SIES (NERUL) COLLEGE OF ARTS, SCIENCE AND COMMERCE

NAAC ACCREDITED 'A' GRADE COLLEGE (ISO 9001:2008 CERTIFIED INSTITUTION) NERUL, NAVI MUMBAI – 400706

PROJECT REPORT ON

DIET RECOMMENDED SYSTEM

SUBMITTED BY

Rohan Thakur

PROJECT GUIDE

Asst. Professor Manasvi Sharma

SUBMITTED IN PARTIAL FULFILLMENT THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

MSc. (COMPUTER SCIENCE)

SEMESTER – IV, 2021 - 2022

SIES (NERUL) COLLEGE OF ARTS SCIENCE AND COMMERECE

NAAC ACCERDITED 'A' GRADE COLLEGE

(ISO 9001:2015 CERTIFIED INSTITUTION)

NERUL, NAVI MUMBAI - 400706

THIS IS TO CERTIFY THAT THE PROJECT TITLED

DIET RECOMMANDED SYSTEM

IS UNDERTAKEN BY

ROHAN KIRAN THAKUR

Seat No: 20

In partial fulfilment of MSc - IT / CS Degree (Semester _ **IV**)Examination in the academic year **2021-2022** and has not been submitted for any other examination and does not form part of any other course undergone by the candidate. It is further certified that he/she has completed all the required phases of the project.

Project Guide External Examiner

Head of Department Principal

ACKNOWLEDGMENT

I extend my heartfelt gratitude and thanks to Asst. Professor Mansvi Sharma sir for providing me excellent guidance to work on this project and for their understanding and assistance by providing all the necessary information needed for my project topic.

I would also like to acknowledge all the staffs for providing a helping hand to us in times of queries & problems. The project is a result of the efforts of all the peoples who are associated with the project directly or indirectly, who helped us to work to complete the project within the specified time frame. They motivated me in the project and gave a feedback on it to improve my adroitness.
Thanks to all my teachers, who were a part of the project in numerous ways and for the help and inspiration they extended to me and for providing the needed motivation.
With all Respects & Gratitude, I would like to thanks to all the people, who have helped for the development of the Project.

ROHAN THAKUR

MSc. Computer Science (Part-II)

SIES (Nerul) College of Arts, Science, and Commerce.

Table of Content

PROJECT REPORT ON:

Diet Recommended System

INTRODUCTION:

Recommender systems (RS) suggest items of interest to users of information systems or e-business systems and have evolved in recent decades. A typical and well-known example is Amazon's suggest service for products. We believe the idea behind recommender systems can be adapted to cope with the special requirements of the health domain. Recommendations based on single foods or food groups are easier to implement when only a few foods serve as the major sources of an essential dietary component (e.g., dairy products, which are the primary source of calcium in the Indian diet).

During the last decades huge amounts of data have been collected in clinical databases representing patients' health states (e.g., as laboratory results, treatment plans, medical reports, diet plans). Hence, digital information available for patient-oriented decision making has increased drastically but is often scattered across different sites. As a solution, personal diet recommender systems (DRS) are meant to centralize an individual's health data and to allow access for the owner as well as for authorized health professionals

Nutrients are Essential compounds that the body can't make or can't make insufficient quantity. According to the World Health Organization (WHO), these nutrients must come from food, and they're vital for disease prevention, growth, and good health. Macronutrients are eaten in large amounts and include the primary building blocks of your diet protein, carbohydrates, and fat which provide your body with energy. Vitamins and minerals are micronutrients, and small doses go a long way. Most of disease occurred due to efficiency of nutrients. To fill these nutrients, we can suggest natural diet (that have no side effects), and precautions to user.

IMPLEMENTATION DETAIL

Data Collection:

For This Project I required two Datasets. The primary data would be collected from data.world that contain foods details with Nutrition's values. The second Dataset I got from Kaggle that contain Disease details with Inefficient Nutrition's and Precautions.

System Requirement Specification:

- Anaconda: - To work on this project I am using Anaconda, as it is a distribution of the Python programming languages for scientific computing i.e data science, machine learning applications, large-scale data processing, predictive analytics, etc., that aims to simplify package management and deployment.
- Google Colab will be used as online python environment.
- Python: - Python is an interpreted, high-level, general-purpose programming language. Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python has been built with extraordinary Python libraries that are used in Big Data for solving problems that are as follow:
    1. NumPy
    2. Seaborn
    3. Pandas
    4. Matplotlib

Algorithms:

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group and dissimilar to the data points in other groups. It is basically a collection of objects on the basis of similarity and dissimilarity between them.

- K-Means Clustering: - k-means Clustering is an unsupervised learning algorithm that is used to solve the clustering problems in machine learning or data science. In this topic, we will learn what is K-means clustering algorithm, how the algorithm works, along with the Python implementation of k-means clustering.
- My main reason behind using k-means because it allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

Classification is a technique where we categorize data into a given number of classes. The main goal of a classification problem is to identify the category/class to which a new data will fall under. Here I am using the classification algorithms as my data set consist of multiple binaries and some categorical values. The classification algorithms that best suits my Prediction based on factors are follows:

- Random Forest: - The random forest algorithm is also known as the random forest classifier. The RF algorithm comprises a random collection or a random selection of a forest tree. It creates a random sample of multiple decision trees and merges them together to obtain a more stable and accurate prediction through cross validation.
- My main reason behind using Random Forest because it works well with a mixture of numerical and categorical features and if we generate with more trees based on features then it won't allow over-fitting trees in the model.

EXPERIMENTAL SET UP AND RESULTS DATASET

For my project I required two datasets

```
[6]  disease_nutrition.info()
     # disease_nutrition.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 91 entries, 0 to 90
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype 
---  ------                 --------------  ----- 
 0   disease_id             91 non-null     int64 
 1   disease                91 non-null     object
 2   ineficient_nutritions  91 non-null     object
 3   Precaution_1           91 non-null     object
 4   Precaution_2           83 non-null     object
 5   Precaution_3           64 non-null     object
 6   Precaution_4           40 non-null     object
dtypes: int64(1), object(6)
memory usage: 5.1+ KB
```

The disease dataset consist of 91 rows and 7 columns.

```
food_nutrition.shape
# food_nutrition.head()
```

```
(6332, 37)
```

The food dataset consist of 6332 rows and 37 columns.

Data Cleaning:

The dataset contain ‘null’ value replace with 0.

```
[73] food_nutrition.fillna(value = 0, inplace = True)
```

Data Preprocessing:

The dataset contain ‘beaf’ ,’Stek’,’pork’ value remove that row.

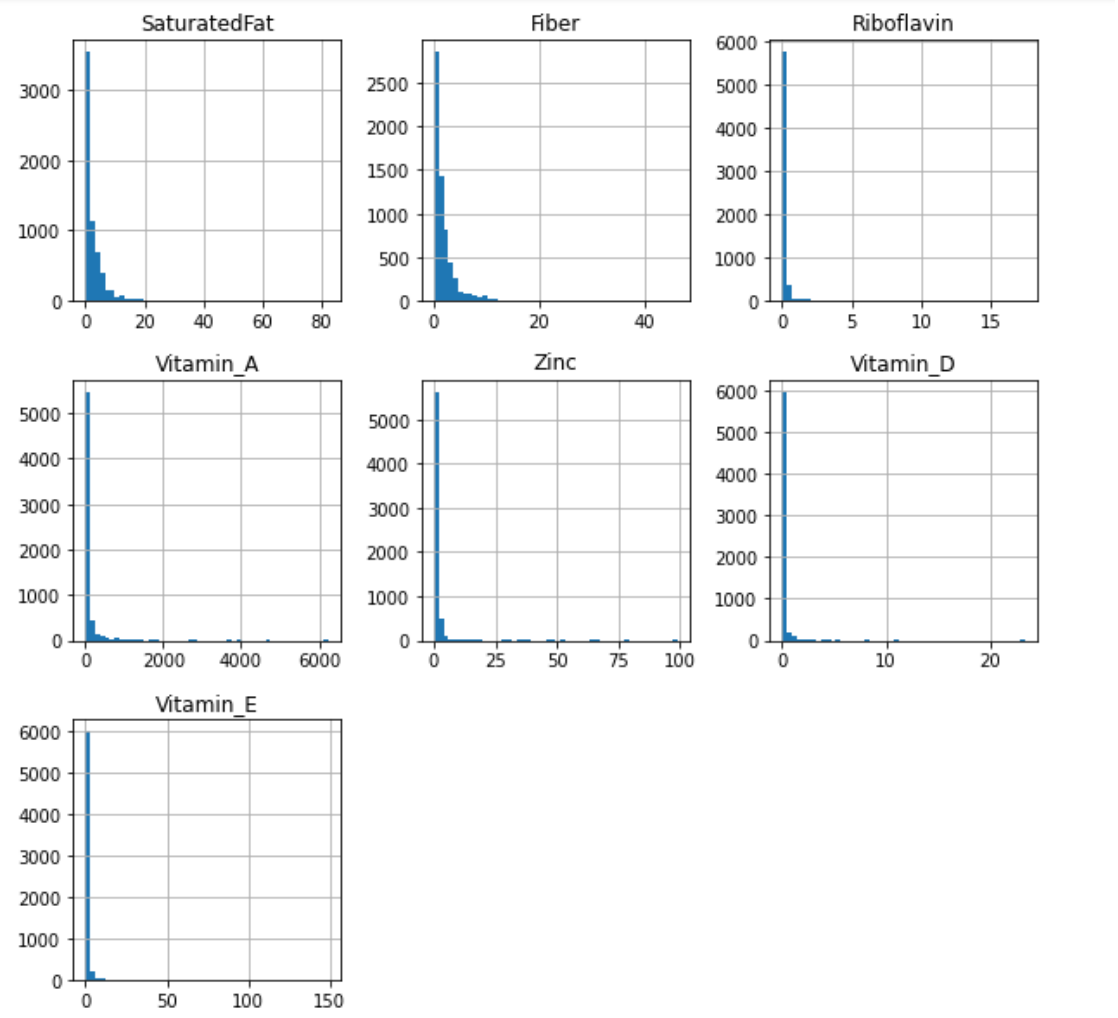
```
[74] food_nutrition = food_nutrition[food_nutrition["Description"].str.contains("beaf") == False]
```

```
[ ]  df_int = food_nutrition.select_dtypes(include=['int'])
     # dataframe with integer format Data Types to Float
     df_int
```

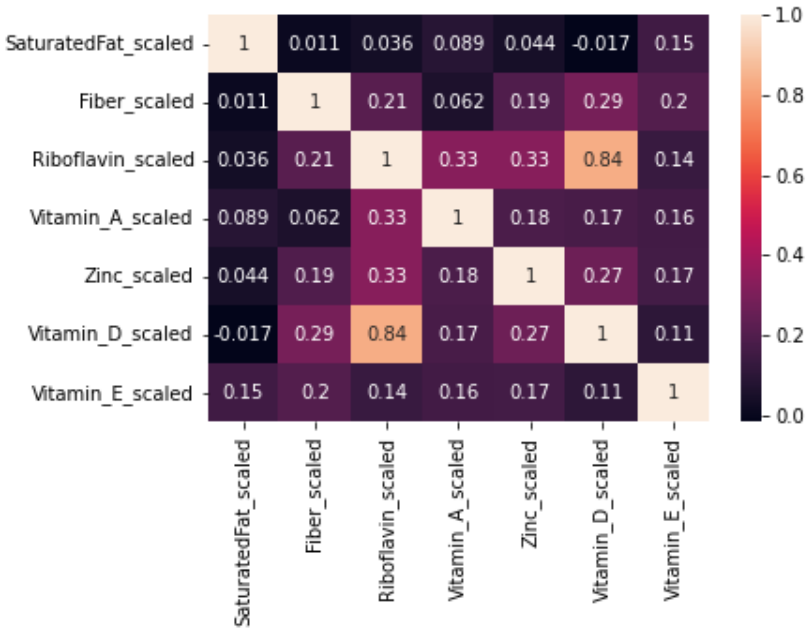
```
[ ]  int_columns=['Carotene','Cryptoxanthin','Cholesterol','Lutein and Zeaxanthin',
                  'Lycopene','Retinol','Calcium','Magnesium','Phosphorus','Potassium','Sodium','Vitamin_A']
     # Replace Data Types to Float
     food_nutrition[int_columns] = food_nutrition[int_columns].astype('float')
```

Dataset with integer format Data Types to Float

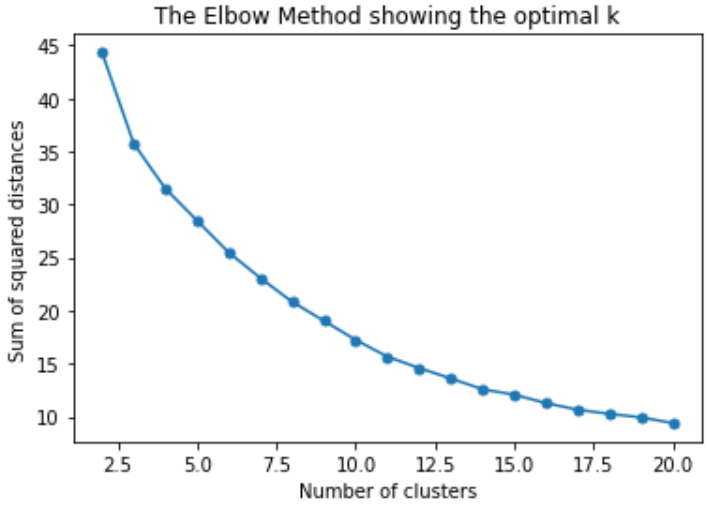
Display bar charts with insufficient nutritions food Quantity in dataset with the



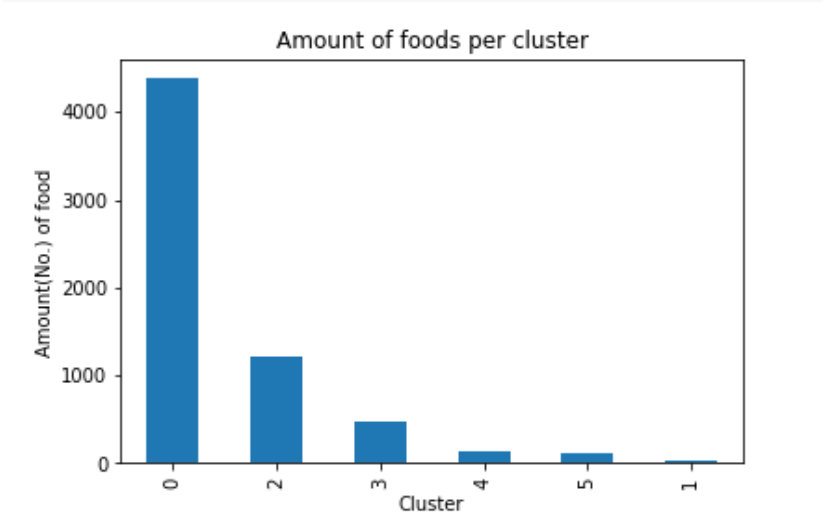
How food can co-related to each other using heat map



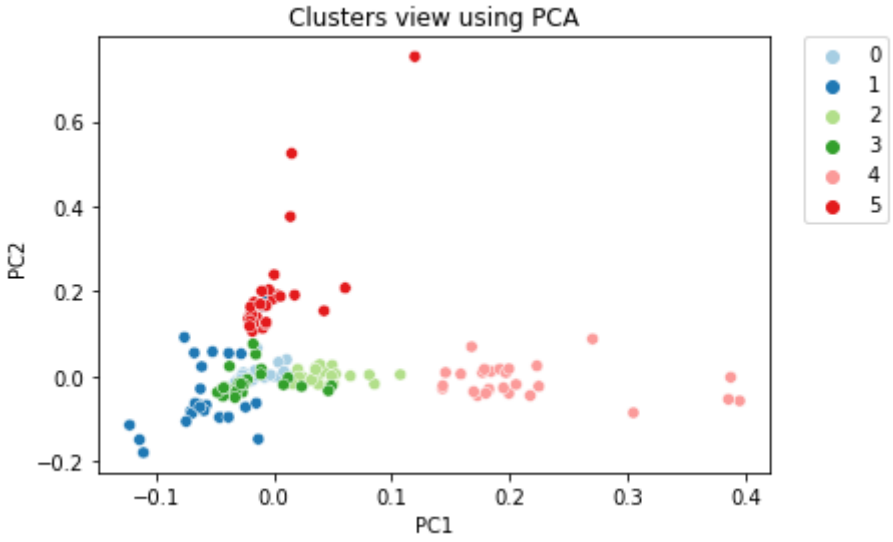
The Elbow Method for Finding optimal k Number of clusters with respect to sum of squared distances. There are 20 cluster to take find optimal clusters, here we got k=6



Visualizing the clusters using bar plot total food with respect to total clusters you choose.



PCA For visualization of datapoints in two dimensions.



Here I convert Unsupervised to Supervised, I applied k-means for labeling then extra column i.e ‘cluster’ was added in data. after applying k-means I am going to apply random forest to find good food.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
X=df_food_scaled[columns_to_cluster_scaled]   # Features
y=df_food_scaled['cluster']

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

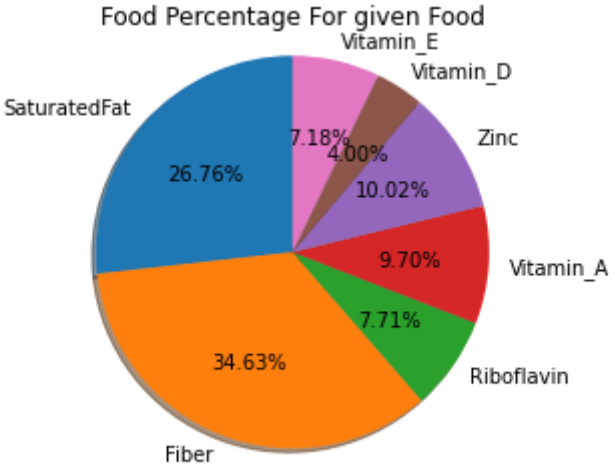
```
clf=RandomForestClassifier(n_estimators=100)
clf.fit(X_train,y_train)
#Train the model using the training sets
y_pred=clf.predict(X_test)
y_pred
```

```
array([2, 0, 0, ..., 5, 0, 0], dtype=int32)
```

```
clf.estimators_[0]
sample_tree = clf.estimators_[20]
```

```
DecisionTreeClassifier(max_features='auto', random_state=2121836698)
```

After applying both algorithm I get the data frame of 20 foods with their nutrition's amount quantity. with the help of pie chart display following:



ANALYSIS OF RESULT

Random Forest:

- Classification Report: -
  Here the classification report for test data and predictions

```
# View the classification report for test data and predictions
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1333
           1       1.00      0.50      0.67         2
           2       0.97      0.97      0.97       352
           3       0.96      0.99      0.97       139
           4       1.00      1.00      1.00        43
           5       0.84      0.87      0.86        31

    accuracy                           0.98      1900
   macro avg       0.96      0.89      0.91      1900
weighted avg       0.98      0.98      0.98      1900
```

Here we have 0.99 precision of 0's with 0.99 of recall it means we have 99% of 0 actual data out of that algorithm is able to predict 99% of data. Similarly we have 0.84 precision of 5's with 0.87 of recall it means we have 85% of actual data out of that algorithm is able to predict 84% of data.

- Accuracy: -

```
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.988421052631579
```

- Confusion Matrix: -

```
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[1298,    0,    2,    0,    0,    0],
       [   0,    6,    0,    1,    0,    1],
       [   5,    0,  368,    0,    0,    0],
       [   0,    0,    0,  149,    0,    1],
       [   0,    0,    4,    1,   31,    0],
       [   3,    1,    2,    1,    0,   26]])
```

CONCLUSION:

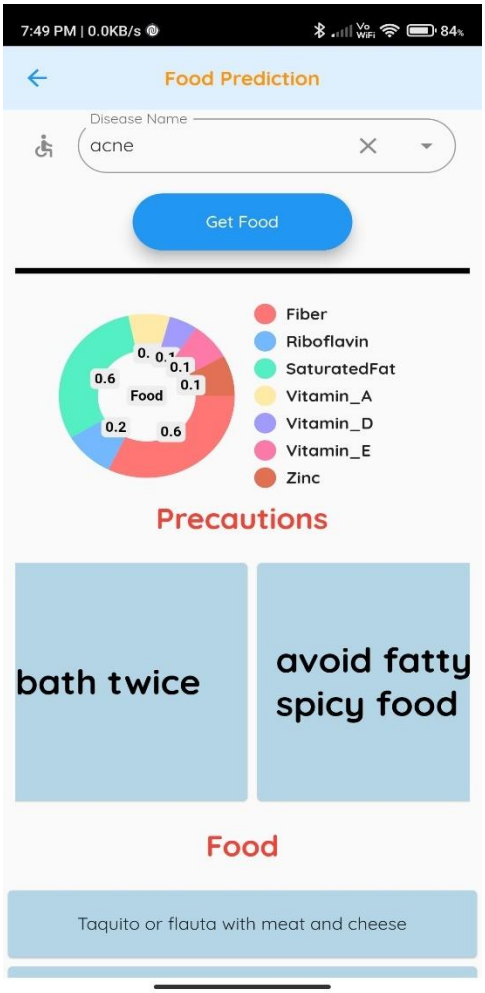
- This project satisfying need will help to put patients in control of their own health data and therefore increase patients' autonomy. An approach of integrating recommender systems into personal Diet recommender system (DRS) was outlined.
- we can suggest natural diet (that have no side effects), precautions to user. The proposed system builds a user's health profile and, accordingly, provides individualized nutritional recommendations, also with attention to food geographical origin.
- The importance of nutritional guidance is increasing day by day to lead a healthy and fit life and by accepting the user's preferences and a user's profile in the system a healthy diet plan is generated.

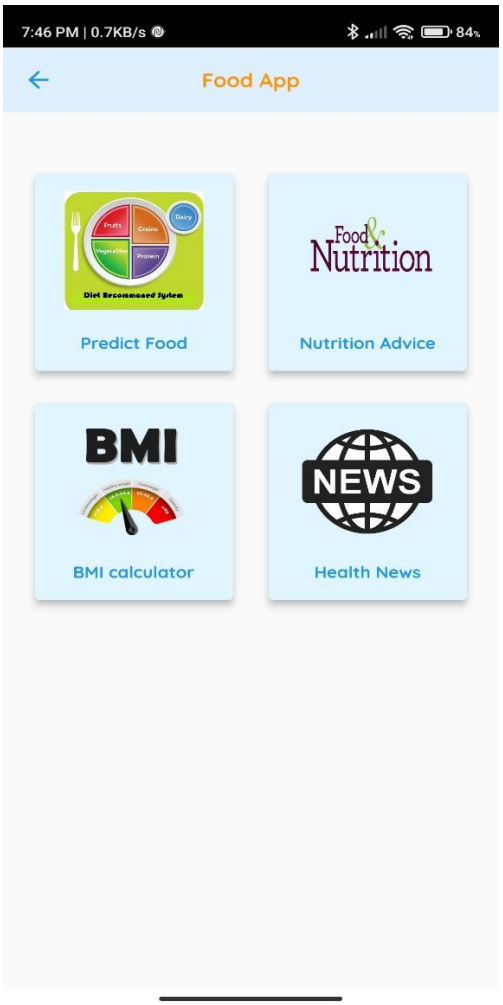
FUTURE ENHANCEMENT:

- More Functionality can be added depending upon the user requirements and Specifications.
- I trying to increase Code performance by using maximum use of core.
- In This project I used k-means (unsupervised algorithm) and random forest (supervised algorithm) for more performance I will try other algorithm
- Using NLP(Natural Language Processing), I can add veg and Non-Veg Difference

USE CASE:

Created a Mobile Application which uses the model successfully and shows Precautions, Food and Pie chart that display quantity of nutrition's of food.





PROGRAM CODE:

```
"""DRS.ipynb

Original file is located at
    https://colab.research.google.com/drive/18_ciqcHustYcpM5gdeOVxLPITycpbmI9

**Project Start**
"""

from google.colab import drive
drive.mount('/content/drive')

import pandas as pd

####### dataset #######
food_nutrition = pd.read_csv("/content/drive/MyDrive/dataset/food_nutrition.csv")
disease_nutrition =
pd.read_csv("/content/drive/MyDrive/dataset/disease_nutrition.csv",encoding='unicode_
escape')

food_nutrition.shape
# food_nutrition.head()

disease_nutrition.info()
# disease_nutrition.head()

food_nutrition.fillna(value = 0, inplace = True)

food_nutrition = food_nutrition[food_nutrition["Description"].str.contains("beaf") == False]

"""**Part-1**"""

####### Methods #######
def get_disease(disease_name):
        if(disease_name not in list(disease_nutrition["disease"])):
                return False
        else:
                return
disease_nutrition[disease_nutrition.disease==disease_name]["disease"].values[0]

def get_disease_id(disease):
        return disease_nutrition[disease_nutrition.disease ==
disease]["disease_id"].values[0]

def get_disease_ie(disease):
        return disease_nutrition[disease_nutrition.disease ==
disease]["ineficient_nutritions"].values[0]

####### get-set Data #######
```

```
users_disease = input("Enter Disease Name: ")
disease_name = get_disease(users_disease)
if(disease_name==False):
  print("Disease Not Found")
else:
  disease_id = get_disease_id(disease_name)
  i= disease_id-101
  fd= disease_nutrition.iloc[i]
  pre_list = [fd["Precaution_1"],fd["Precaution_2"],fd["Precaution_3"],fd["Precaution_4"]]
  print("Precauctions: ",pre_list)
  disease_ie = get_disease_ie(disease_name)
  dis_list = list(disease_ie.split(" "))
  for ele in dis_list:
    if(ele==""):
      dis_list.remove(ele)
  print("Ineficient Nnutritions: ",dis_list)

"""**Part-2**"""

import seaborn as sns
import matplotlib.pyplot as plt

food_nutrition[dis_list].hist(bins=50, figsize=(10,10))
plt.show()

"""**Preprocessing data**"""

from sklearn.preprocessing import MinMaxScaler

columns_to_cluster = dis_list

#MinMaxScaler
#Transform features by scaling each feature to a given range.Here is an example to
scale a data matrix to the [0, 1] range:
mms = MinMaxScaler()
food_scaled = mms.fit_transform(food_nutrition[columns_to_cluster])
print("Scaled Food Value: ", food_scaled[0,:])

columns_to_cluster_scaled = []
for i in dis_list:
  columns_to_cluster_scaled.append(i+"_scaled")

df_food_scaled = pd.DataFrame(food_scaled, columns=columns_to_cluster_scaled)

ax = sns.heatmap(df_food_scaled.corr(), annot=True)
plt.show()

"""**Training the model**"""

from sklearn.cluster import KMeans
```

```
from sklearn.metrics import silhouette_score

n_clusters = range(2,21)
ssd = []
sc = []
dict={}
for n in n_clusters:
    km = KMeans(n_clusters=n, max_iter=300, n_init=10, init='k-means++',
random_state=42)
    km.fit(food_scaled)
    preds = km.predict(food_scaled)
    centers = km.cluster_centers_
    ssd.append(km.inertia_)
    score = silhouette_score(food_scaled, preds, metric='euclidean')
    sc.append(score)                                    #calculate the goodness of a
clustering
    print("Number of Clusters = {}, Silhouette Score = {}".format(n, score))
    dict[n] = score

plt.plot(n_clusters, ssd, marker='.', markersize=10,animated=True)
plt.xlabel('Number of clusters')
plt.ylabel('Sum of squared distances')
plt.title('The Elbow Method showing the optimal k')
plt.show()


k=6
model = KMeans(n_clusters=k, random_state=42).fit(food_scaled)
pred = model.predict(food_scaled)
print('10 first clusters: ', model.labels_[:10])

"""**Visualizing the clusters**"""

df_food_scaled['cluster'] = model.labels_
df_food_scaled['cluster'].value_counts().plot(kind='bar')
plt.xlabel('Cluster')
plt.ylabel('Amount(No.) of food')
plt.title('Amount of foods per cluster')
plt.show()

display(df_food_scaled['cluster'].value_counts())
minor_cluster = df_food_scaled['cluster'].value_counts().tail(1)
print("Amount of food in the smallest cluster: ", int(minor_cluster.values))

df_food_scaled.info()

food_nutrition.info()

df_food_joined = pd.concat([food_nutrition,df_food_scaled], axis=1).set_index('cluster')
```

```
for cluster in range(k):
    display(df_food_joined.loc[cluster, ['Description']].sample(frac=1).head(10))

df_food_joined.head(5)
df_food_joined.info()

"""**Applying PCA to visualize the clusters**"""

from sklearn.decomposition import PCA

pca = PCA(n_components=len(dis_list), random_state=42)
food_pca = pca.fit_transform(food_scaled)
pca.explained_variance_ratio_.sum()

column_list=[]
for i in range(len(dis_list)):
  column_list.append("PC"+str(i))
print(column_list)

df_pca = pd.DataFrame(food_pca, columns=column_list)
df_pca['cluster'] = model.labels_
# df_pca.head()
df_pca.info()

sampled_clusters_pca = pd.DataFrame()

for c in df_pca.cluster.unique():
    df_cluster_sampled_pca = df_pca[df_pca.cluster == c].sample(n=int(minor_cluster),
random_state=42)
    sampled_clusters_pca = pd.concat([sampled_clusters_pca,df_cluster_sampled_pca],
axis=0)
sampled_clusters_pca.cluster.value_counts()

sns.scatterplot(x='PC1', y='PC2', hue='cluster', data=sampled_clusters_pca,
legend="full", palette='Paired')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.title('Clusters view using PCA')
plt.show()

df_user_food_joined = pd.concat([food_nutrition,df_food_scaled],
axis=1).set_index('cluster')
for cluster in df_food_scaled['cluster'].unique():
    display(df_user_food_joined.loc[cluster, ['Description']].sample(frac=1).head(10))

# df_user_food_joined.info()
df_user_food_joined.head()

"""**Recommending Food**"""

df_user_food_joined.reset_index(inplace=True)
```

```
cluster_pct = df_user_food_joined.cluster.value_counts(normalize=True)*20

if int(cluster_pct.round(0).sum()) < 20:
    cluster_pct[cluster_pct < 0.5] = cluster_pct[cluster_pct < 0.5] + 1.0

display(cluster_pct)
print('Total food: ', int(cluster_pct.round(0).sum()))

df_food_joined.reset_index(inplace=True)
df_food_joined.head(3)

df_user_food_joined['cluster_pct'] = df_user_food_joined['cluster'].apply(lambda c:
cluster_pct[c])
df_user_food_joined.drop(columns=columns_to_cluster_scaled, inplace=True)
df_user_food_joined.head(3)

final_Food = pd.DataFrame()

for ncluster, pct in cluster_pct.items():
    foods = df_food_joined[df_food_joined['cluster'] == ncluster].sample(n=int(round(pct,
0)))
    final_Food = pd.concat([final_Food,foods], ignore_index=True)
    if len(final_Food) > 20 :
        flag = 20 - len(final_Food)
        final_Food = final_Food[:flag]
# final_Food.head(10)

final_Food.info()

"""**Appying Random Forest**"""

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

X=df_food_scaled[columns_to_cluster_scaled]  # Features
y=df_food_scaled['cluster']

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

clf=RandomForestClassifier(n_estimators=100)
clf.fit(X_train,y_train)
#Train the model using the training sets
y_pred=clf.predict(X_test)
y_pred

clf.estimators_[0]
sample_tree = clf.estimators_[4]

# Importing required packages for visualization
```

```
from IPython.display import Image
from six import StringIO
from sklearn.tree import export_graphviz
import pydotplus, graphviz

def get_dt_graph(dt_classifier):
    dot_data = StringIO()
    export_graphviz(dt_classifier, out_file=dot_data, filled=True,rounded=True,
                    feature_names=X.columns,
                    class_names=['K1', 'K2','K3', 'K4','K5', 'K6'])
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return graph

gph = get_dt_graph(sample_tree)
Image(gph.create_png())

clf.feature_importances_

imp_df = pd.DataFrame({
    "Nutritions": X_train.columns,
    "Importance": clf.feature_importances_
})

imp_df.sort_values(by="Importance", ascending=False)

from sklearn import metrics
from sklearn.metrics import confusion_matrix

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
cm

# Creating a dataframe for a array-formatted Confusion matrix,so it will be easy for
plotting.
cm_df = pd.DataFrame(cm)

#Plotting the confusion matrix
ax = sns.heatmap(cm_df, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()

final_Food[['Description']]

chartval=[]
for i in columns_to_cluster_scaled:
  chartval.append(final_Food[i].sum())
```

```
labels= columns_to_cluster
sizes= chartval
plt.pie(sizes,labels=labels, startangle=90, shadow=True,autopct='%1.2f%%')
# explode=(0.1, 0.1, 0.1, 0.1,0.1, 0.1, 0.1),
plt.title('Food Percentage For given Food')
plt.axis('equal')
plt.show()
```