Multi-Factor based Nutrition Management System and Recipe Recommendation Engine

***Abstract*— Nutrient management in the context of this research aims to quantize the consumption of essential nutrients in an efficient format such that it leads to a healthy and balanced lifestyle. Increased consciousness towards one's health has recently been in the limelight which creates the need for an intelligent system specially customized for the individual that can analyze your consumption's quality and suggest options that could essentially fulfil your body's need to lead a healthy lifestyle. The research's main goal is to create an intelligent recipe recommender that would aid in the development of a diet that allows all users to make healthy choices in their daily lives while still enjoying food and keeping healthy. The recommender system once implemented as a mobile or web application, can help users who have nutritional deficiencies to maintain a healthy well-balanced diet by suggesting various recipes to the users in video format with additional relevant information which will improve the user's well-being and quality of life.**

Keywords*— *K-Means Clustering, Unsupervised Learning, Euclidean norm, Recipe recommendation, Nutrition

I. INTRODUCTION

Nutrition is defined as the process of providing or obtaining the food necessary for health and growth. Without optimal nutrition, organisms can grow weak, sick and at the very worst can even die. Furthermore, without meeting sufficient nutritional needs, humans can miss developmental milestones and cannot put their bodies through the daily mental and physical tasks.

Recipe recommendation involves ranking and suggesting relevant food products and recipes as outputs by taking various types of inputs such as nutritional values, ingredients, and preferences of the user. This research aims to detect nutritional shortcomings of a user by taking various inputs that can easily be obtained through standard blood test analysis and overcome deficiencies if detected by recommending food and their recipes using an intelligent algorithm.

As computers have become more popular and with the exponential rise in the context of the technical industry, there has been an increase in the number of people who are using computers. As a result of the tremendous development in the use of IT, the landscape around health awareness, living practices, and consumer behaviour's has changed dramatically.

Food consumption plays a central role in human race survival and there exists a need for recipe recommendation systems that can detect nutrition deficiencies to aid nutrition experts and researchers. This research attempts to develop such a system which may potentially be used by professionals. It also tries to include features for smart chefs in Indian kitchens.

II. LITERATURE SURVEY

Web-based services have revolutionized the way information is consumed. Dietary recommendation systems have attracted not just end consumers but also several food-related applications and services. Said systems aim to match the preferences of a user to a recipe. Several systems have been utilized ranging from content-based and collaborative filtering to evolved and sophisticated methods.[2] The proposed Graph Convolution Network (GCN) outperforms the state-of-the-art baselines. Further in- depth analyses reveal that GCN could alleviate the sparsity issue in food recommendation but the standard nutritional values are not considered.

Food preferences have manifested themselves as social, cultural, and economic force. Historically, recorded data for food preparation, preferences, and consumption patterns collected from households and individuals have played a huge role in realizing systems to recommend recipes and diets to users. The publication in consideration aims to scrape useful data from the world wide web and analyse its usefulness in food recommendations and patterns.[3] Recipe preference distributions exhibit more regional differences than ingredient preference distributions. The authors of [3] present a comprehensive multi- dimensional approach which allows to dig into the nature and evolution of users' online food preferences.

People's lives have grown increasingly focused on living a healthy lifestyle. The latter necessitates maintaining a healthy diet while taking into account the types and amounts of foods consumed. It also calls for leading an active lifestyle that includes adequate physical activity in order to control calorie and nutrient intake and consumption. As a result, people seek out nutrition specialists to conduct health assessments, which are costly, time-consuming, and difficult to come by.[5] The authors of [5] have built a recommender system using Fuzzy Reasoning. Results show that recommendations are on a par with and sometimes surpass those of human nutritionists. The main advantage is it performs meal planning with performing health state assessment or evaluation while a key drawback is a lack of personalisation to the user to make it applicable to the real world.

Food recommender systems are useful in guiding users in identifying the foods they want to eat. Deciding what to eat is a complicated and multi-faceted process that is influenced by a variety of elements including the ingredients, the appearance of the recipe, the user's personal food preferences, and multiple contexts such as previous meals.[6] The authors formulate the meal recommendation problem in their paper as predicting user preference for recipes based on three major aspects that influence a user's food choice. The first being the user's (and other users') history, second the contents of a dish, and finally the recipe's descriptive image. To tackle this difficult problem, the authors created Hierarchical Attention-based Food Recommendation (HAFR), a dedicated neural network-based solution capable of capturing the collaborative filtering effect, such as what similar users eat and inferring a user's preference for the ingredient level. The authors have built a recommender system using a Hierarchical Attention Network which outperforms several competing recommender solutions like Factorization Machine and Visual Bayesian Personalized Ranking with an average improvement of 12%. On the contrary, it does not incorporate user's health and nutritional requirements, no relations drawn between various ingredients, does not consider users' physical profile when recommending recipes.

A majority of the research done on the topic either focuses on a specific deficiency of the user and provides recipes based on that or fails to provide a custom personalized experience to the user based on their nutritional values and preferences. Our research aims to use standard nutritional values related to gender and age along with nutritional information including deficiencies of the user while also infusing the user’s personal preferences which makes the system unique to existing models. The nutritional profile analysis and recipe recommendation along with result ranking based on user’s needs is bundled into one functional system making it the novel feature of this research.

III. METHODOLOGY

The research aims to build a recommendation engine for users by considering multiple factors such as the user’s nutritional values, age, gender and personal preferences as inputs to give a personalised list of recipes in video and textual format with relevant nutritional information.

A. *Data Collection and Preparation*

The data used in this research was obtained from the National Institute of Nutrition (NIN). An Indian public health, nutrition and translational research centre[26] which falls under the Indian Council of Medical Research.

The data collected consists of standardized values of 5 major nutrients such as proteins, fats, carbohydrates including sugars and dietary fibres, vitamins and minerals which are then sub divided into 25 distinct nutrients. Another data set that contains the standard nutritional requirements for a person based on their gender and age range compiled by NIN is considered for computing the nutritional requirement for each user.

The data provided by NIN had different observations for the same ingredient or there existed a slight variation in the ingredient which was averaged out to maintain data consistency.

B. *System Design*

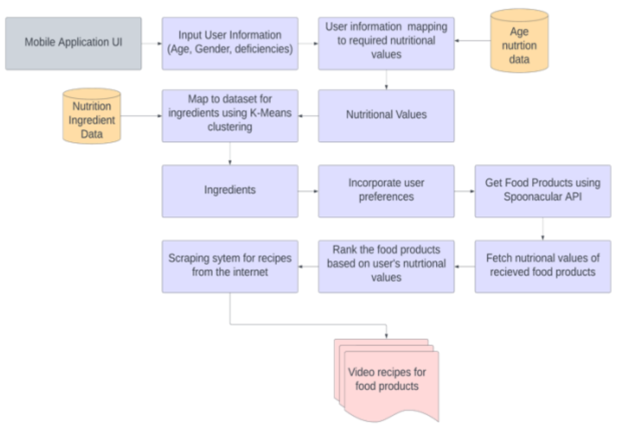
Mobile Application UI
Input User Information (Age, Gender, deficiencies)
User information mapping to required nutritional values
Age nutrtion data
Nutrition Ingredient Data
Map to dataset for ingredients using K-Means clustering
Nutritional Values
Ingredients
Incorporate user preferences
Get Food Products using Spoonacular API
Scraping sytem for recipes from the internet
Rank the food products based on user's nutrtional values
Fetch nutrional values of recieved food products
Video recipes for food products

Fig. 1. System Design

Fig. 3. represents the high-level system design used to develop a working scalable and deployable application as a product of this research. The general flow of the application is as follows, user is prompted to input their nutritional values, gender and age with the help of a simple file upload. The user’s data is then analysed by performing a difference calculation of the user’s values with the standard nutritional values from the NIN dataset, from this we obtain the deficiencies of the user and run a K-Means clustering algorithm to obtain ingredients with appropriate nutritional values which will help in bettering the user’s nutritional profile, with these ingredients food recipes are found using the Spoonacular API[27]. The recipes obtained are then ranked based on the user’s nutritional profile and then the system scrapes for recipes from the internet using the YouTube API.

C. *Algorithm Explanation*

Three primary components, namely deficiency detection, ingredient identification, and food product ranking form the base of the algorithm. Each component provides an intermediary output which feeds into the next component to finally list a set of ranked food product and their recipes for the users to consume.

Deficiency detection: The user’s consumption and biological profile are the primary input in this component. The input is a set of nutritional values that depict the general consumption pattern of the said user. It consists of 25 unique nutritional values i.e.., Protein, Ash, Fat, Dietary-Fibre, Carbohydrate, Energy, Thiamine, Riboflavin, Niacin, Pantac, Vitamin B6, Vitamin B7, Vitamin B9, Vitamin C, Aluminium, Calcium, Copper, Iron, Magnesium, Manganese, Nickel, Phosphor, Potassium, Sodium, and Zinc. The user’s age range and gender are also received in this stage. These values are compared against the recommended standard values for the age range and gender. The shortcomings of any nutrient are calculated by finding the arithmetic difference between the two sets (user's input and the standard values). The shortcoming depicts the nutritional deficiency/overload in the user's standard consumption scenario. This deficiency/overload set is then adjusted to derive the required nutrition by performic arithmetic addition with the recommended standard values for the given age range and gender. This final set represents the nutritional requirements of the user for a particular consumption by the user. This result acts as the input to the next component i.e.., ingredient identification.

Ingredient identification: The second major component in the solution, ingredient identification uses the nutritional requirements of the user as input. It refers to a dataset that contains 299 food ingredients with each ingredient depicted by a set of values with nutrient information similar to the user's input. The principal aim of this component is to identify a subset of the ingredients from the dataset which fulfil the user's nutritional requirements. This is achieved by using a clustering algorithm called K-Means which is an unsupervised method to quantize vectors by partitioning x observations into y clusters where each observation is in a cluster with minimal spatial distance to the points that behave like the mean of each cluster. Traditional K-Means produce clusters with uneven sizes but as per the solution design, the cluster centres are rearranged to guarantee a number of ingredients returned each time that is most similar to the user's nutritional requirement. This component outputs a set of ingredients that contain the necessary nutrients which can be used to identify whole food products that can be presented to the user with their recipe and basic nutritional information.

Food product ranking: The final component before the results are provided to the user is the food product ranking process. The input to this component is a set of food products with their nutritional information. The web is scraped for each of the nutrients that the user's nutritional information set contains. These food products are ranked by finding the Euclidian distance between each of the food products and the nutritional requirement of the user. Euclidian distance is calculated as the distance between the two sets when represented as a point mapped on a 25 dimension map. The food products are ranked in the order of the sets that are closest to the user's nutritional requirements in this spatial mapping. The web is then scraped for recipes for each of these food products and presented to the user.

The above-mentioned sequence of components forms the base of the solution which is built upon to verify the sanity of the solution designed.

D. Algorithm Implementation

Implementation to verify the sanity of the solution provided is carried out on Jupyter Notebook as python scripts. The following section contains a detailed description regarding the implementation of all functional components in the solution provided.

1. Deficiency detection and nutritional requirement analysis

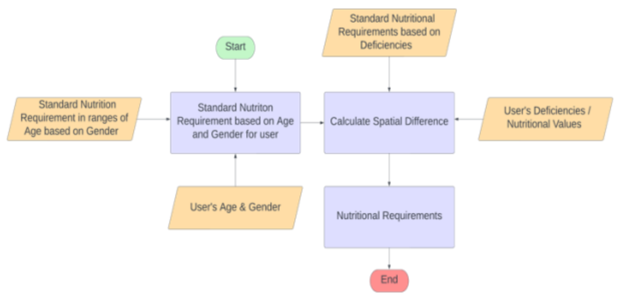
Start
Standard Nutritional Requirements based on Deficiencies
Standard Nutrition Requirement in ranges of Age based on Gender
Standard Nutriton Requirement based on Age and Gender for user
Calculate Spatial Difference
User's Deficiencies / Nutritional Values
User's Age & Gender
Nutritional Requirements
End

Fig. 2. Comparing standard nutrition values and user's profile to detect deficiencies

This component receives defined inputs from the user regarding their age range (one of 11-14, 15-18, 19-24, 25-50, 51+), their gender (either male or female), and a nutrition consumption profile with 25 well-defined nutrients (Protein, Ash, Fat, Dietary-Fibre, Carbohydrate, Energy, Thiamine, Riboflavin, Niacin, Pantac, Vitamin B6, Vitamin B7, Vitamin B9, Vitamin C, Aluminium, Calcium, Copper, Iron, Magnesium, Manganese, Nickel, Phosphor, Potassium, Sodium, and Zinc). It also refers to the recommended standard for each of the nutrients across age range and gender. The user's nutritional profile is arithmetically subtracted from the defined set of recommended standard nutrient values to identify nutrient overload/deficiency in the user's nutrient consumption. This derived set is then added to the standard set to identify the nutritional requirement for the user's single consumption. This critical component sets the base for ingredient identification.

Deficiency Detection procedure:

```
1:  procedure nutrition-requirements (user-nutrition-
        values, age, gender)
2:      ms ← male-standard-nutritional-values[age]
3:      fs ← female-standard-nutritional-values[age]
4:      if gender = male then
5:          difference ← ms - user-nutrition-values
6:          nutritional-requirements  ms + (-
            difference)
7:      else
8:          difference ← fs - user-nutrition-values
9:          nutritional-requirements  fs + (-
            difference)

10:     return nutritional-requirements
11: end
```

2. Ingredient identification

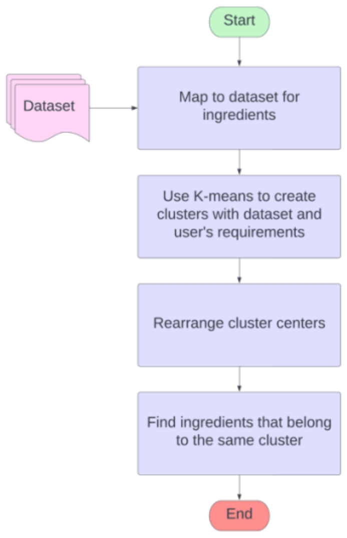
Start
Dataset
Map to dataset for ingredients
Use K-means to create clusters with dataset and user's requirements
Rearrange cluster centers
Find ingredients that belong to the same cluster
End

Fig. 3. Detecting ingredients required for the user

This component identifies the set of ingredients that fulfil the user's nutritional requirements by finding ingredients most similar to the user's requirements. This task is achieved by running a K-Means clustering algorithm on 299 unique ingredients with the same ser of nutrients the user's input contains to guarantee that each of the nutrients has an impact on the final result. K-Means is a widely known unsupervised clustering algorithm that partitions a set of n observations into y clusters where y n. It does so by randomly picking y cluster centres. Assign each of the remaining n - y observations into one of the y clusters. Find the distance between the observation and each of the y cluster centres. Reassign the observation to a cluster in such a way that the distance is minimal. This iteration continues until each of the observations is assigned to a cluster with minimum distance between the observation and the cluster centre. To ensure that a set number of ingredients are always returned irrespective of the input, the clusters derived from K-Means are recentred to pick the 9 most similar ingredients compared to the user's nutritional requirements. This component returns a set of ingredients to satisfy the user's nutritional requirement.

Ingredient detection procedure:

```
1:  procedure identify-ingredients (nutritional-requirements)
2:      import ingredient-list
3:      ingredient-list ← ingredient-list + nutritional-requirements
4:      n-clusters ← ingredient-list-size/cluster-size
5:      k-means-object ← K-Means(n-clusters)
6:      rearranged-cluster ← Rearrange-Cluster(k-means-object)
7:      required-cluster ← Cluster-Number(rearranged-cluster, nutritional-requirements)
8:      ingredients ← []
9:      while (ingredient in ingredient-list) do
10:         ingredient-cluster ← Cluster-Number(rearranged-cluster, ingredient)
11:         if ingredient-cluster = required-cluster then
12:         ingredients ← ingredients + ingredient
13:     done
14:     return ingredients
15: end
```

3. Get food products

A vital component in the solution design flow, this module ensures to get food products. At this stage, the user could optionally reject some of the suggested ingredients due to various reasons such as allergic reactions, unavailability of certain ingredients, or just personal choice. Once the user's favourable list of ingredients is received, an upstream call is made to Spoonacular API. It returns a set of food products along with their nutritional information, the and ingredients used. The ingredients used in the final food product are generally a subset of the ingredients sent as a query as the API returns meaningful and practical food products back.

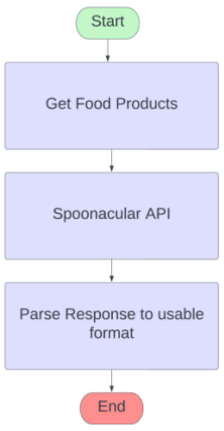

Fig. 4. Obtaining food products with the specific set of ingredients

Get Food Products procedure:

```
1:  procedure get-food-products(ingredients)
2:      user-modified-ingredients ← PersonalizeIngredients(ingredients)
3:      food-products ← CallSpoonacular(user-modified-ingredients)
```

```
4:      parsed-response ← ParseAPIResponse(food-products)
5:      return parsed-response
6: end
```

4. Rank food products

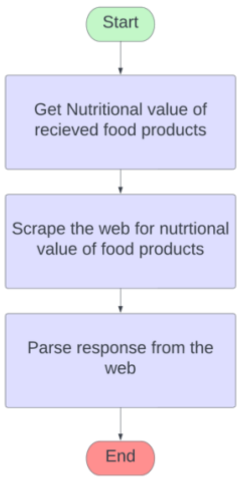
Start
Get Nutritional value of recieved food products
Scrape the web for nutrtional value of food products
Parse response from the web
End

Fig. 5. Scraping nutritional value of cooked products

The final component that concludes the implementation of the base algorithm. It receives parsed API responses from the previous stage and the user's nutritional requirement set as input. It uses Euclidean distance as the method to rank the food products in comparison with the user's nutritional requirements. The formula for Euclidean distance is as follows.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

where,

p, q are points in an n-dimensional plane

n is the number of dimensions (25 in our case)

pi and qi are co-ordinates of p and q in dimension i.

The user's nutritional requirement set is taken as vector q and the nutritional information is considered as vector p. The Euclidean distance is calculated for each of the food products by calculating the arithmetic difference between the two vectors. The food vectors are then ranked in the order of the least difference.

Rank Food Products procedure:

```
1: procedure rank-food-procedure(food-products,
                nutritional-requirements)
2:      distance ← []
3:      while (food-product in food-products) do
4:           dist ← EuclideanDistance(food-product,
                 nutritional-requirements)
5:           distance ← distance + dist
6:      done
7:      ranked-food-products ←
            RankFoodProducts(distance, food-products)
8:      return ranked-food-products
9: end
```

5. Get recipes for ranked food products

The last functional component in the solution flow is to get recipes for each of the ranked food products back to the user to enhance user experience. This module receives the ranked food products as input. An upstream request is made to YouTube's consumer API. YouTube's Consumer APIs are freely available for an application to consume at will. The recipe for each of the food products is queried in this process and the most relevant video from the list the API returns is picked. A final set is created with recipes and nutritional information for each of the food products in order of their rankings and returned to the user.

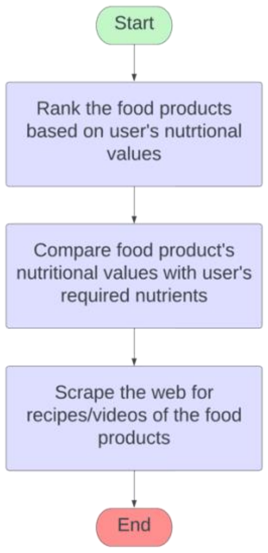
Start
Rank the food products based on user's nutrtional values
Compare food product's nutritional values with user's required nutrients
Scrape the web for recipes/videos of the food products
End

Fig. 6. Obtaining best recommendations for user

Get Recipes procedure:

```
1: procedure get-recipe-procedure(ranked-food-
             products)
2:      recipeURLs ← []
3:      while (food-product in ranked-food-product) do
4:           videoURL ← YoutubeAPI(food-
             product, sort = relevance)
5:           recipeURLs ← recipeURLs +
                       videoURL
6:      done
```

```
7:      return recipeURLs
8:  end
```

E. *Flutter Application*

A flutter application has been developed to showcase the functionality of the algorithm designed.

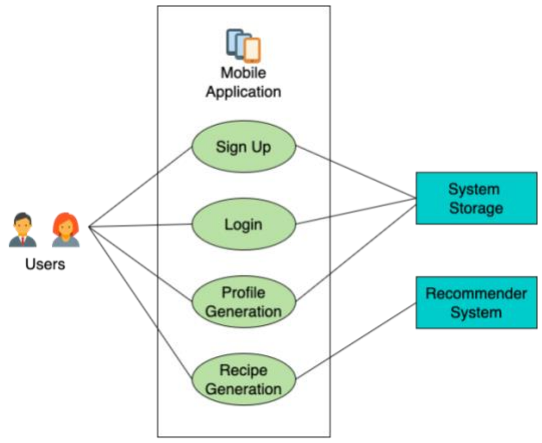

Fig. 7. Flutter application architecture

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, macOS, Windows, and the web from a single codebase.

The application developed follows a client-server architecture, wherein the recommender system is made accessible by a Flask backend which acts as a standalone REST API server. The frontend is developed with the help of Flutter and performs REST API calls to the backend with the relevant inputs obtained from the user.

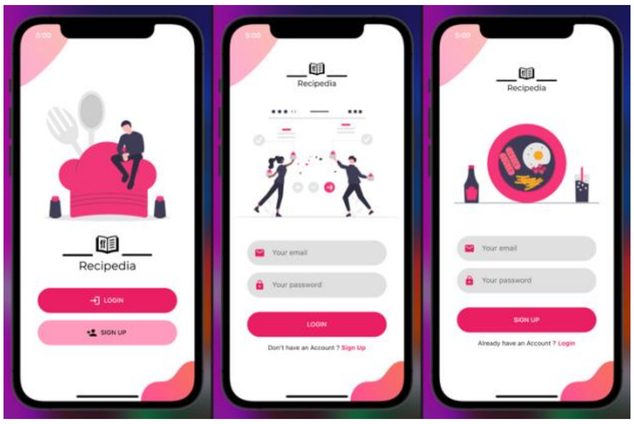

Fig. 8. Secure user authentication

The user must first create an account on the application using a unique email ID and password combination. The user is securely authenticated using Firebase Authentication service which follows industry wide security standards. Once authenticated successfully the user is directed to their personalized home screen shown in Fig. 9.

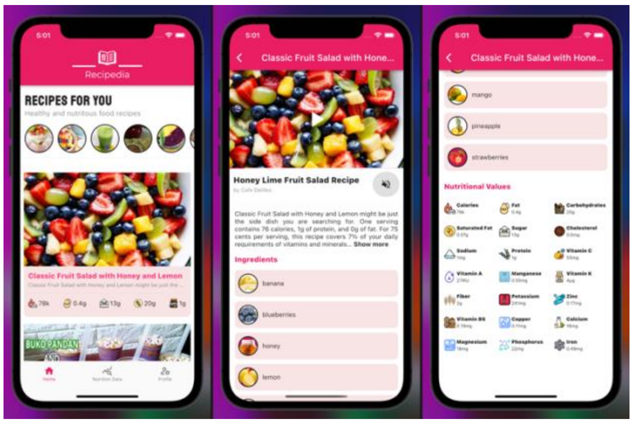

Fig. 9. Personalized recipes for the user

Fig. 9. showcases the different screens a user views after a successful authentication. The home screen consists of a list of recipes ranked in order of the user's preferences and nutrition profile. The user can click on any given recipe to get more information regarding it. The recipe focused screen contains the recipe in a video format obtained from YouTube along with textual information regarding the ingredients used and the nutritional values of the selected recipe. The user can always view their personal nutritional profile and update their data frequently as and when required with a simple file upload operation. The recipe recommendation engine automatically suggests new recipes based on the change in the user's nutritional profile and preferences.

IV. RESULTS

The final recipe video recommendation is made based on multiple stages of intermediate results. Initially, two main datasets are used. All the intermediate results and computations are made based on these datasets. One consists of the standard nutritional values of a given age and gender. The other dataset consists of 300 ingredients along with the nutritional component values of each ingredient. On receiving the user inputs through the UI, it is compared with the standard values of the user.

Assume a user of a certain age and gender has a standard vector:

$S_x = [s_1,s_2,s_3,.....,s_n]$, where $s_1,s_2,s_3,.....,s_n$ are all standard nutritional component values such as carbohydrates, proteins, saturated fats and so on.

User input is fed in the form of the user vector:
$U_x = [u_1,u_2,u_3,.....,u_n]$, where $u_1, u_2, u_3,....., u_n$ are the user's nutritional component values such as carbohydrates, proteins, saturated fats and so on.

A deficiency vector Dx is created by subtracting Ux from Sx, and its absolute value is considered.

$$D_x = \text{Abs}(\, S_x - U_x\,)$$

This deficiency vector Dx is further used as an input to the clustering algorithm along with all the ingredients.

The second aforementioned dataset comes into play for the clustering algorithm. Assume each ingredient vector to be I_x=[I_1, I_2, I_3,....,I_{300}] each of which is a vector with values [i_1,i_2,i_3,.....,i_n], where i_1, i_2, i_3,....., i_n are the ingredient's nutritional component values such as carbohydrates, proteins, saturated fats and so on.

Dx is then appended to Ix to create the final dataset for clustering (F_x).

$$F_x = D_x + I_x$$

The Euclidean norm is computed in order to create a data point for each vector fi in F_x. On computing the clusters which comprise ingredients similar to D_x, we essentially identify and obtain the ingredients that could potentially cover up the deficiencies.

We further move to choose all the ingredients in the cluster that D_x exists. These ingredients are then fed into the Spoonacular API which proceeds to suggest various recipes. Upon obtaining multiple recipe recommendations, we further personalize the recommendations by choosing the top choices among the API recommendations by comparing the Euclidean norm of each recommendation with the user.

Thus, the recommendations as well as the ingredients each pass through a layer of personalization which ultimately aids in obtaining robust and accurate results.

Given below are some visualizations that can better help understand the nuances of the developed recommendation engine.

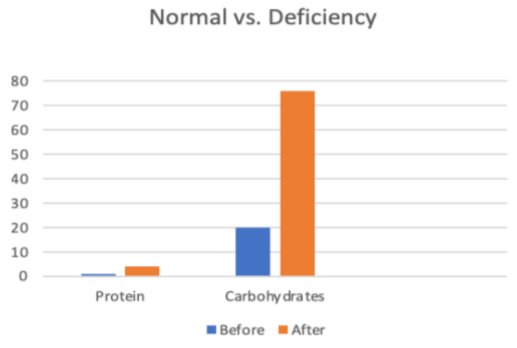

Fig. 10. shows the levels of protein that the engine recommends for a user without a deficiency vs. a user with a deficiency in protein and carbohydrates. (in mg)

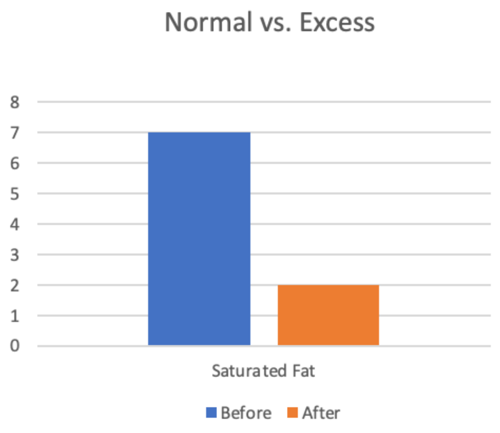

Fig. 11. shows the levels of saturated fat the engine recommends for a user that consumes a healthy diet vs. a user that consumes a fatty diet. (in mg)

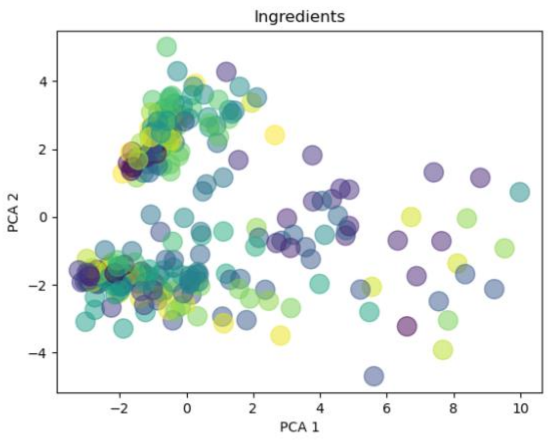

Fig. 12. Results from K-Means Clustering

Fig. 10. is the visualization that comprises 30 clusters with 10 data points(ingredients) in each cluster. There is no visible hyperplane in this 2D visualization that segregates the clusters because there exists numerous features (nutritional components) based on which its being clustered. We choose the cluster that contains the vector we appended (deficiency vector) since it'll have various ingredients that can cover the deficits and in turn be used to make some food product.

V. CONCLUSION

In conclusion, the result of this research is an algorithm which ingests a user's age, gender and nutritional profile to obtain a list of suitable ingredients for the user. The user's personal preferences are also considered for recipe recommendation and depending on the user's preferences certain ingredients are removed. The final list of ingredients are then fed into the Spoonacular API that gives various choices of final food products that can be made using the ingredients obtained from K-Means clustering. Another layer of personalization is added by ranking the choices suggested by the API on considering the user's nutritional profile. The YouTube API is used to scrape the web to find videos of recipes of the final recommended food products. These videos are then displayed to the user along with other relevant information through the flutter

application with an elegant UI and user-friendly design that encompasses a smart and personalized recipe recommendation engine.
.