```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import seaborn as sns
```

```
dataset = pd.read_csv("mcdonalds.csv")
dataset
```

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	hea
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	
4	No	Yes	No	Yes	Yes	Yes	Yes	No	No	
...	...	...	...	...	...	...	...	...	...	
1448	No	Yes	No	Yes	Yes	No	No	No	Yes	
1449	Yes	Yes	No	Yes	No	No	Yes	Yes	No	
1450	Yes	Yes	No	Yes	No	Yes	No	Yes	Yes	

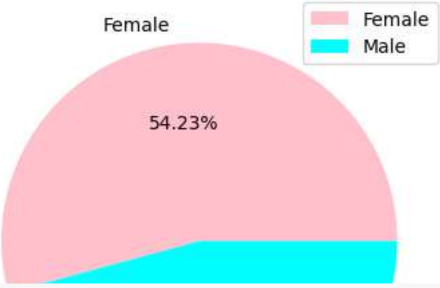
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1453 entries, 0 to 1452
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   yummy           1453 non-null   object
 1   convenient      1453 non-null   object
 2   spicy           1453 non-null   object
 3   fattening       1453 non-null   object
 4   greasy          1453 non-null   object
 5   fast            1453 non-null   object
 6   cheap           1453 non-null   object
 7   tasty           1453 non-null   object
 8   expensive       1453 non-null   object
 9   healthy         1453 non-null   object
 10  disgusting      1453 non-null   object
 11  Like            1453 non-null   object
 12  Age             1453 non-null   int64
 13  VisitFrequency  1453 non-null   object
 14  Gender          1453 non-null   object
dtypes: int64(1), object(14)
memory usage: 170.4+ KB
```

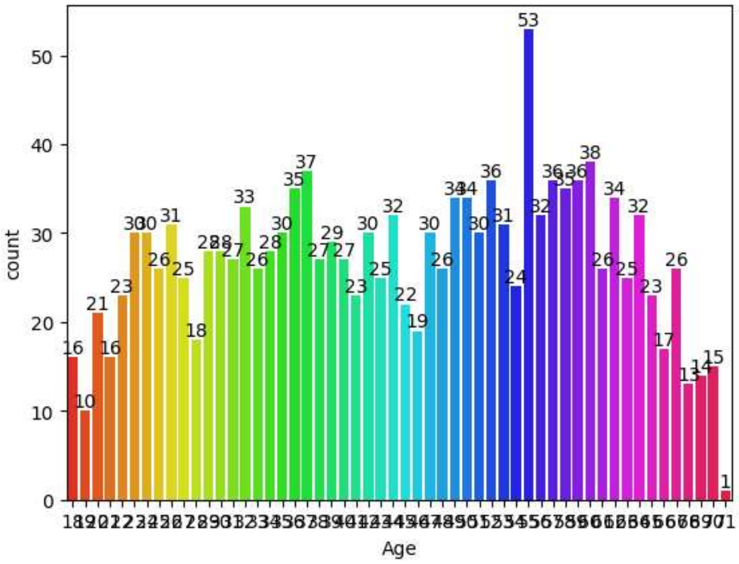
```
dataset.mean()
```

```
<ipython-input-5-e55bc0ed4499>:1: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future ver
  dataset.mean()
Age    44.604955
dtype: float64
```

```
Gender = ["Female", "Male"]
Color = ["pink", "cyan"]
Size = dataset["Gender"].value_counts()
plt.pie(Size, labels=Gender, colors=Color, autopct="%.2f%%")
plt.legend()
plt.show()
```



```
f = sns.countplot(x=dataset["Age"], palette='hsv')
f.bar_label(f.containers[0])
plt.rcParams['figure.figsize'] = (25, 8)
```



```
dataset["Like"]=dataset["Like"].replace({'I hate it!-5': '-5', 'I love it!+5':'+5'})
dataset.dtypes
```

```
yummy             object
convenient        object
spicy             object
fattening         object
greasy            object
fast              object
cheap             object
tasty             object
expensive         object
healthy           object
disgusting        object
Like              object
Age                int64
VisitFrequency    object
Gender            object
dtype: object
```

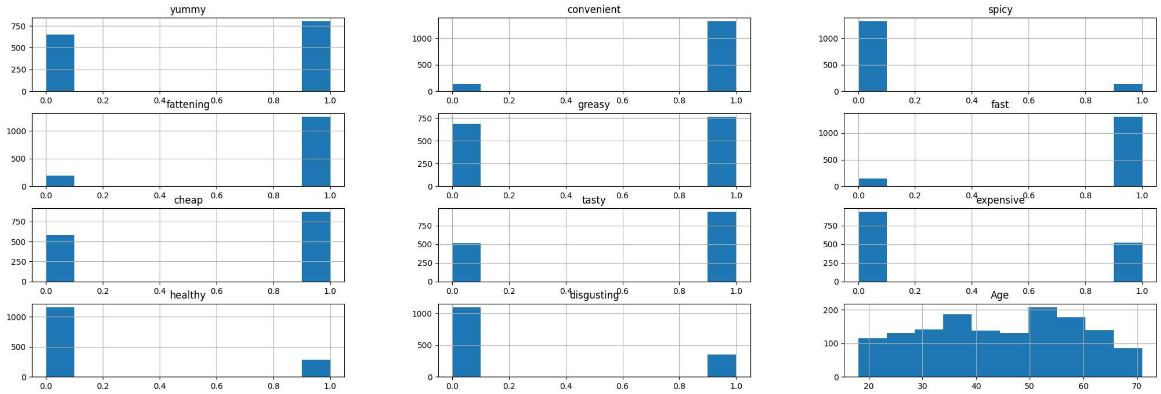
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()

cols=["yummy", "convenient", "spicy", "fattening", "greasy", "fast", "cheap", "tasty", "expensive", "healthy", "disgusting"]

for i in cols:
    dataset[i]=le.fit_transform(dataset[i])
```

```
dataset
```

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age
0	0	1	0	1	0	1	1	0	1	0	0	-3	61
1	1	1	0	1	1	1	1	1	1	0	0	+2	51
2	0	1	1	1	1	1	0	1	1	1	0	+1	62
3	1	1	0	1	1	1	1	1	0	0	1	+4	69

```
dataset.hist()
```

```
array([[<Axes: title={'center': 'yummy'}>,
        <Axes: title={'center': 'convenient'}>,
        <Axes: title={'center': 'spicy'}>],
       [<Axes: title={'center': 'fattening'}>,
        <Axes: title={'center': 'greasy'}>,
        <Axes: title={'center': 'fast'}>],
       [<Axes: title={'center': 'cheap'}>,
        <Axes: title={'center': 'tasty'}>,
        <Axes: title={'center': 'expensive'}>],
       [<Axes: title={'center': 'healthy'}>,
        <Axes: title={'center': 'disgusting'}>,
        <Axes: title={'center': 'Age'}>]], dtype=object)
```



```
x = dataset.loc[:, cols]
x
```

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting
0	0	1	0	1	0	1	1	0	1	0	0
1	1	1	0	1	1	1	1	1	1	0	0
2	0	1	1	1	1	1	0	1	1	1	0
3	1	1	0	1	1	1	1	1	0	0	1
4	0	1	0	1	1	1	1	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...
1448	0	1	0	1	1	0	0	0	1	0	1
1449	1	1	0	1	0	0	1	1	0	1	0
1450	1	1	0	1	0	1	0	1	1	0	0
1451	1	1	0	0	0	1	1	1	0	1	0
1452	0	1	0	1	1	0	0	0	1	0	1

1453 rows × 11 columns

```
from sklearn.decomposition import PCA
from sklearn import preprocessing

pca_data = preprocessing.scale(x)

pca = PCA(n_components=11)
pc = pca.fit_transform(x)
```

```
names=["pc1", "pc2", "pc3", "pc4", "pc5", "pc6", "pc7", "pc8", "pc9", "pc10", "pc11"]
pf = pd.DataFrame(data=pc, columns=names)
pf
```

	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	pc9	pc10	pc
0	0.425367	-0.219079	0.663255	-0.401300	0.201705	-0.389767	-0.211982	0.163235	0.181007	0.515706	-0.5670
1	-0.218638	0.388190	-0.730827	-0.094724	0.044669	-0.086596	-0.095877	-0.034756	0.111476	0.493313	-0.5004
2	0.375415	0.730435	-0.122040	0.692262	0.839643	-0.687406	0.583112	0.364379	-0.322288	0.061759	0.2427
3	-0.172926	-0.352752	-0.843795	0.206998	-0.681415	-0.036133	-0.054284	-0.231477	-0.028003	-0.250678	-0.0510
4	0.187057	-0.807610	0.028537	0.548332	0.854074	-0.097305	-0.457043	0.171758	-0.074409	0.031897	0.0822
...	...	...	...	...	...	...	...	...	...	...	
1448	1.550242	0.275031	-0.013737	0.200604	-0.145063	0.306575	-0.075308	0.345552	-0.136589	-0.432798	-0.4560
1449	-0.957339	0.014308	0.303843	0.444350	-0.133690	0.381804	-0.326432	0.878047	-0.304441	-0.247443	-0.1936
1450	-0.185894	1.062662	0.220857	-0.467643	-0.187757	-0.192703	-0.091597	-0.036576	0.038255	0.056518	-0.0128
1451	-1.182064	-0.038570	0.561561	0.701126	0.047645	0.193687	-0.027335	-0.339374	0.022267	-0.002573	-0.1053
1452	1.550242	0.275031	-0.013737	0.200604	-0.145063	0.306575	-0.075308	0.345552	-0.136589	-0.432798	-0.4560

1453 rows × 11 columns

```
std_dev = []
for i in names:
    std_dev.append(np.std(pf[i]))

print("Standard Deviation")
np.array(std_dev)
```

```
Standard Deviation
array([0.75678896, 0.60724649, 0.50444578, 0.39866134, 0.33728888,
       0.31016782, 0.28959761, 0.27502727, 0.2651598 , 0.24875617,
       0.23682131])
```

```
pov = pca.explained_variance_ratio_
print("Proportion of Variance")
pov
```

```
Proportion of Variance
array([0.29944723, 0.19279721, 0.13304535, 0.08309578, 0.05948052,
       0.05029956, 0.0438491 , 0.03954779, 0.0367609 , 0.03235329,
       0.02932326])
```

```
cp = np.cumsum(pca.explained_variance_ratio_)
print("Cumulative Proportion")
cp
```

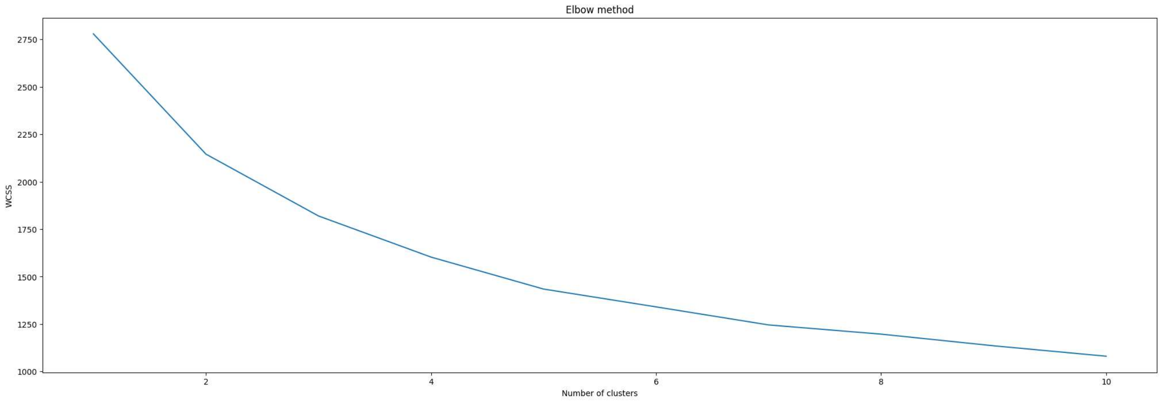
```
Cumulative Proportion
array([0.29944723, 0.49224445, 0.6252898 , 0.70838558, 0.7678661 ,
       0.81816566, 0.86201476, 0.90156255, 0.93832345, 0.97067674,
       1.        ])
```

```
components = pca.components_
num_pc = pca.n_features_
pc_name = ["PC"+str(i) for i in range(1, num_pc+1)]
loadings_df = pd.DataFrame
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/deprecation.py:101: FutureWarning: Attribute `n_features_` was deprecated in
  warnings.warn(msg, category=FutureWarning)
```

```
from sklearn.cluster import KMeans

wcss=[]
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(pf)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss)
plt.title("Elbow method")
plt.xlabel("Number of clusters")
plt.ylabel("WCSS")
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_in
  warnings.warn(
```



```
#K-means clustering

kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(x)
dataset['cluster_num'] = kmeans.labels_ #adding to df
print (kmeans.labels_) #Label assigned for each data point
print (kmeans.inertia_) #gives within-cluster sum of squares.
print(kmeans.n_iter_) #number of iterations that k-means algorithm runs to get a minimum within-cluster sum of squares
print(kmeans.cluster_centers_) #Location of the centroids on each cluster.
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
[2 0 0 ... 0 1 3]
1603.0604440558923
7
[[0.85448916 0.9628483  0.13312693 0.90712074 0.61919505 0.86068111
  0.10835913 0.93188854 0.89783282 0.20433437 0.10526316]
 [0.88793103 0.98103448 0.0862069  0.79482759 0.32931034 0.96034483
  0.92241379 0.97586207 0.01724138 0.32068966 0.04310345]
 [0.02302632 0.89144737 0.07236842 0.92434211 0.66776316 0.96381579
  0.93421053 0.15460526 0.01315789 0.07236842 0.38815789]
 [0.0203252  0.68292683 0.08536585 0.91463415 0.69512195 0.73170732
  0.06504065 0.08943089 0.87804878 0.06097561 0.71544715]]
```

```
from collections import Counter
Counter(kmeans.labels_)
```

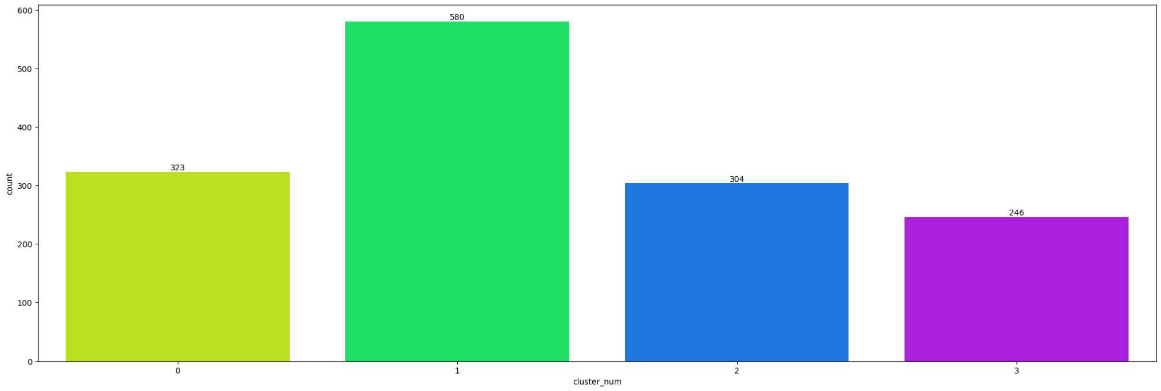
```
Counter({2: 304, 0: 323, 1: 580, 3: 246})
```

```
loadings = pca.components_
num_pc = pca.n_features_
pc_list = ["PC"+str(i) for i in list(range(1, num_pc+1))]
loadings_df = pd.DataFrame.from_dict(dict(zip(pc_list, loadings)))
loadings_df['variable'] = x.columns.values
loadings_df = loadings_df.set_index('variable')
loadings_df
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/deprecation.py:101: FutureWarning: Attribute `n_features_`
  warnings.warn(msg, category=FutureWarning)
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
variable										
yummy	-0.476933	0.363790	-0.304444	0.055162	-0.307535	0.170738	-0.280519	0.013041	0.572403	-0.110284
convenient	-0.155332	0.016414	-0.062515	-0.142425	0.277608	-0.347830	-0.059738	-0.113079	-0.018465	-0.665818
spicy	-0.006356	0.018809	-0.037019	0.197619	0.070620	-0.355087	0.707637	0.375934	0.400280	-0.075634
fattening	0.116232	-0.034094	-0.322359	-0.354139	-0.073405	-0.406515	-0.385943	0.589622	-0.160512	-0.005338
greasy	0.304443	-0.063839	-0.802373	0.253960	0.361399	0.209347	0.036170	-0.138241	-0.002847	0.008707
fast	-0.108493	-0.086972	-0.064642	-0.097363	0.107930	-0.594632	-0.086846	-0.627799	0.166197	0.239532
cheap	-0.337186	-0.610633	-0.149310	0.118958	-0.128973	-0.103241	-0.040449	0.140060	0.076069	0.428087
tasty	-0.471514	0.307318	-0.287265	-0.002547	-0.210899	-0.076914	0.360453	-0.072792	-0.639086	0.079184
expensive	0.329042	0.601286	0.024397	0.067816	-0.003125	-0.261342	-0.068385	0.029539	0.066996	0.454399
healthy	-0.213711	0.076593	0.192051	0.763488	0.287846	-0.178226	-0.349616	0.176303	-0.185572	-0.038117
disgusting	0.374753	-0.139656	-0.088571	0.369539	-0.729209	-0.210878	-0.026792	-0.167181	-0.072483	-0.289592

```
f = sns.countplot(x=dataset["cluster_num"], palette='hsv')
f.bar_label(f.containers[0])
plt.rcParams['figure.figsize'] = (25, 8)
```



```
dataset1=pd.get_dummies(dataset, prefix=['cluster_num'], columns=['cluster_num'])
```

```
dataset1
```

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age
0	0	1	0	1	0	1	1	0	1	0	0	-3	61

```
result=[]
for i in cols:
    Gender = dataset.groupby(['cluster_num'])[i].mean()
    Gender = Gender.to_frame().reset_index()
    result.append(Gender)
```

```
for i in range(len(result)):
    print(pd.DataFrame(result[i]))
```

```
   cluster_num     yummy
0            0  0.854489
1            1  0.887931
2            2  0.023026
3            3  0.020325
   cluster_num  convenient
0            0    0.962848
1            1    0.981034
2            2    0.891447
3            3    0.682927
   cluster_num     spicy
0            0  0.133127
1            1  0.086207
2            2  0.072368
3            3  0.085366
   cluster_num  fattening
0            0   0.907121
1            1   0.794828
2            2   0.924342
3            3   0.914634
   cluster_num    greasy
0            0  0.619195
1            1  0.329310
2            2  0.667763
3            3  0.695122
   cluster_num      fast
0            0  0.860681
1            1  0.960345
2            2  0.963816
3            3  0.731707
   cluster_num     cheap
0            0  0.108359
1            1  0.922414
2            2  0.934211
3            3  0.065041
   cluster_num     tasty
0            0  0.931889
1            1  0.975862
2            2  0.154605
3            3  0.089431
   cluster_num  expensive
0            0   0.897833
1            1   0.017241
2            2   0.013158
3            3   0.878049
   cluster_num   healthy
0            0  0.204334
1            1  0.320690
2            2  0.072368
3            3  0.060976
   cluster_num  disgusting
0            0    0.105263
1            1    0.043103
2            2    0.388158
3            3    0.715447
```

```
dataset1=pd.get_dummies(dataset1, prefix=['VisitFrequency'], columns=['VisitFrequency'])
```

```
dataset1
```

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	...	cluster_num_0	clu
0	0	1	0	1	0	1	1	0	1	0	...	0	
1	1	1	0	1	1	1	1	1	1	0	...	1	
2	0	1	1	1	1	1	0	1	1	1	...	1	
3	1	1	0	1	1	1	1	1	0	0	...	0	
4	0	1	0	1	1	1	1	0	0	1	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1448	0	1	0	1	1	0	0	0	1	0	...	0	
1449	1	1	0	1	0	0	1	1	0	1	...	0	

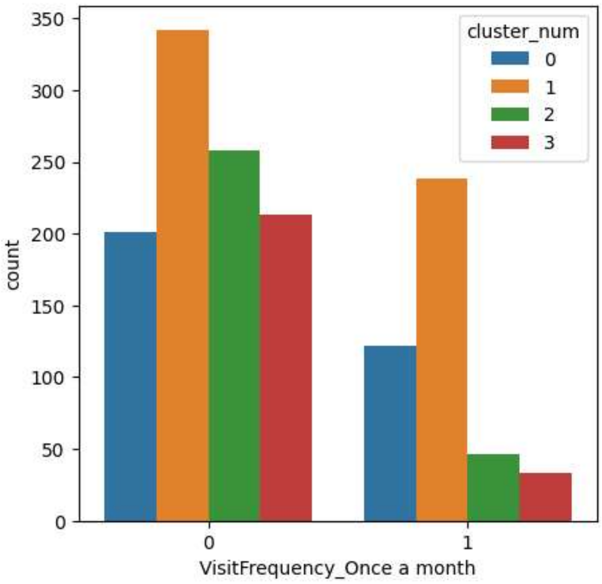
```
cols
```

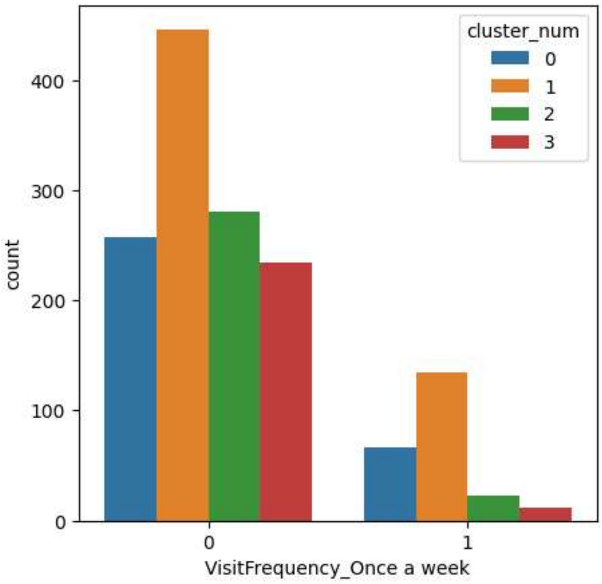
```
['yummy',
 'convenient',
 'spicy',
 'fattening',
 'greasy',
 'fast',
 'cheap',
 'tasty',
 'expensive',
 'healthy',
 'disgusting']
```

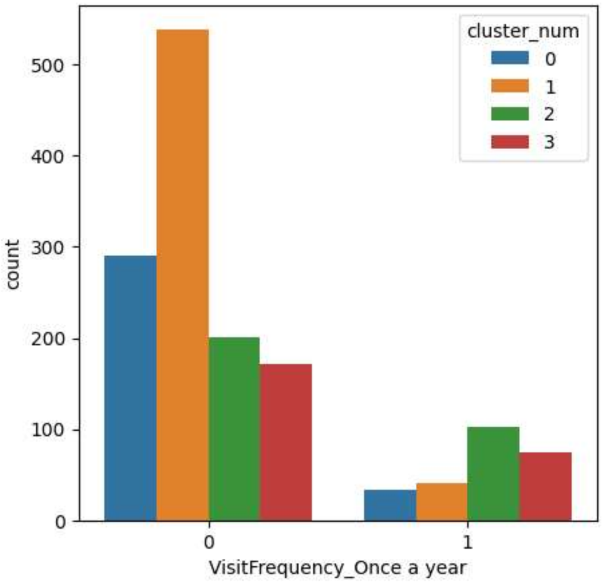
```
xx=dataset[cols]
```

```
for i in dataset1.columns:
    plt.rcParams['figure.figsize'] = (5, 5)
    plt.rcParams['font.size'] = 10
    sns.countplot(x=dataset1[i], hue=dataset['cluster_num'], data=dataset1)
    plt.show()
```

VisitFrequency_Never


count
VisitFrequency_Once a month
cluster_num


count
VisitFrequency_Once a week
cluster_num


count
VisitFrequency_Once a year
cluster_num