In [40]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
a=pd.read_csv("C:\All Datasets\All_Diets.csv")
a
```

Out[2]:

| | Diet_type | Recipe_name | Cuisine_type | Protein(g) | Carbs(g) | Fat(g) | Extraction_day | Extractio |
|---|---|---|---|---|---|---|---|---|
| **0** | paleo | Bone Broth From 'Nom Nom Paleo' | american | 5.22 | 1.29 | 3.20 | 2022-10-16 | 1 |
| **1** | paleo | Paleo Effect Asian-Glazed Pork Sides, A Sweet ... | south east asian | 181.55 | 28.62 | 146.14 | 2022-10-16 | 1 |
| **2** | paleo | Paleo Pumpkin Pie | american | 30.91 | 302.59 | 96.76 | 2022-10-16 | 1 |
| **3** | paleo | Strawberry Guacamole recipes | mexican | 9.62 | 75.78 | 59.89 | 2022-10-16 | 1 |
| **4** | paleo | Asian Cauliflower Fried "Rice" From 'Nom Nom P... | chinese | 39.84 | 54.08 | 71.55 | 2022-10-16 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **7801** | dash | Brown Butter-Sunchoke Soup With Brussels Sprou... | italian | 85.20 | 288.14 | 137.15 | 2022-10-16 | 2 |
| **7802** | dash | Make-Your-Own-Salad | american | 141.98 | 123.18 | 323.50 | 2022-10-16 | 2 |
| **7803** | dash | Luis Buñuel Dry Martini | world | 0.01 | 0.39 | 0.00 | 2022-10-16 | 2 |
| **7804** | dash | Cornflake Semi-Fried Chicken Tenders | american | 155.38 | 239.88 | 260.84 | 2022-10-16 | 2 |
| **7805** | dash | Emeril's Classic Manhattan | american | 0.02 | 0.83 | 0.00 | 2022-10-16 | 2 |

7806 rows × 8 columns

In [3]:
```
a.isnull().sum()
```

Out[3]:
```
Diet_type          0
Recipe_name        0
Cuisine_type       0
Protein(g)         0
Carbs(g)           0
Fat(g)             0
Extraction_day     0
Extraction_time    0
dtype: int64
```

In [5]:
```
from sklearn.preprocessing import LabelEncoder
l=LabelEncoder()
a["Recipe_name"]=l.fit_transform(a["Recipe_name"])
a["Cuisine_type"]=l.fit_transform(a["Cuisine_type"])
a["Diet_type"]=l.fit_transform(a["Diet_type"])
```

In [6]:
```
a
```

Out[6]:

| | Diet_type | Recipe_name | Cuisine_type | Protein(g) | Carbs(g) | Fat(g) | Extraction_day | Extractio |
|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 509 | 0 | 5.22 | 1.29 | 3.20 | 2022-10-16 | 1 |
| **1** | 3 | 4558 | 17 | 181.55 | 28.62 | 146.14 | 2022-10-16 | 1 |
| **2** | 3 | 4731 | 0 | 30.91 | 302.59 | 96.76 | 2022-10-16 | 1 |
| **3** | 3 | 5965 | 13 | 9.62 | 75.78 | 59.89 | 2022-10-16 | 1 |
| **4** | 3 | 194 | 5 | 39.84 | 54.08 | 71.55 | 2022-10-16 | 1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **7801** | 0 | 577 | 9 | 85.20 | 288.14 | 137.15 | 2022-10-16 | 2 |
| **7802** | 0 | 3321 | 0 | 141.98 | 123.18 | 323.50 | 2022-10-16 | 2 |
| **7803** | 0 | 3295 | 18 | 0.01 | 0.39 | 0.00 | 2022-10-16 | 2 |
| **7804** | 0 | 1095 | 0 | 155.38 | 239.88 | 260.84 | 2022-10-16 | 2 |
| **7805** | 0 | 1588 | 0 | 0.02 | 0.83 | 0.00 | 2022-10-16 | 2 |

7806 rows × 8 columns

In [9]:
```
a["Extraction_day"]=pd.to_datetime(a["Extraction_day"])
a["By_year"]=a["Extraction_day"].dt.year
a["By_Day"]=a["Extraction_day"].dt.day
a["By_month"]=a["Extraction_day"].dt.month
```

In [11]:
```
del a["Extraction_day"]
```

In [12]:
```
a
```

Out[12]:

| | Diet_type | Recipe_name | Cuisine_type | Protein(g) | Carbs(g) | Fat(g) | Extraction_time | By_year |
|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 509 | 0 | 5.22 | 1.29 | 3.20 | 17:20:09 | 2022 |
| **1** | 3 | 4558 | 17 | 181.55 | 28.62 | 146.14 | 17:20:09 | 2022 |
| **2** | 3 | 4731 | 0 | 30.91 | 302.59 | 96.76 | 17:20:09 | 2022 |
| **3** | 3 | 5965 | 13 | 9.62 | 75.78 | 59.89 | 17:20:09 | 2022 |
| **4** | 3 | 194 | 5 | 39.84 | 54.08 | 71.55 | 17:20:09 | 2022 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **7801** | 0 | 577 | 9 | 85.20 | 288.14 | 137.15 | 20:40:44 | 2022 |
| **7802** | 0 | 3321 | 0 | 141.98 | 123.18 | 323.50 | 20:40:44 | 2022 |
| **7803** | 0 | 3295 | 18 | 0.01 | 0.39 | 0.00 | 20:40:44 | 2022 |
| **7804** | 0 | 1095 | 0 | 155.38 | 239.88 | 260.84 | 20:40:44 | 2022 |
| **7805** | 0 | 1588 | 0 | 0.02 | 0.83 | 0.00 | 20:40:44 | 2022 |

7806 rows × 10 columns

In [26]:
```
a[["hour","min","sec"]]=a["Extraction_time"].str.split(":",expand=True)
```

In [28]:
```
del (a["Extraction_time"])
```

In [29]:
```
a
```

Out[29]:

| | Diet_type | Recipe_name | Cuisine_type | Protein(g) | Carbs(g) | Fat(g) | By_year | By_Day | By_m |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 509 | 0 | 5.22 | 1.29 | 3.20 | 2022 | 16 | |
| **1** | 3 | 4558 | 17 | 181.55 | 28.62 | 146.14 | 2022 | 16 | |
| **2** | 3 | 4731 | 0 | 30.91 | 302.59 | 96.76 | 2022 | 16 | |
| **3** | 3 | 5965 | 13 | 9.62 | 75.78 | 59.89 | 2022 | 16 | |
| **4** | 3 | 194 | 5 | 39.84 | 54.08 | 71.55 | 2022 | 16 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **7801** | 0 | 577 | 9 | 85.20 | 288.14 | 137.15 | 2022 | 16 | |
| **7802** | 0 | 3321 | 0 | 141.98 | 123.18 | 323.50 | 2022 | 16 | |
| **7803** | 0 | 3295 | 18 | 0.01 | 0.39 | 0.00 | 2022 | 16 | |
| **7804** | 0 | 1095 | 0 | 155.38 | 239.88 | 260.84 | 2022 | 16 | |
| **7805** | 0 | 1588 | 0 | 0.02 | 0.83 | 0.00 | 2022 | 16 | |

7806 rows × 12 columns

In [31]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7806 entries, 0 to 7805
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Diet_type     7806 non-null   int32  
 1   Recipe_name   7806 non-null   int64  
 2   Cuisine_type  7806 non-null   int64  
 3   Protein(g)    7806 non-null   float64
 4   Carbs(g)      7806 non-null   float64
 5   Fat(g)        7806 non-null   float64
 6   By_year       7806 non-null   int64  
 7   By_Day        7806 non-null   int64  
 8   By_month      7806 non-null   int64  
 9   hour          7806 non-null   object 
 10  min           7806 non-null   object 
 11  sec           7806 non-null   object 
dtypes: float64(3), int32(1), int64(5), object(3)
memory usage: 701.4+ KB
```

In [35]:
```
a["hour"]=a["hour"].astype(int)
a["min"]=a["min"].astype(int)
a["sec"]=a["sec"].astype(int)
```

In [36]:
```
a
```

Out[36]:

| | Diet_type | Recipe_name | Cuisine_type | Protein(g) | Carbs(g) | Fat(g) | By_year | By_Day | By_m |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 509 | 0 | 5.22 | 1.29 | 3.20 | 2022 | 16 | |
| **1** | 3 | 4558 | 17 | 181.55 | 28.62 | 146.14 | 2022 | 16 | |
| **2** | 3 | 4731 | 0 | 30.91 | 302.59 | 96.76 | 2022 | 16 | |
| **3** | 3 | 5965 | 13 | 9.62 | 75.78 | 59.89 | 2022 | 16 | |
| **4** | 3 | 194 | 5 | 39.84 | 54.08 | 71.55 | 2022 | 16 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **7801** | 0 | 577 | 9 | 85.20 | 288.14 | 137.15 | 2022 | 16 | |
| **7802** | 0 | 3321 | 0 | 141.98 | 123.18 | 323.50 | 2022 | 16 | |
| **7803** | 0 | 3295 | 18 | 0.01 | 0.39 | 0.00 | 2022 | 16 | |
| **7804** | 0 | 1095 | 0 | 155.38 | 239.88 | 260.84 | 2022 | 16 | |
| **7805** | 0 | 1588 | 0 | 0.02 | 0.83 | 0.00 | 2022 | 16 | |

7806 rows × 12 columns

In [37]:

```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7806 entries, 0 to 7805
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Diet_type     7806 non-null   int32  
 1   Recipe_name   7806 non-null   int64  
 2   Cuisine_type  7806 non-null   int64  
 3   Protein(g)    7806 non-null   float64
 4   Carbs(g)      7806 non-null   float64
 5   Fat(g)        7806 non-null   float64
 6   By_year       7806 non-null   int64  
 7   By_Day        7806 non-null   int64  
 8   By_month      7806 non-null   int64  
 9   hour          7806 non-null   int32  
 10  min           7806 non-null   int32  
 11  sec           7806 non-null   int32  
dtypes: float64(3), int32(4), int64(5)
memory usage: 610.0 KB
```

In [51]:

```
x=a.iloc[:,1:].values
```

In [52]:

```
from sklearn.cluster import KMeans
km=KMeans(n_clusters=5,init="k-means++",random_state=20)
km.fit(x)
```

Out[52]:

```
KMeans(n_clusters=5, random_state=20)
```

In [73]:
```
a
```

Out[73]:

|  | Diet_type | Recipe_name | Cuisine_type | Protein(g) | Carbs(g) | Fat(g) | By_year | By_Day | By_m |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 3 | 509 | 0 | 5.22 | 1.29 | 3.20 | 2022 | 16 |  |
| **1** | 3 | 4558 | 17 | 181.55 | 28.62 | 146.14 | 2022 | 16 |  |
| **2** | 3 | 4731 | 0 | 30.91 | 302.59 | 96.76 | 2022 | 16 |  |
| **3** | 3 | 5965 | 13 | 9.62 | 75.78 | 59.89 | 2022 | 16 |  |
| **4** | 3 | 194 | 5 | 39.84 | 54.08 | 71.55 | 2022 | 16 |  |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |  |
| **7801** | 0 | 577 | 9 | 85.20 | 288.14 | 137.15 | 2022 | 16 |  |
| **7802** | 0 | 3321 | 0 | 141.98 | 123.18 | 323.50 | 2022 | 16 |  |
| **7803** | 0 | 3295 | 18 | 0.01 | 0.39 | 0.00 | 2022 | 16 |  |
| **7804** | 0 | 1095 | 0 | 155.38 | 239.88 | 260.84 | 2022 | 16 |  |
| **7805** | 0 | 1588 | 0 | 0.02 | 0.83 | 0.00 | 2022 | 16 |  |

7806 rows × 13 columns

In [77]:
```
x1=a.iloc[:,1:-1].values
y2=a.iloc[:,0].values
```

In [79]:
```
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
x2=s.fit_transform(x1)
```

In [80]:
```
a["Pred Diet Type"].value_counts()
```

Out[80]:
```
3    1810
0    1510
1    1509
4    1496
2    1481
Name: Pred Diet Type, dtype: int64
```

In [81]:
```
from imblearn.over_sampling import SMOTE
sm=SMOTE()
x4,y4=sm.fit_resample(x2,y2)
```

In [83]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x4,y4,random_state=20,test_size=0
```

In [84]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import BaggingClassifier
```

In [94]:
```
kn=KNeighborsClassifier(n_neighbors=5,metric="minkowski",p=2)
bg=BaggingClassifier(base_estimator=kn,n_estimators=5)
bg.fit(x_train,y_train)
```

Out[94]:
```
BaggingClassifier(base_estimator=KNeighborsClassifier(), n_estimators=5)
```

In [97]:
```
p1=bg.predict(x_train)
p1
```

Out[97]:
```
array([3, 0, 2, ..., 0, 0, 2])
```

In [99]:
```
from sklearn.metrics import accuracy_score
accuracy_score(p1,y_train)*100
```

Out[99]:
```
99.11580148317171
```

In [101]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(p1,y_train)
```

Out[101]:
```
array([[1398,    1,    1,    2,    4],
       [   0, 1379,    4,    4,    2],
       [   0,   16, 1382,    7,    0],
       [   0,    0,   10, 1395,    1],
       [   3,    5,    0,    2, 1396]], dtype=int64)
```

In [104]:
```
from sklearn.metrics import classification_report
classification_report(p1,y_train)
```

Out[104]:
```
'              precision    recall  f1-score   support\n\n           0       1.
00      0.99      1.00      1406\n           1       0.98      0.99      0.99
1389\n           2       0.99      0.98      0.99      1405\n           3
0.99      0.99      0.99      1406\n           4       1.00      0.99      0.99
1406\n\n    accuracy                           0.99      7012\n   macro avg
0.99      0.99      0.99      7012\nweighted avg       0.99      0.99      0.99
7012\n'
```

In [110]:
```
from sklearn.model_selection import StratifiedKFold
sk=StratifiedKFold(n_splits=5,random_state=20,shuffle=True)
sk.get_n_splits(x_train,y_train)
```

Out[110]:
```
5
```

In [111]:
```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import accuracy_score
```

In [112]:
```
bg.fit(x_train,y_train)
```

Out[112]:
```
BaggingClassifier(base_estimator=KNeighborsClassifier(), n_estimators=5)
```

In [117]:

```
scores=cross_val_score(bg,x_train,y_train)
pred=cross_val_predict(bg,x_train,y_train)
print((scores)*100)
print(pred)
ac=accuracy_score(pred,y_train)*100
print(ac)
```

```
[97.50534569 98.43193158 98.28815977 98.07417974 99.14407989]
[3 0 2 ... 0 0 2]
98.27438676554479
```

In [ ]: