Meal Plan Recommendation

Meal plan recommendation system that uses content-based filtering to recommend meals based on dietary restrictions and preferences.This system aims to promote healthier eating habits by offering personalized meal options that adhere to user-specified dietary needs. The solution will be used in a website application. This application intends to build an unsupervised learning model from unlabelled data.

Reading the data

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data=pd.read_csv('Data/recipes.csv')
data.head()
```

Out[2]:

	RecipeId	Name	AuthorId	AuthorName	CookTime	PrepTime	TotalTime	DatePublished	Description	
0	38	Low-Fat Berry Blue Frozen Dessert	1533	Dancer	PT24H	PT45M	PT24H45M	1999-08-09T21:46:00Z	Make and share this Low-Fat Berry Blue Frozen ...	c("https://img.sndimg.com/f
1	39	Biryani	1567	elly9812	PT25M	PT4H	PT4H25M	1999-08-29T13:12:00Z	Make and share this Biryani recipe from Food.com.	c("https://img.sndimg.com/f
2	40	Best Lemonade	1566	Stephen Little	PT5M	PT30M	PT35M	1999-09-05T19:52:00Z	This is from one of my first Good House Keepi...	c("https://img.sndimg.com/f
3	41	Carina's Tofu-Vegetable Kebabs	1586	Cyclopz	PT20M	PT24H	PT24H20M	1999-09-03T14:54:00Z	This dish is best prepared a day in advance to...	c("https://img.sndimg.com/f
4	42	Cabbage Soup	1538	Duckie067	PT30M	PT20M	PT50M	1999-09-19T06:19:00Z	Make and share this Cabbage Soup recipe from F...	"https://img.sndimg.com/foc

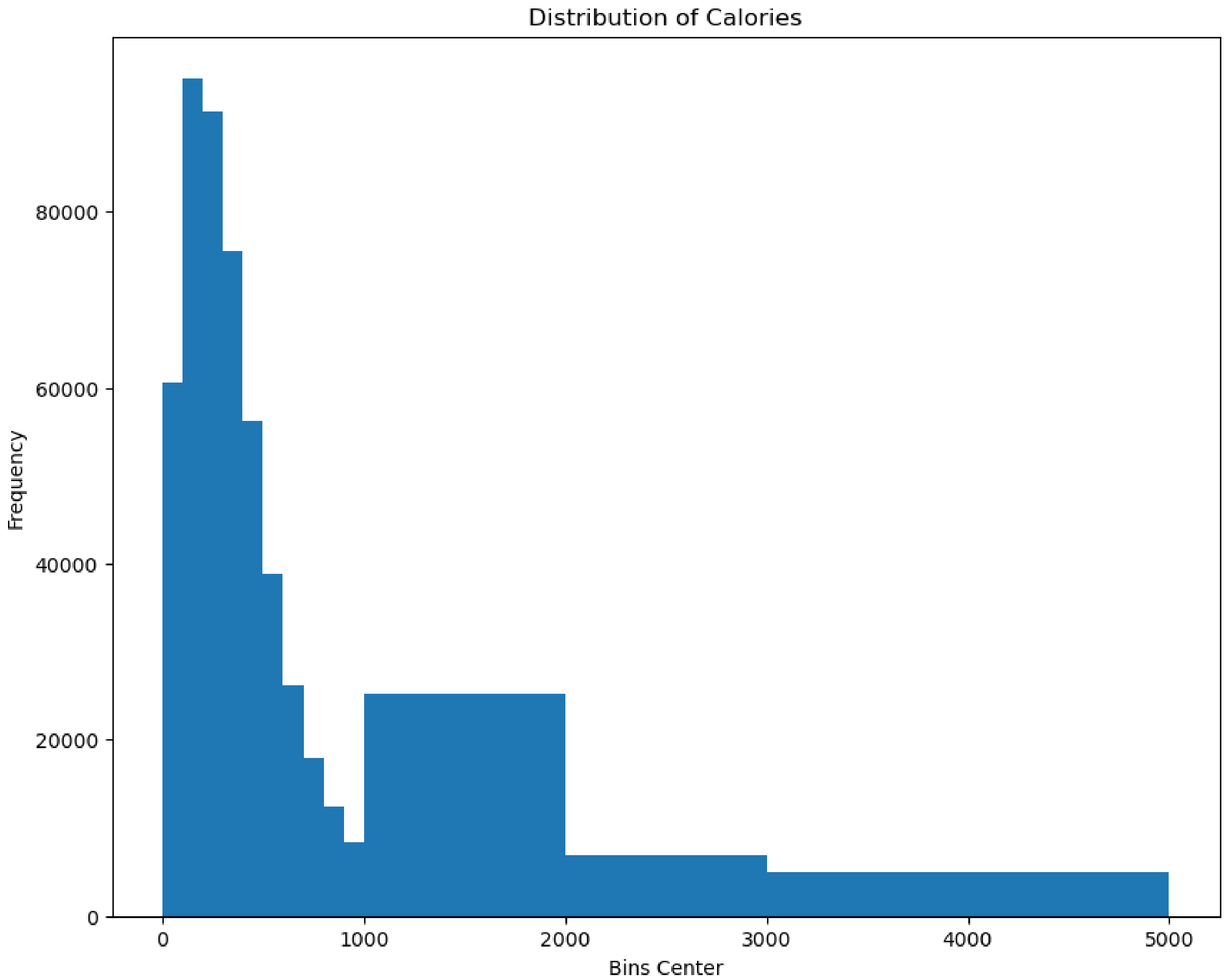
5 rows × 28 columns

In [3]:
```
data.info()
```

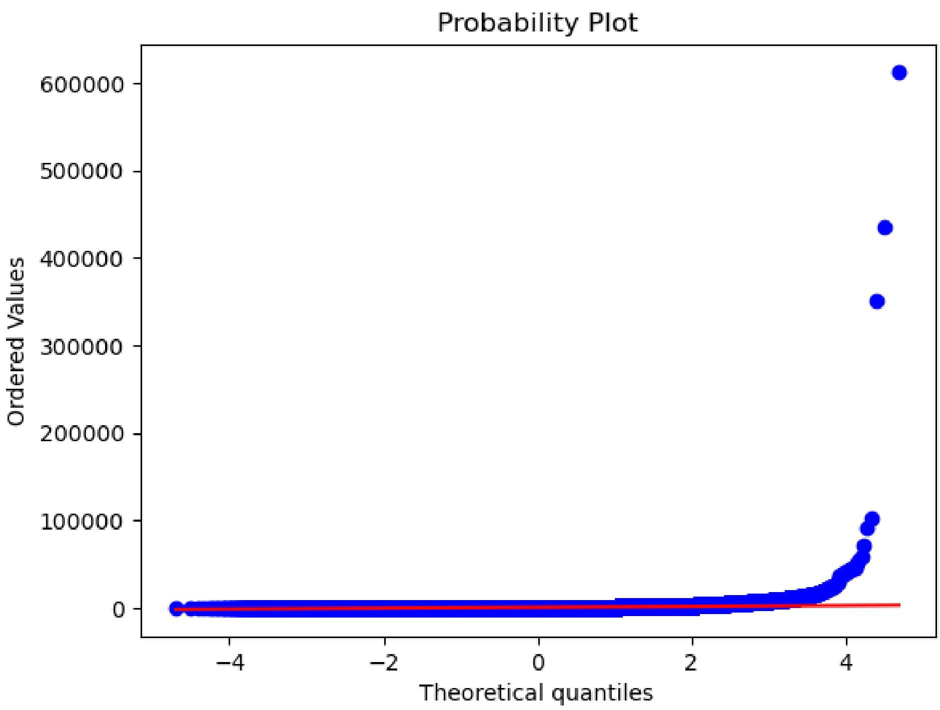
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522517 entries, 0 to 522516
Data columns (total 28 columns):
 #   Column                      Non-Null Count   Dtype  
---  ------                      --------------   -----  
 0   RecipeId                    522517 non-null  int64  
 1   Name                        522517 non-null  object 
 2   AuthorId                    522517 non-null  int64  
 3   AuthorName                  522517 non-null  object 
 4   CookTime                    439972 non-null  object 
 5   PrepTime                    522517 non-null  object 
 6   TotalTime                   522517 non-null  object 
 7   DatePublished               522517 non-null  object 
 8   Description                 522512 non-null  object 
 9   Images                      522516 non-null  object 
 10  RecipeCategory              521766 non-null  object 
 11  Keywords                    505280 non-null  object 
 12  RecipeIngredientQuantities  522514 non-null  object 
 13  RecipeIngredientParts       522517 non-null  object 
 14  AggregatedRating            269294 non-null  float64
 15  ReviewCount                 275028 non-null  float64
 16  Calories                    522517 non-null  float64
 17  FatContent                  522517 non-null  float64
 18  SaturatedFatContent         522517 non-null  float64
 19  CholesterolContent          522517 non-null  float64
 20  SodiumContent               522517 non-null  float64
 21  CarbohydrateContent         522517 non-null  float64
 22  FiberContent                522517 non-null  float64
 23  SugarContent                522517 non-null  float64
 24  ProteinContent              522517 non-null  float64
 25  RecipeServings              339606 non-null  float64
 26  RecipeYield                 174446 non-null  object 
 27  RecipeInstructions          522517 non-null  object 
dtypes: float64(12), int64(2), object(14)
memory usage: 111.6+ MB
```

In [4]:

```
fig, ax = plt.subplots(figsize=(10, 8))
plt.title('Distribution of Calories')
plt.ylabel('Frequency')
plt.xlabel('Bins Center')
ax.hist(data.Calories.to_numpy(),bins=[0,100,200,300,400,500,600,700,800,900,1000,1000,2000,3000,5000
plt.show()
```



In [5]:
```
import pylab
import scipy.stats as stats
stats.probplot(data.Calories.to_numpy(), dist="norm", plot=pylab)
pylab.show()
```



The plot shows a significant deviation from the straight line, indicating that the Calories data does not follow a normal distribution.The plot indicates positive skewness, as the data points curve upwards away from the line at higher quantiles. This suggests that there are a few data points with much higher calorie values than the rest.

Preparing Data

Selecting the columns I am interested in order to build a model that takes into account the recipes nutritional characteristics.

In [6]:
```
dataset=data.copy()
columns=['RecipeId','Name','CookTime','PrepTime','TotalTime','RecipeIngredientParts','Calories','FatC
dataset=dataset[columns]
```

Setting max values to the parameters the user will be able to set for the recommendation of meals

In [7]:
```
max_Calories=2000
max_daily_fat=100
max_daily_Saturatedfat=13
max_daily_Cholesterol=300
max_daily_Sodium=2300
max_daily_Carbohydrate=325
max_daily_Fiber=40
max_daily_Sugar=40
max_daily_Protein=200
max_list=[max_Calories,max_daily_fat,max_daily_Saturatedfat,max_daily_Cholesterol,max_daily_Sodium,ma
```

In [8]:
```
extracted_data=dataset.copy()
for column,maximum in zip(extracted_data.columns[6:15],max_list):
    extracted_data=extracted_data[extracted_data[column]<maximum]
```

In [9]:
```
extracted_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 375703 entries, 0 to 522515
Data columns (total 16 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   RecipeId               375703 non-null  int64  
 1   Name                   375703 non-null  object 
 2   CookTime               313207 non-null  object 
 3   PrepTime               375703 non-null  object 
 4   TotalTime              375703 non-null  object 
 5   RecipeIngredientParts  375703 non-null  object 
 6   Calories               375703 non-null  float64
 7   FatContent             375703 non-null  float64
 8   SaturatedFatContent    375703 non-null  float64
 9   CholesterolContent     375703 non-null  float64
 10  SodiumContent          375703 non-null  float64
 11  CarbohydrateContent    375703 non-null  float64
 12  FiberContent           375703 non-null  float64
 13  SugarContent           375703 non-null  float64
 14  ProteinContent         375703 non-null  float64
 15  RecipeInstructions     375703 non-null  object 
dtypes: float64(9), int64(1), object(6)
memory usage: 48.7+ MB
```

Exploring Correlation

Understanding the relationships between different nutritional components can help in designing balanced meal plans. For example, if calories are highly correlated with fat, meals with higher fat content are likely to have higher calories.

In [10]:
```
extracted_data.iloc[:,6:15].corr()
```

Out[10]:

	Calories	FatContent	SaturatedFatContent	CholesterolContent	SodiumContent	CarbohydrateContent	FiberCont
Calories	1.000000	0.767356	0.603317	0.478934	0.501082	0.711640	0.458
FatContent	0.767356	1.000000	0.767357	0.440515	0.381944	0.223549	0.192
SaturatedFatContent	0.603317	0.767357	1.000000	0.512186	0.319671	0.176623	0.044
CholesterolContent	0.478934	0.440515	0.512186	1.000000	0.335843	0.066104	-0.047
SodiumContent	0.501082	0.381944	0.319671	0.335843	1.000000	0.294636	0.260
CarbohydrateContent	0.711640	0.223549	0.176623	0.066104	0.294636	1.000000	0.580
FiberContent	0.458711	0.192142	0.044003	-0.047346	0.260479	0.580535	1.000
SugarContent	0.180895	0.042603	0.090721	-0.036112	-0.055518	0.390120	0.068
ProteinContent	0.689447	0.468088	0.388618	0.675302	0.500457	0.255447	0.273

Preprocessing

In [11]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
prep_data=scaler.fit_transform(extracted_data.iloc[:,6:15].to_numpy())
```

In [12]:
```
prep_data
```

Out[12]:
```
array([[-0.55093359, -0.91281917, -0.77924852, ...,  0.15672078,
         2.35502102, -0.68338127],
       [ 1.47428542,  1.13139595, -0.0647135 , ...,  3.91055068,
         2.56324444,  1.25158691],
       [-0.92414618, -1.11248669, -1.12222533, ...,  0.4855234 ,
         0.98513013, -0.60183088],
       ...,
       [ 0.49162165,  0.73206091,  1.85024037, ..., -0.61048534,
         1.76322815, -0.56476253],
       [ 0.25704672,  0.03797856,  1.02137974, ..., -0.61048534,
         1.54404561, -0.63148557],
       [-1.40937801, -1.09347074, -1.12222533, ..., -0.82968708,
        -0.94367625, -0.74269064]])
```

Training the Model

In [13]:
```
from sklearn.neighbors import NearestNeighbors
neigh = NearestNeighbors(metric='cosine',algorithm='brute')
neigh.fit(prep_data)
```

Out[13]:
```
▾ NearestNeighbors
NearestNeighbors(algorithm='brute',
metric='cosine')
```

In [14]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import FunctionTransformer
transformer = FunctionTransformer(neigh.kneighbors,kw_args={'return_distance':False})
pipeline=Pipeline([('std_scaler',scaler),('NN',transformer)])
```

In [15]:
```
pipeline.transform(extracted_data.iloc[0:1,6:15].to_numpy())[0]
```

Out[15]:
```
array([     0, 333440, 349044, 109248,  19679])
```

Testing the Model

In [16]:
```
extracted_data.iloc[pipeline.transform(extracted_data.iloc[0:1,6:15].to_numpy())[0]]
```

Out[16]:

	RecipeId	Name	CookTime	PrepTime	TotalTime	RecipeIngredientParts	Calories	FatContent	SaturatedFatContent	Ch
0	38	Low-Fat Berry Blue Frozen Dessert	PT24H	PT45M	PT24H45M	c("blueberries", "granulated sugar", "vanilla ...	170.9	2.5	1.3	
463750	480841	Mango Salsa	PT5M	PT10M	PT15M	c("fresh mango", "tomatoes", "sweet onion", "f...	152.5	0.8	0.2	
485171	503065	Glazed Pineapple With Cinnamon Creme Fraiche	PT10M	PT10M	PT20M	c("lime", "honey", "ground cinnamon", "ground ...	172.5	2.2	1.2	
158110	165636	Lemon Float Punch	PT120H	PT5M	PT120H5M	c("lemons", "sugar", "water", "ginger ale", "l...	158.4	1.7	0.9	
28595	32172	L & B's Concoction	PT5M	PT5M	PT10M	c("strawberry", "strawberry", "milk", "blueber...	167.3	2.0	1.0	

In [17]:
```
extracted_data[extracted_data['RecipeIngredientParts'].str.contains("egg",regex=False)]
```

Out[17]:

	RecipeId	Name	CookTime	PrepTime	TotalTime	RecipeIngredientParts	Calories	FatContent	SaturatedFatContent	Ch
3	41	Carina's Tofu-Vegetable Kebabs	PT20M	PT24H	PT24H20M	c("extra firm tofu", "eggplant", "zucchini", "...	536.1	24.0	3.8	
7	45	Buttermilk Pie With Gingersnap Crumb Crust	PT50M	PT30M	PT1H20M	c("sugar", "margarine", "egg", "flour", "salt"...	228.0	7.1	1.7	
12	50	Biscotti Di Prato	PT50M	PT20M	PT1H10M	c("flour", "sugar", "baking powder", "salt", "...	89.4	2.6	0.3	
18	56	Buttermilk Pie	PT1H	PT20M	PT1H20M	c("butter", "margarine", "sugar", "flour", "eg...	395.9	19.1	9.8	
22	60	Blueberry Dessert	NaN	PT35M	PT35M	c("Bisquick baking mix", "sugar", "butter", "m...	381.1	17.3	8.8	
...	...	...	...	...	...	...	...	...	...	
522484	541351	Spinach & Mushroom Quiche with Boursin	PT1H	PT20M	PT1H20M	c("butter", "onion", "sweet pepper", "carrots"...	197.6	11.0	4.0	
522490	541357	Chocolate Rum Snowballs	PT8M	PT15M	PT23M	c("rolled oats", "sweetened flaked coconut", "...	127.8	6.2	4.1	
522500	541367	Thick Peanut Pancakes	PT10M	PT45M	PT55M	c("plain flour", "baking powder", "baking soda...	712.9	25.4	8.6	
522510	541377	Slow-Cooker Classic Coffee Cake	PT3H	PT20M	PT3H20M	c("all-purpose flour", "brown sugar", "butter"...	358.9	19.8	10.5	
522512	541379	Meg's Fresh Ginger Gingerbread	PT35M	PT1H	PT1H35M	c("fresh ginger", "unsalted butter", "dark bro...	316.6	12.5	7.6	

83668 rows × 16 columns

In [18]:
```
extracted_data[~extracted_data['RecipeIngredientParts'].str.contains("chicken",regex=False)]
```

Out[18]:

	RecipeId	Name	CookTime	PrepTime	TotalTime	RecipeIngredientParts	Calories	FatContent	SaturatedFatContent	Ch
0	38	Low-Fat Berry Blue Frozen Dessert	PT24H	PT45M	PT24H45M	c("blueberries", "granulated sugar", "vanilla ...	170.9	2.5	1.3	
3	41	Carina's Tofu-Vegetable Kebabs	PT20M	PT24H	PT24H20M	c("extra firm tofu", "eggplant", "zucchini", "...	536.1	24.0	3.8	
4	42	Cabbage Soup	PT30M	PT20M	PT50M	c("plain tomato juice", "cabbage", "onion", "c...	103.6	0.4	0.1	
7	45	Buttermilk Pie With Gingersnap Crumb Crust	PT50M	PT30M	PT1H20M	c("sugar", "margarine", "egg", "flour", "salt"...	228.0	7.1	1.7	
8	46	A Jad - Cucumber Pickle	NaN	PT25M	PT25M	c("rice vinegar", "haeo")	4.3	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
522508	541375	Amazing Ground Beef Stroganoff	PT20M	PT30M	PT50M	c("hamburger", "onion", "celery", "water chest...	422.3	28.6	12.6	
522509	541376	Spanish Coffee with Tia Maria	NaN	PT10M	PT10M	c("lemon wedge", "granulated sugar", "cognac",...	84.3	2.1	1.2	
522510	541377	Slow-Cooker Classic Coffee Cake	PT3H	PT20M	PT3H20M	c("all-purpose flour", "brown sugar", "butter"...	358.9	19.8	10.5	
522512	541379	Meg's Fresh Ginger Gingerbread	PT35M	PT1H	PT1H35M	c("fresh ginger", "unsalted butter", "dark bro...	316.6	12.5	7.6	
522515	541382	Quick & Easy Asian Cucumber Salmon Rolls	NaN	PT15M	PT15M	c("wasabi paste", "dill", "English cucumber", ...	16.1	0.6	0.1	

326317 rows × 16 columns

Creating Function

In [19]:
```
def scaling(dataframe):
    scaler=StandardScaler()
    prep_data=scaler.fit_transform(dataframe.iloc[:,6:15].to_numpy())
    return prep_data,scaler

def nn_predictor(prep_data):
    neigh = NearestNeighbors(metric='cosine',algorithm='brute')
    neigh.fit(prep_data)
    return neigh

def build_pipeline(neigh,scaler,params):
    transformer = FunctionTransformer(neigh.kneighbors,kw_args=params)
    pipeline=Pipeline([('std_scaler',scaler),('NN',transformer)])
    return pipeline

def extract_data(dataframe,ingredient_filter,max_nutritional_values):
    extracted_data=dataframe.copy()
    for column,maximum in zip(extracted_data.columns[6:15],max_nutritional_values):
        extracted_data=extracted_data[extracted_data[column]<maximum]
    if ingredient_filter!=None:
        for ingredient in ingredient_filter:
            extracted_data=extracted_data[extracted_data['RecipeIngredientParts'].str.contains(ingre

    return extracted_data

def apply_pipeline(pipeline,_input,extracted_data):
    return extracted_data.iloc[pipeline.transform(_input)[0]]
```

```python
def recommend(dataframe,_input,max_nutritional_values,ingredient_filter=None,params={'return_distanc
    extracted_data=extract_data(dataframe,ingredient_filter,max_nutritional_values)
    prep_data,scaler=scaling(extracted_data)
    neigh=nn_predictor(prep_data)
    pipeline=build_pipeline(neigh,scaler,params)
    return apply_pipeline(pipeline,_input,extracted_data)
```

In [20]:
```python
test_input=extracted_data.iloc[0:1,6:15].to_numpy()
recommend(dataset,test_input,max_list)
```

Out[20]:

	RecipeId	Name	CookTime	PrepTime	TotalTime	RecipeIngredientParts	Calories	FatContent	SaturatedFatContent	Ch
0	38	Low-Fat Berry Blue Frozen Dessert	PT24H	PT45M	PT24H45M	c("blueberries", "granulated sugar", "vanilla ...	170.9	2.5	1.3	
463750	480841	Mango Salsa	PT5M	PT10M	PT15M	c("fresh mango", "tomatoes", "sweet onion", "f...	152.5	0.8	0.2	
485171	503065	Glazed Pineapple With Cinnamon Creme Fraiche	PT10M	PT10M	PT20M	c("lime", "honey", "ground cinnamon", "ground ...	172.5	2.2	1.2	
158110	165636	Lemon Float Punch	PT120H	PT5M	PT120H5M	c("lemons", "sugar", "water", "ginger ale", "l...	158.4	1.7	0.9	
28595	32172	L & B's Concoction	PT5M	PT5M	PT10M	c("strawberry", "strawberry", "milk", "blueber...	167.3	2.0	1.0	

In [21]:
```python
dataset.to_csv('Data/dataset.csv',compression='gzip',index=False)
```