing reusability to the developer.
5. Profile Component- This component renders the user details which is a common functionality provided in every application where the user can view his/her registration details.
6. Admin Component- This component is only rendered in case of an admin user type. This page will provide admins the functionality to edit the whole Customer database. Here the CRUD (create, read, update, delete) operations are performed and the data is shown in a tabular format.
7. Form Component- This component is a helper component for the admin page wherein in order to perform the CRUD operation, we use a form structure to take the inputs from the user. This also provides reusability of code in order to execute the CRUD operations.

Table 1. Component Description

|  |  |
| --- | --- |
| **Component** | **Functionality** |
| Signup Component | User registration |
| Login Component | User login |
| Home Component | Dashboard- navigation |
| Insights Component | K-means Clustering results visualization |
| Profile Component | Current user details page |
| Admin Component | Admin page with CRUD functionalities |
| Form Component | Dialogue form for CRUD functionality |

The Table 1 sums up all the components used for creating the frontend application and its functionalities. One more important thing to mention is that all the design aspects i.e. the UX/UI of the page is being provided using Angular’s Material UI. It is a very versatile UI library that can be used to create responsive forms, tables, input elements, navbars and many more. Within these UI elements they also provide some quality-of-life features like sorting, pagination and filtering, which are being used in the admin page for better user experience.

## Backend - Flask

Flask microservice is used as the backend for this project. Flask is a micro web framework written in Python. It is delegated a microframework because it does not need specific libraries or tools. It has no data abstraction layer, structure approval, or any other components where prior outside libraries give normal capabilities. The backend has been divided into three files for maintaining the simplicity and modularity of the code. In order to serve the frontend, several endpoints have been created which will cater to the needs of the user for all the tasks. The first file named application.py is the main file containing the flask app along with al the endpoints. Mongo\_db.py contains the functions and the operations with respect to the MongoDB database and the clustering.py contains the code for the K-Means Clustering.

There are seven endpoints that have been created for the proper functioning of the web app and simultaneously maintaining the modularity of the code and the ease of understanding. Table 1 shows the endpoints and its use in a gist. The endpoints are:

1. “/” – This is the test route for the app which can be used at any point of time to test of the app is up and running. It is a GET request. As a response, this route returns a JSON string stating that the app is running.
2. “/new\_user” – This endpoint is used for adding a new user in the database. It is used when a new user signs up and when a data is to be added manually by the admin using the frontend. This is a POST request which accepts a JSON string and adds it in the database. It sends the JSON data to the mongo\_db.py file which first checks if any data with the specific customer ID is already present, if not then it adds the data else it shows a message to the user that the customer id is already present.
3. “/login/” – This is a POST request which accepts the customer id as a JSON string and checks if the data is present in the database. If the data is present, then it returns the complete user data back as a response.
4. “/read/” – This endpoint is used for two operations, one for reading a single record and the other for reading all records. It accepts a customer id from the from the frontend and returns all data of a particular user from the database. If the user id is 0 then it returns all the records from the database as a JSON response.
5. “/delete\_user/” – This POST request endpoint is used to delete user data from the database. It accepts the customer id from the frontend and checks for its presence, if it is found, then it is deleted else it sends a message that the user id is not present.
6. “/modify\_user/<old\_customer\_id>” – This endpoint is used to modify the existing data from the frontend. It is a POST request and can only be accessed by the admin. It accepts the data in two forms, first is that it takes the old customer id as string in the URL, and it also takes the entire new data as a JSON response. Whenever the admin modifies that data, the old customer id sent helps to find the existing record in the database and the after the old data is found, it is deleted to remove any occurrence of the data. Once the data is deleted, the new JSON string containing the new data is added in the database and a response is sent to the frontend regarding the change that has been made.
7. “/k\_means\_clustering/” – This is the last endpoint that helps to feed the K-Means clustering results to the frontend which can be visualized in the form of graphs and charts. It is a GET request which reads all the records, performs data preprocessing and then trains the model and finally send the results to the frontend in the form of a list.

In the data preprocessing, first the unnecessary columns are dropped which creates no meaning while training the data. Then the categorical columns which are in the form of string, are one-hot encoded to convert them into indexes which then can be fed into the model for training. Once the preprocessing is done, a new dataframe is created which can be used directly for training. This dataframe is then trained in the K-Means clustering algorithm, which creates 4 clusters, and the new clusters are attached in the dataframe which sends it back to the frontend as a response.

|  |  |
| --- | --- |
| **Endpoints** | **Uses** |
| “/” | Test route for API functioning |
| “/new\_user” | Add new data |
| “/login” | Verify and send user data for login |
| “/read” | Read a single record or all records |
| “/delete” | Delete existing record |
| “/modify\_user/<old\_customer\_id>” | Modify existing data |
| “/k\_means\_clustering/” | Perform K-Means clustering |

## Authentication and Authorization

For authentication and authorization purpose, JSON web tokens are used. JSON Web Token is a proposed Internet standard for creating data with optional signature and/or optional encryption whose payload holds JSON that asserts some number of claims. The tokens are signed either using a private secret or a public/private key. This is implemented by following three main steps as follows-

1. Generating tokens
2. Define the payload
3. Generate the token using jwt.sign() method with the id of the user data and a secret key.
4. Send the token object as a response whenever the login/register request is received.
5. Store the generated tokens in local storage, this can be done during the login and signup process.
6. Intercepting and verifying the token
7. We create a token-interceptor service that implements the HttpInterceptor module.
8. Now, we clone the request and set the header: (authorization, intercepted token) and call the next method
9. Now, on the server side we create a verifytoken() method that parses the header for the token and verify it using the secret key which was used while generating the token.
10. If, the token is valid then we call the next method.
11. We apply the verifytoken() method on the routes we want to protect from un-authorized users.

## NoSQL Database – MongoDB

The database used for this project is MongoDB which is a NoSQL database. We have used this database because of its simplicity and its ease of cloud deployment as MongoDB has its own cloud database which can be used to deploy the data efficiently on the cloud. The database is initially deployed on cloud using a python script which reads data from the csv file containing 12 columns which are:

* CardID
* CustID
* CustID\_old
* Title
* Country
* Address
* PostalCode
* City
* Mail
* noAd
* BirthDate
* ValidityDate

Out of these, two columns are removed namely CustID\_old and noAd as these two columns do not add value to the data. The columns BirthDate and the ValidityDate needed preprocessing as the data type of these two columns were ambiguous and there was no consistency in the data type. Therefore, these two columns were modified in such a way that it would be interpreted as date objects and then uploaded to the database. Each row was read from the csv file and each cell from each row was converted into a record which was then added to the database. In order to add the data onto the database, the URL is generated from the MongoDB Atlas cloud which is used as endpoint for uploading the data.

## Heroku Deployment

Heroku has been used to deploy both the frontend and the backend. Although both the stacks are deployed on the same platform, the deployment process is different which we will be explaining below:

#### Flask Deployement: Create an application on heroku with a name of our choice and then move on to the files on the system for local changes. Once the entire flask application is ready and tested on localhost, it is to be deployed on heroku for public access. In the repository where all the files are created, two new files are created named requirement.txt and the other one is the Procfile. The requirements.txt contains all the libraries that have been used for the development of the project along with its versions to avoid any conflicts as the interdependency of libraries and its versions is important while working in python. A library called gunicorn is to be installed which acts as the deployment engine for flask applications. Procfile is basically a file that contains commands for heroku to execute once the application is deployed. In Procfile, this line is to be written:

***web: gunicorn application:application***

This line tells Heroku that the application that we are trying to deploy is a web application and this will be deployed using gunicorn server. First “a*pplication”* indicates the name of the file, and the second *“application”* indicates the name of the application inside the file. Once all these files are created, then Heroku client is downloaded, and the following steps are performed:

1. Initialize empty git repository: ***git init***
2. Login to Heroku: ***heroku login***
3. Initialize remote origin to Heroku: ***heroku git: remote -a {name of the project on heroku}***
4. ***git add .***
5. ***git commit -m “Comment”***
6. ***git push heroku master***

Once all these steps are performed, the deployment will start, and it will take some time. Finally, when the deployment is completed, the URL to access the app will be present on the Heroku page where the app is created.

#### Angular Deployement: Similar to the flask deployment, another app needs to be created in heroku with a name of choice. Once the local Angular application is ready, few things needs to be done in order to deploy it on heroku. In pakage.json file add heroku-postbuild command under scripts. Also, if a specific version of Angular/Node is used then it needs to be mentioned by creating an engines field and mentioning the versions for each frameworks.Now in order to run the Angular application we need to create a node application that will run it on a particular port.Now the application is ready to be pushed to heroku using git.One last important thing is to keep the unecessary files in git.ignore file. The steps required to push the code to heroku is same as it was for the flask application which is as follows:

1. Initialize empty git repository: ***git init***
2. Login to Heroku: ***heroku login***
3. Initialize remote origin to Heroku: ***heroku git: remote -a {name of the project on heroku}***
4. ***git add .***
5. ***git commit -m “Comment”***
6. ***git push heroku master***

## Unsupervised Learning (K-Means Clustering)

Unsupervised learning, commonly referred to as unsupervised machine learning, analyzes and groups unlabeled datasets using machine learning algorithms. These algorithms identify hidden patterns or data clusters without the assistance of a human. It is the best option for exploratory data analysis, cross-selling tactics, consumer segmentation, and picture identification because of its capacity to find similarities and differences in information. Clustering, association, and dimensionality reduction are the three basic tasks that unsupervised learning models are used for. Each learning method is defined here, along with examples of common approaches and algorithms for doing them successfully.

#### Clustering: Clustering isa data mining technique which groups unlabeled data based on their similarities or differences. Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information. Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic. A data point may only be included in one cluster according to the grouping method known as exclusive clustering. Another name for this is "hard" clustering. Exclusive clustering is exemplified by the K-means algorithm. Data points can be members of many clusters with varying degrees of membership under overlapping clusters, which is different from exclusive clustering. Overlapping clustering is demonstrated by "soft" or fuzzy k-means clustering.

An unsupervised clustering approach known as hierarchical clustering, sometimes referred to as hierarchical cluster analysis (HCA), can be classified as either agglomerative or divisive. A "bottoms-up technique" is referred known as agglomerative clustering. Its data points are initially separated into distinct groupings, and once one cluster has been obtained, they are progressively combined based on similarity. The following four techniques are frequently used to gauge similarity:

1. Ward’s Linkage
2. Average Linkage
3. Complete (maximum) Linkage
4. Single (minimum) Linkage

An unsupervised method known as a probabilistic model aid in the resolution of density estimates or "soft" clustering issues. Data points are grouped in probabilistic clustering according to how likely it is that they fall under a given distribution. The Gaussian Mixture Model (GMM), one of the most used probabilistic clustering techniques, was developed in the 1960s.

#### Association Rules: A rule-based approach for identifying connections between variables in a particular dataset is called an association rule. Market basket analysis usually employs these techniques, which help businesses comprehend the connections between various items. Businesses may create more effective cross-selling techniques and recommendation engines by better understanding the consumer consumption patterns. Examples of this can be found on Spotify's "Discover Weekly" playlist or in Amazon's "Customers Who Bought This Item Also Bought" section. While several alternative algorithms, including Apriori, Eclat, and FP-Growth, are employed to produce association rules, the Apriori approach is the most frequently employed.

#### Dimensionality Reduction: More data generally produces more accurate results, but it can also affect how machine learning algorithms work (for example, overfitting) and make it challenging to visualize datasets. When a dataset has an excessive amount of characteristics or dimensions, the dimensionality reduction technique is utilized. It keeps the dataset's integrity as much as feasible while reducing the quantity of data inputs to a tolerable level. There are several different dimensionality reduction techniques that may be utilized, and it is frequently used during the preprocessing of data.

#### K-Means Clustering: Centroids are calculated via the K-means clustering algorithm, which then iterates until the best centroid is discovered. It is assumed that the number of clusters is known. The flat clustering algorithm is yet another name for it. The letter "K" in K-means stands for the number of clusters that the algorithm identified from the data. According to this strategy, data points are grouped into clusters so that the total of their squared distances from the centroid is as little as it can be. Important to notice is that less unique data points within clusters result in more identical data points within the same cluster.

The goal of k-means clustering, a vector quantization technique that originated in signal processing, is to divide n observations into k clusters, where each observation belongs to the cluster that has the closest mean (also known as the cluster centroid or cluster center), which serves as a prototype for the cluster. As a result, the data space is divided into Voronoi cells. The geometric median is the only one that minimizes Euclidean distances; k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more challenging Weber problem. For instance, k-medians and k-medoids can be used to find better Euclidean solutions.

Although the problem is computationally challenging (NP-hard), effective heuristic algorithms quickly reach a local optimum. These often follow an iterative refining strategy used by both k-means and Gaussian mixture modeling, which is similar to the expectation-maximization procedure for mixtures of Gaussian distributions. They both employ cluster centers to represent the data, but the Gaussian mixture model allows for different-shaped clusters whereas k-means clustering tries to discover clusters of equivalent spatial dimensions. The popular supervised machine learning technique for classification known as the k-nearest neighbor classifier, which is frequently confused with k-means due to its name, and the unsupervised k-means algorithm have a loose link. New data is sorted into existing clusters by applying the 1-nearest neighbor classifier to the cluster centers produced by k-means. This is sometimes referred to as the Rocchio algorithm or nearest centroid classifier.

# RESULTS AND INFERENCES

After, the complete development for both frontend and backend, the overall flow of the applications is as follows-

1. User login/signup as the landing page for every user shown in Figure1 and Figure 2.

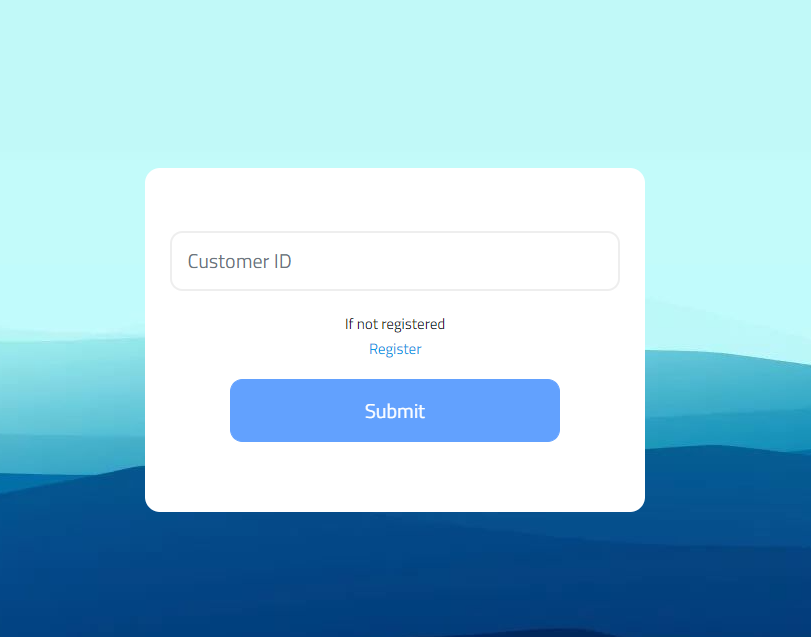


Figure 1. Login page

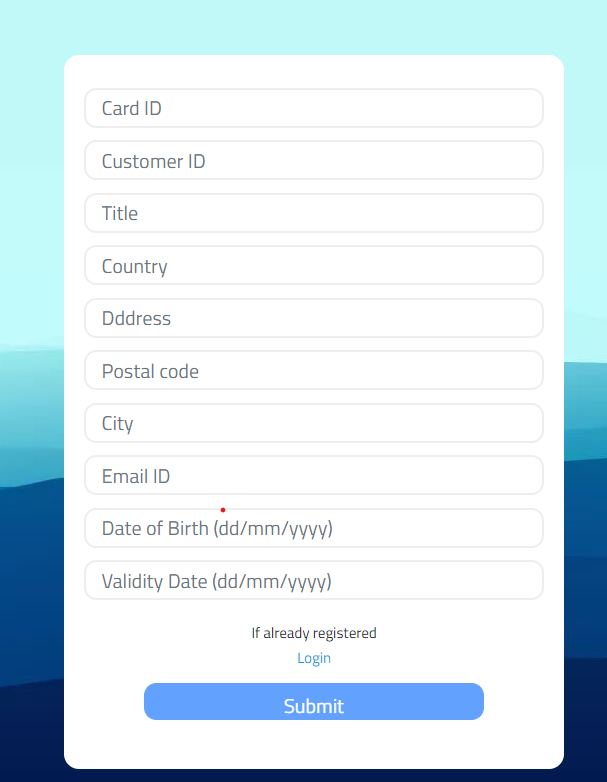


Figure 2. Signup page

1. Insights page- Home page after login or signup operation. Within this page we provide different graphical visualizations of the K-means clustering results as shown in Figure 3, Figure 4, Figure 5, Figure 6 and Figure 7.
2. Cluster Distribution

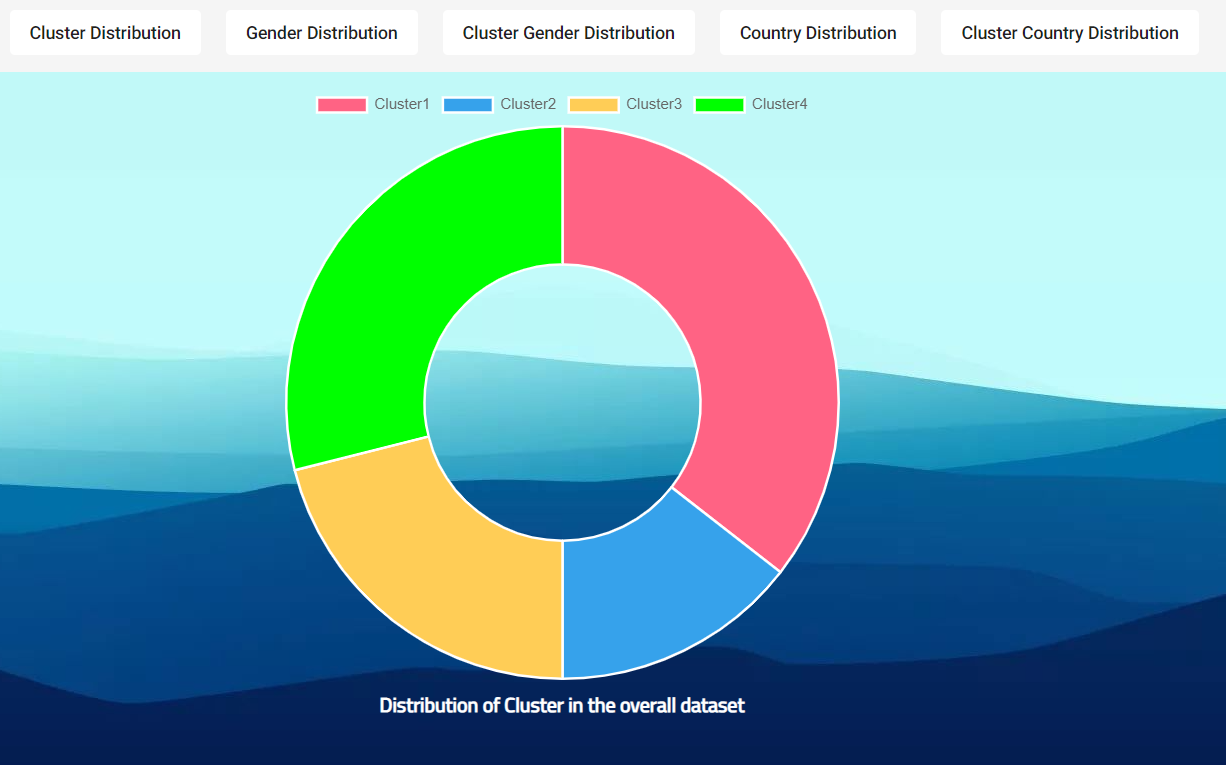


Figure 3. Cluster Distribution

1. Gender Distribution

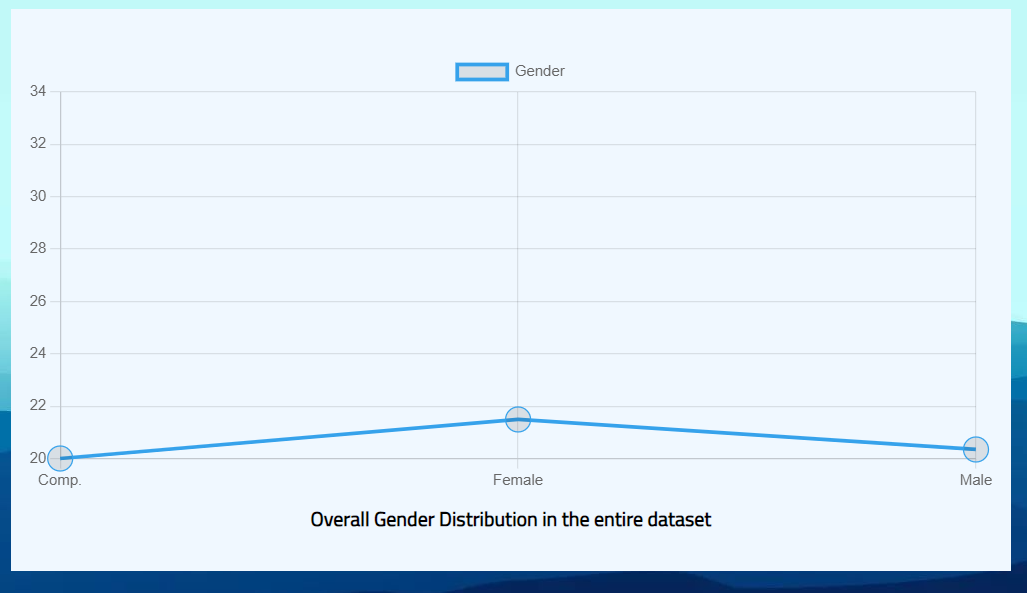


Figure 4. Gender Distribution

1. Cluster gender distribution

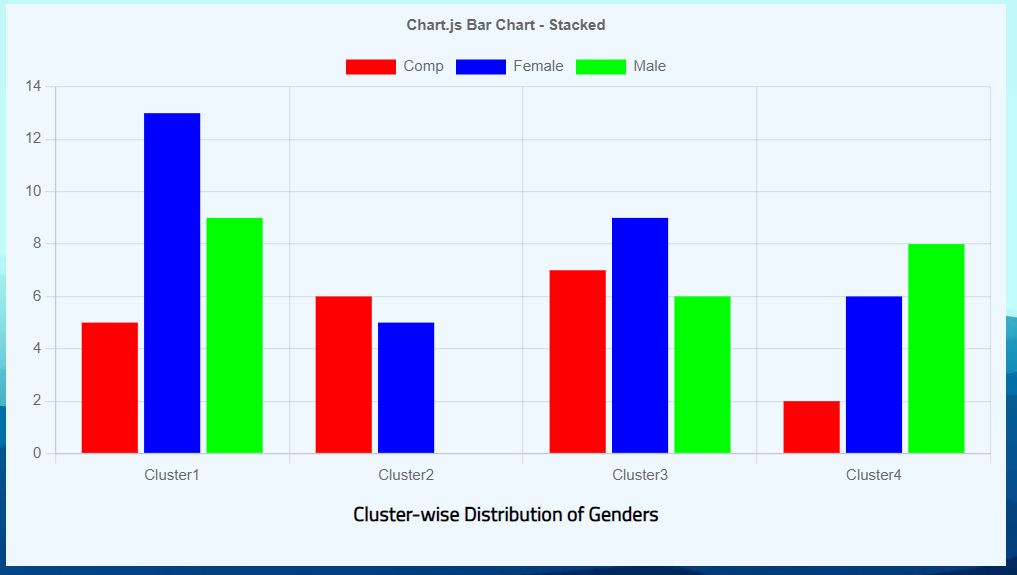


Figure 5. Cluster gender distribution

1. Country distribution

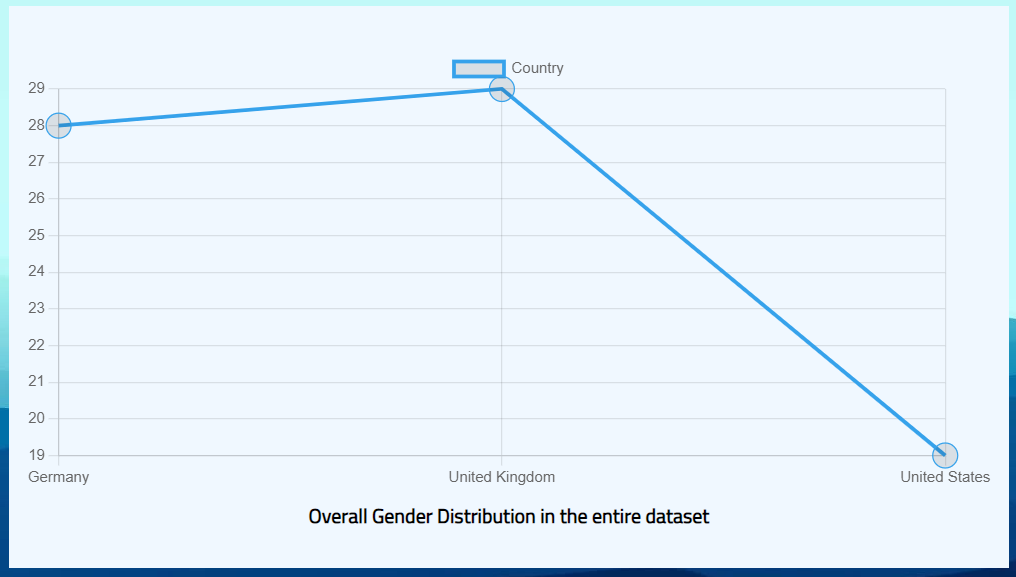


Figure 6. Country distribution

1. Cluster country distribution

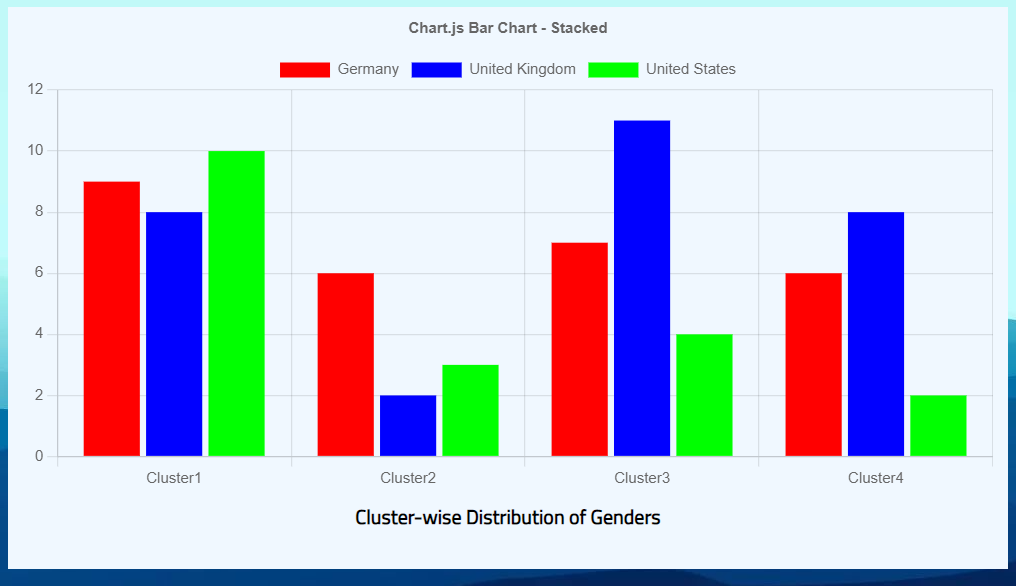


Figure 7. Cluster Country distribution

1. Profile page- Here the current user data is shown in Figure 8

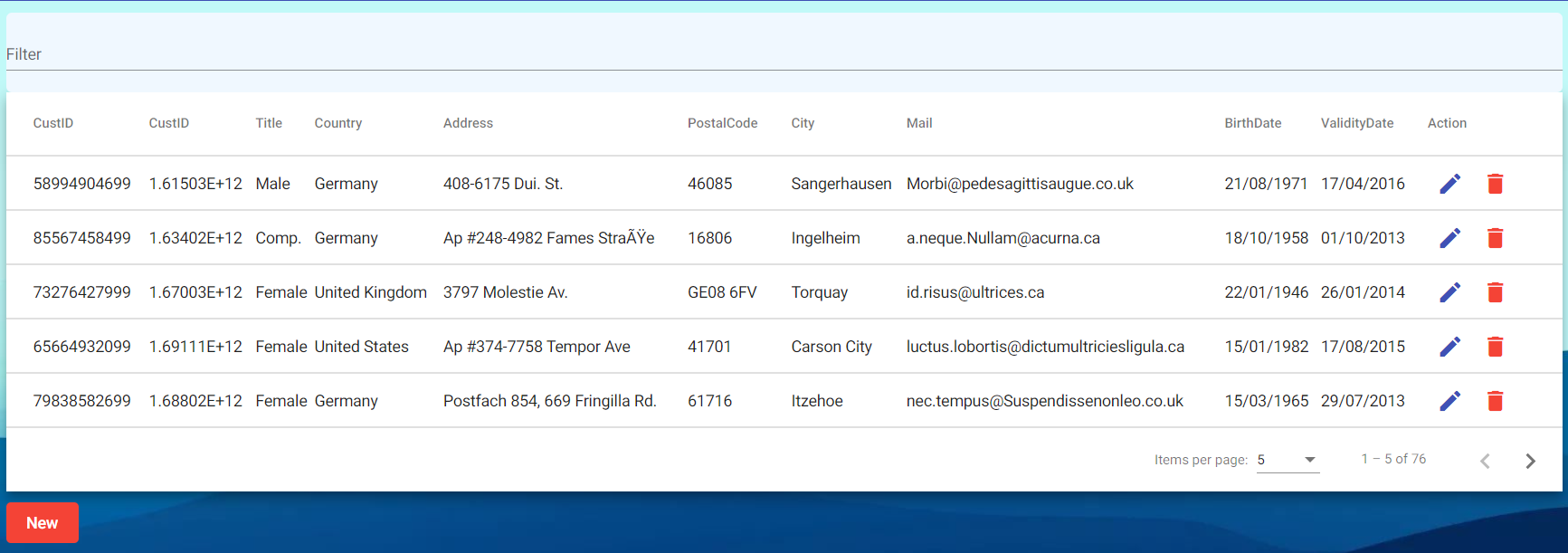




Figure 8. Profile page

1. Admin Page- This page can only be accessed by the admins to add, edit or delete the customer data from the database as shown in Figure 9.

This concludes the overall journey of the user in the web application. The application is robust and provides all the previously discussed functionalities seamlessly.

# Conclusion

The developed application is robust and seamlessly able to show the results of the K-means clustering algorithm based on the dataset used using the various visualizations and clearly demonstrating the relationship between the clusters formed and the dependent parameters in the dataset. This web app takes into account, both the user and the admin aspect which form the basis of any e-commerce application.

All the useful and important functionalities have been implemented but this can also be improved upon by enhancing the aesthetics of the application and quality of life features. The developed application is quite robust and can take in any amount of data without the fear of application crashing as the developed code is quite optimized. The modularity of the code has been taken into consideration with uttermost importance which provides the best performance and makes this web application highly scalable.

##### References

1. [1] D T Pham, S S Dimov, C D Nguyen, “Selection of K in K-means clustering”, First Published January 1, 2005, Manufacturing Engineering Centre, Cardiff University.
2. [2] S. Na, L. Xumin and G. Yong, "Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm," 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, 2010, pp. 63-67, doi: 10.1109/IITSI.2010.74.
3. [3] K. P. Sinaga and M. Yang, "Unsupervised K-Means Clustering Algorithm," in IEEE Access, vol. 8, pp. 80716-80727, 2020, doi: 10.1109/ACCESS.2020.2988796.
4. [4] Alan Fuad Jahwar, Adnan Mohsin Abdulazeez. (2020). META-HEURISTIC ALGORITHMS FOR K-MEANS CLUSTERING: A REVIEW . PalArch’s Journal of Archaeology of Egypt / Egyptology, 17(7), 12002-12020.
5. [5] M A Syakur1, B K Khotimah1, E M S Rochman1 and B D Satoto1, IOP Conference Series: Materials Science and Engineering, Volume 336, The 2nd International Conference on Vocational Education and Electrical Engineering (ICVEE) 9 November 2017
6. [6] Mohamed Alloghani, Dhiya Al-Jumeily, Jamila Mustafina, Supervised and Unsupervised Learning for Data Science, 2020 ISBN : 978-3-030-22474-5
7. [7] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, Olivier Bachem Proceedings of the 36th International Conference on Machine Learning, PMLR 97:4114-4124, 2019.
8. [8] Bui, Son Micro frontend : Microservice implementation on Web Development (2021)
9. [9] I. Enryd, ‘The UX-Frontend Development Boundary : Bridging the Gap with WYSIWYG Tools’, Dissertation, 2019.
10. [10] Sokeland, Eli “Understanding "Best Practices" within software development”, <http://cardinalscholar.bsu.edu/handle/123456789/203115>
11. [11] Gefei Zhang, Jianjun Zhao, “Visualizing Interactions in AngularJS-based Single Page Web Applications”, 2018 - ksiresearch.org
12. [12] Gefei Zhang, Jianjun Zhao, “Scenario Testing of AngularJS-Based Single Page Web Applications”, Current Trends in Web Engineering, 2020, Volume 11609 ISBN : 978-3-030-51252-1
13. [13] Diogo, M.; Cabral, B.; Bernardino, J. Consistency Models of NoSQL Databases. Future Internet 2019, 11, 43. <https://doi.org/10.3390/fi11020043>
14. [14] Sabrina Sicari, Alessandra Rizzardi, Alberto Coen-Porisini, “Security&privacy issues and challenges in NoSQL databases”, Computer Networks, Volume 206, 2022, 108828, ISSN 1389-1286,
15. [15] Ghimire, Devndra (2020), “Comparative study on Python web frameworks: Flask and Django”, https://urn.fi/URN:NBN:fi:amk-2020052513398