In [1]:

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

In [2]:

```
#Loading the dataset
df = pd.read_csv("../input/mcdonalds/mcdonalds.csv")
```

In [3]:

```
df.columns.tolist()
```

Out[3]:

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

In [4]:

```
df.shape
```

Out[4]:

```
(1453, 15)
```

In [5]:

```
df.head(6)
```

Out[5]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age	VisitFrequency
0	No	Yes	No	Yes	No	Yes	Yes	No	Yes	No	No	-3	61	Every three months
1	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	+2	51	Every three months
2	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	+1	62	Every three months
3	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	No	No	Yes	+4	69	Once a week
4	No	Yes	No	Yes	Yes	Yes	Yes	No	No	Yes	No	+2	49	Once a month
5	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No	No	No	+2	55	Every three months

In [6]:

```
df.dtypes
```

Out[6]:

```
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
```

In [7]:

```
df['yummy'].value_counts()
```

Out[7]:

```
Yes    803
No     650
Name: yummy, dtype: int64
```

In [8]:

```
df['VisitFrequency'].value_counts()
```

Out[8]:

```
Once a month             439
Every three months       342
Once a year              252
Once a week              235
Never                    131
More than once a week     54
Name: VisitFrequency, dtype: int64
```

In [9]:

```
df['Like'].value_counts()
```

Out[9]:

```
+3              229
+2              187
0               169
+4              160
+1              152
I hate it!-5    152
I love it!+5    143
-3               73
-4               71
-2               59
-1               58
Name: Like, dtype: int64
```

In [10]:

```
df['convenient'].value_counts()
```

Out[10]:

```
Yes    1319
No      134
Name: convenient, dtype: int64
```

In [11]:

```
df['Age'].value_counts()
```

Out[11]:

```
55    53
60    38
37    37
59    36
57    36
52    36
58    35
36    35
49    34
62    34
50    34
32    33
44    32
56    32
64    32
53    31
26    31
24    30
35    30
51    30
47    30
42    30
23    30
39    29
29    28
34    28
30    28
38    27
40    27
31    27
25    26
33    26
61    26
67    26
48    26
43    25
27    25
63    25
54    24
41    23
22    23
65    23
45    22
20    21
46    19
28    18
66    17
21    16
18    16
70    15
69    14
68    13
19    10
71     1
Name: Age, dtype: int64
```

In [12]:

```
MD_x = df.iloc[:, 0:11].values
MD_x = (MD_x == "Yes").astype(int)
col_means = np.round(np.mean(MD_x, axis=0), 2)

print(col_means)
```

```
[0.55 0.91 0.09 0.87 0.53 0.9  0.6  0.64 0.36 0.2  0.24]
```

In [13]:

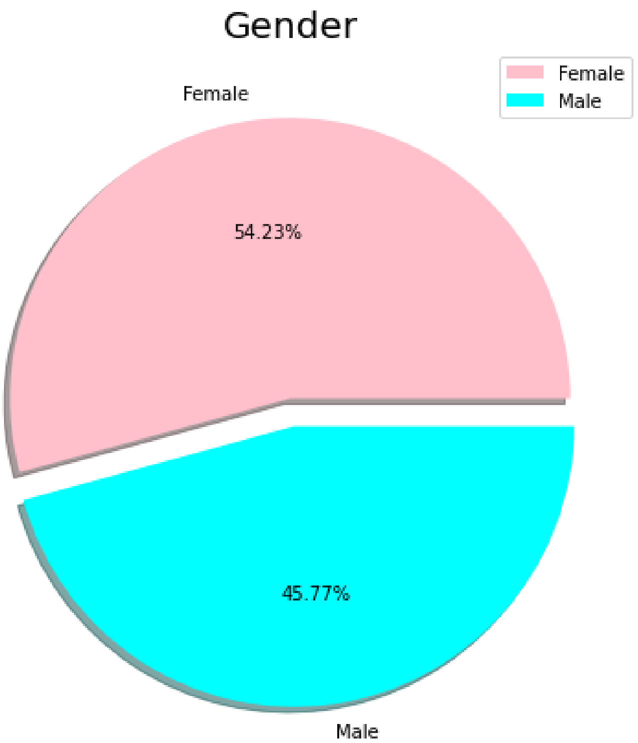
```
#EXPLORING DATA
```

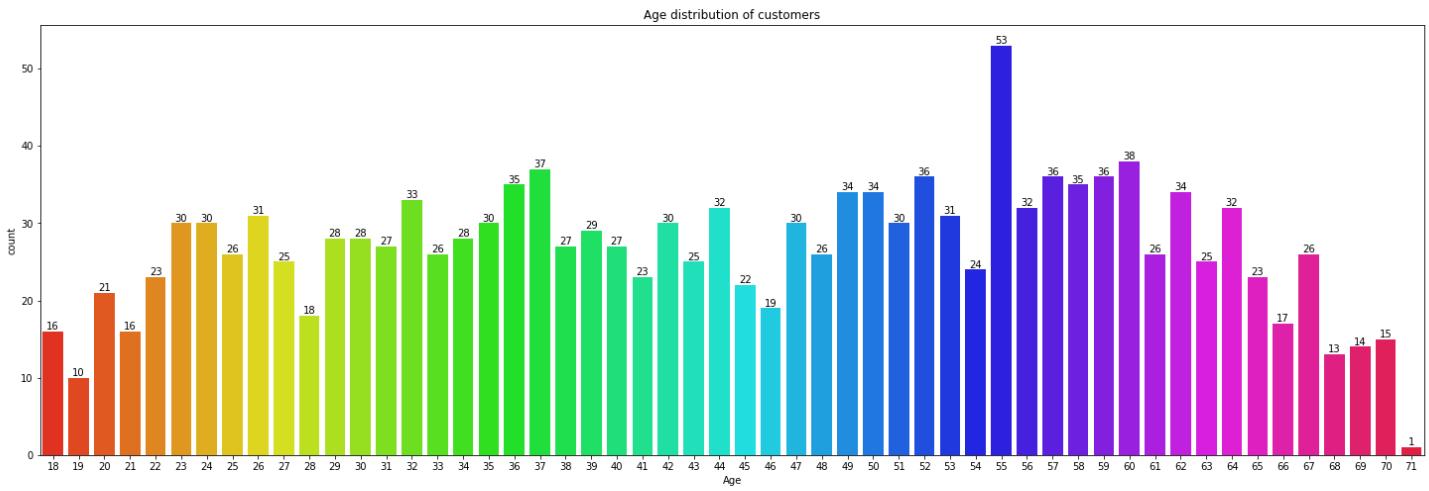
```
#EXPLORING DATA

#Customer segmentation - based on socio-demographs (Age & Gender)

#Gender
labels = ['Female', 'Male']
size = df['Gender'].value_counts()
colors = ['pink', 'cyan']
explode = [0, 0.1]
plt.rcParams['figure.figsize'] = (7, 7)
plt.pie(size, colors = colors, explode = explode, labels = labels, shadow = True, autopc
t = '%.2f%%')
plt.title('Gender', fontsize = 20)
plt.axis('off')
plt.legend()
plt.show()
#we infer that there are more female customers than male.

#Age
plt.rcParams['figure.figsize'] = (25, 8)
f = sns.countplot(x=df['Age'],palette = 'hsv')
f.bar_label(f.containers[0])
plt.title('Age distribution of customers')
plt.show()
# Mcdonalds recieve more customers of age between 50-60 and 35-40.
```

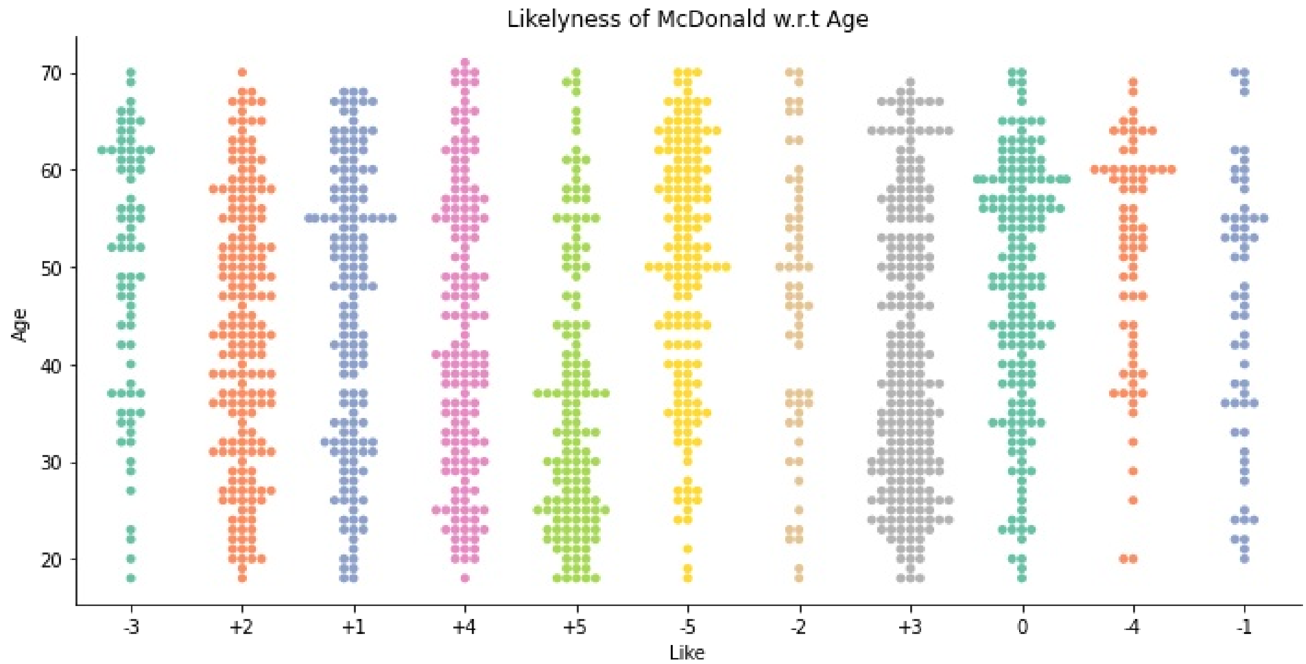




In [14]:

```
#Customer segmentation - based on pyschographic segmentation
```

```python
#For convinence renaming the category
df['Like']= df['Like'].replace({'I hate it!-5': '-5','I love it!+5':'+5'})
#Like
sns.catplot(x="Like", y="Age",data=df,
            orient="v", height=5, aspect=2, palette="Set2",kind="swarm")
plt.title('Likelyness of McDonald w.r.t Age')
plt.show()
```



In [15]:

```python
#Label encoding for categorical - Converting 11 cols with yes/no

from sklearn.preprocessing import LabelEncoder
def labelling(x):
    df[x] = LabelEncoder().fit_transform(df[x])
    return df

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

for i in cat:
    labelling(i)
df
```

Out[15]:

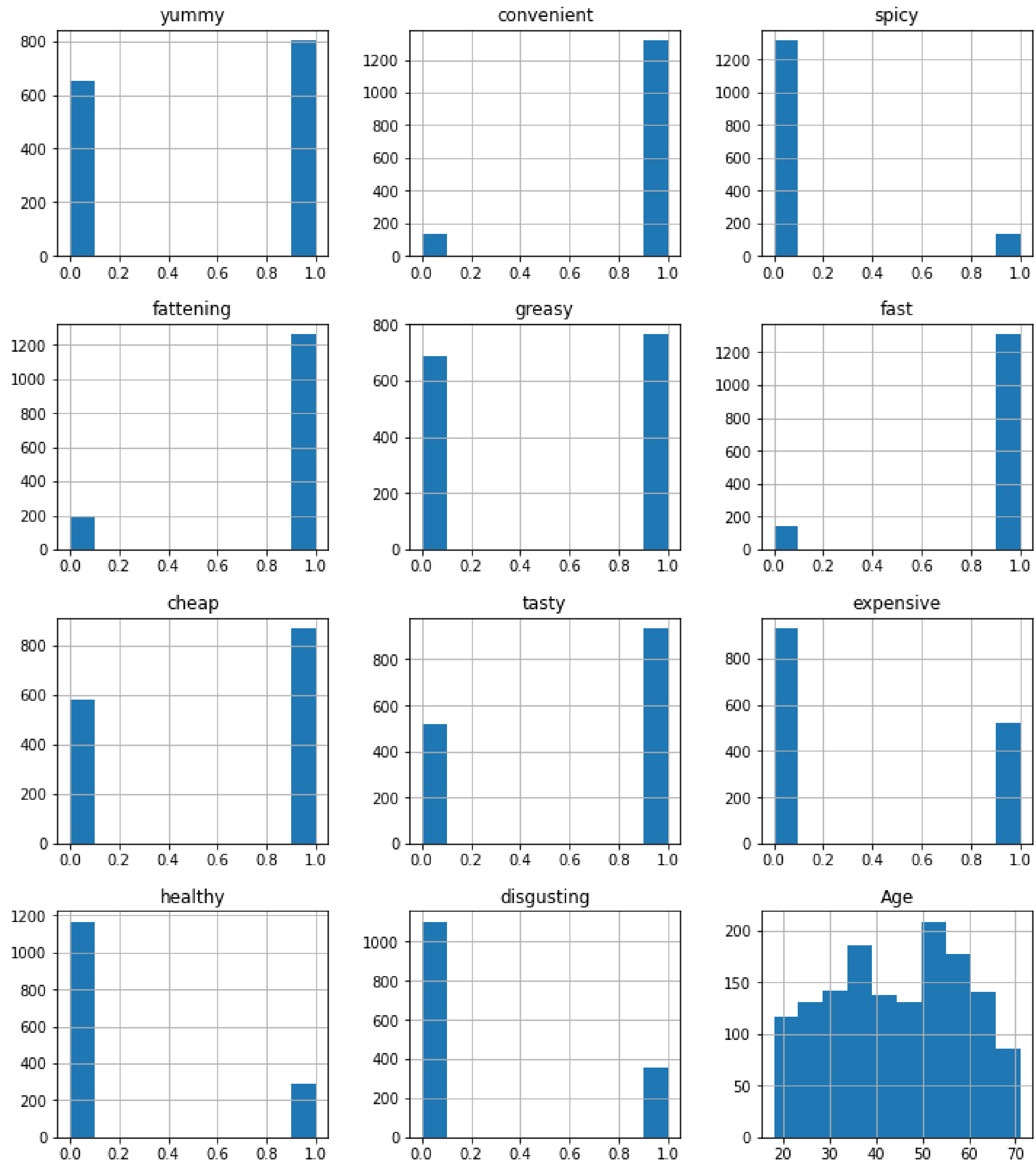
	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age	VisitFrequer
0	0	1	0	1	0	1	1	0	1	0	0	-3	61	Every th mon
1	1	1	0	1	1	1	1	1	1	0	0	+2	51	Every th mon
2	0	1	1	1	1	1	0	1	1	1	0	+1	62	Every th mon
3	1	1	0	1	1	1	1	1	0	0	1	+4	69	Once a we
4	0	1	0	1	1	1	1	0	0	1	0	+2	49	Once a mo
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1448	0	1	0	1	1	0	0	0	1	0	1	-5	47	Once a y
1449	1	1	0	1	0	0	1	1	0	1	0	+2	36	Once a we
1450	1	1	0	1	0	1	0	1	1	0	0	+3	52	Once a mo
1451	1	1	0	0	0	1	1	1	0	1	0	+4	41	Every th mon
1452	0	1	0	1	1	0	0	0	1	0	1	-3	30	Every th

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting	Like	Age	VisitFrequer mon

1453 rows × 15 columns

In [16]:

```
#Histogram of the each attributes
plt.rcParams['figure.figsize'] = (12,14)
df.hist()
plt.show()
```



In [17]:

```
#Considering only first 11 attributes
df_eleven = df.loc[:,cat]
df_eleven
```

Out[17]:

	yummy	convenient	spicy	fattening	greasy	fast	cheap	tasty	expensive	healthy	disgusting

	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

In [18]:

```
#Considering only the 11 cols and converting it into array
x = df.loc[:,cat].values
x
```

Out[18]:

```
array([[0, 1, 0, ..., 1, 0, 0],
       [1, 1, 0, ..., 1, 0, 0],
       [0, 1, 1, ..., 1, 1, 0],
       ...,
       [1, 1, 0, ..., 1, 0, 0],
       [1, 1, 0, ..., 0, 1, 0],
       [0, 1, 0, ..., 1, 0, 1]])
```

PRINCIPAL COMPONENT ANALYSIS

In [19]:

```
#Principal component analysis

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
```

Out[19]:

	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	pc9	pc10	pc11
0	0.425367	-0.219079	0.663255	-0.401300	0.201705	-0.389767	-0.211982	0.163235	0.181007	0.515706	-0.567074
1	-0.218638	0.388190	-0.730827	-0.094724	0.044669	-0.086596	-0.095877	-0.034756	0.111476	0.493313	-0.500440
2	0.375415	0.730435	-0.122040	0.692262	0.839643	-0.687406	0.583112	0.364379	-0.322288	0.061759	0.242741
3	-0.172926	-0.352752	-0.843795	0.206998	-0.681415	-0.036133	-0.054284	-0.231477	-0.028003	-0.250678	-0.051034
4	0.187057	-0.807610	0.028537	0.548332	0.854074	-0.097305	-0.457043	0.171758	-0.074409	0.031897	0.082245
...	...	...	...	...	...	...	...	...	...	...	...
1448	1.550242	0.275031	-0.013737	0.200604	-0.145063	0.306575	-0.075308	0.345552	-0.136589	-0.432798	-0.456076
1449	-0.957339	0.014308	0.303843	0.444350	-0.133690	0.381804	-0.326432	0.878047	-0.304441	-0.247443	-0.193671
1450	-0.185894	1.062662	0.220857	-0.467643	-0.187757	-0.192703	-0.091597	-0.036576	0.038255	0.056518	-0.012800

	pc1	pc2	pc3	pc4	pc5	pc6	pc7	pc8	pc9	pc10	pc11
1451	-1.182064	-0.038570	0.561561	0.701126	0.047645	0.193687	-0.027335	-0.339374	0.022267	-0.002573	-0.105316
1452	1.550242	0.275031	-0.013737	0.200604	-0.145063	0.306575	-0.075308	0.345552	-0.136589	-0.432798	-0.456076

1453 rows × 11 columns

In [20]:

```
#Proportion of Variance (from PC1 to PC11)
pca.explained_variance_ratio_
```

Out[20]:

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

In [21]:

```
np.cumsum(pca.explained_variance_ratio_)
```

Out[21]:

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

In [22]:

```
pca = PCA()
pca.fit(df_eleven)

# Get loadings and number of principal components
loadings = pca.components_
num_pc = pca.n_features_
pc_list = ["PC" + str(i) for i in range(1, num_pc+1)]
loadings_df = pd.DataFrame(loadings.T, columns=pc_list)
loadings_df['variable'] = df_eleven.columns.values
loadings_df = loadings_df.set_index('variable')
loadings_df
```

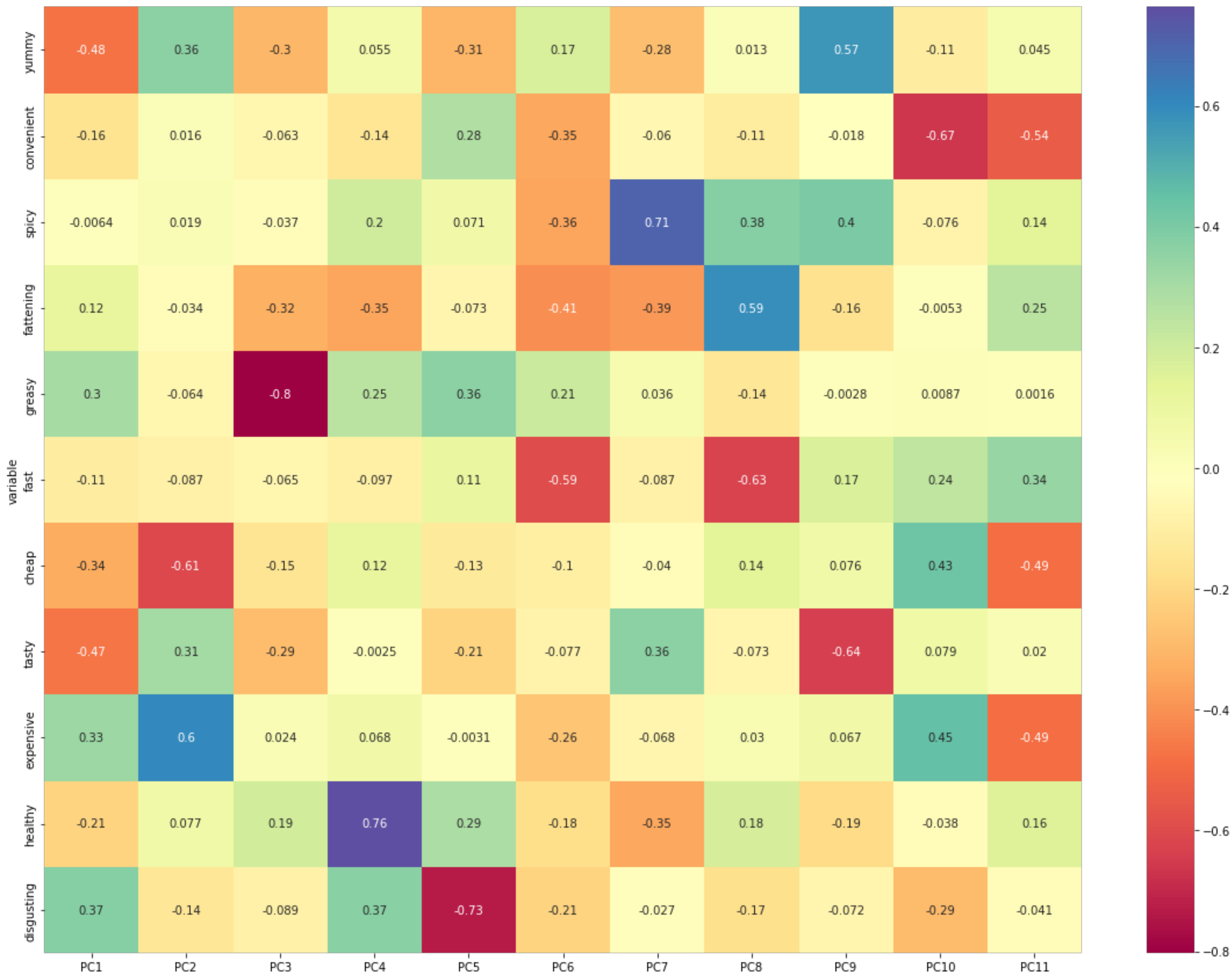
Out[22]:

variable	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
yummy	-0.476933	0.363790	-0.304444	0.055162	-0.307535	0.170738	-0.280519	0.013041	0.572403	-0.110284	0.045439
convenient	-0.155332	0.016414	-0.062515	-0.142425	0.277608	-0.347830	-0.059738	-0.113079	-0.018465	-0.665818	-0.541616
spicy	-0.006356	0.018809	-0.037019	0.197619	0.070620	-0.355087	0.707637	0.375934	0.400280	-0.075634	0.141730
fattening	0.116232	-0.034094	-0.322359	-0.354139	-0.073405	-0.406515	-0.385943	0.589622	-0.160512	-0.005338	0.250910
greasy	0.304443	-0.063839	-0.802373	0.253960	0.361399	0.209347	0.036170	-0.138241	-0.002847	0.008707	0.001642
fast	-0.108493	-0.086972	-0.064642	-0.097363	0.107930	-0.594632	-0.086846	-0.627799	0.166197	0.239532	0.339265
cheap	-0.337186	-0.610633	-0.149310	0.118958	-0.128973	-0.103241	-0.040449	0.140060	0.076069	0.428087	-0.489283
tasty	-0.471514	0.307318	-0.287265	-0.002547	-0.210899	-0.076914	0.360453	-0.072792	-0.639086	0.079184	0.019552
expensive	0.329042	0.601286	0.024397	0.067816	-0.003125	-0.261342	-0.068385	0.029539	0.066996	0.454399	-0.490069
healthy	-0.213711	0.076593	0.192051	0.763488	0.287846	-0.178226	-0.349616	0.176303	-0.185572	-0.038117	0.157608
disgusting	0.374753	-	-	0.369539	-	-	-	-	-	-	-

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10	PC11
disgusting	0.374753	0.139656	0.088571	0.369539	0.729209	0.210878	0.026792	0.167181	0.072483	0.289592	0.040662

In [23]:

```
#Correlation matrix plot for loadings
plt.rcParams['figure.figsize'] = (20,15)
ax = sns.heatmap(loadings_df, annot=True, cmap='Spectral')
plt.show()
```

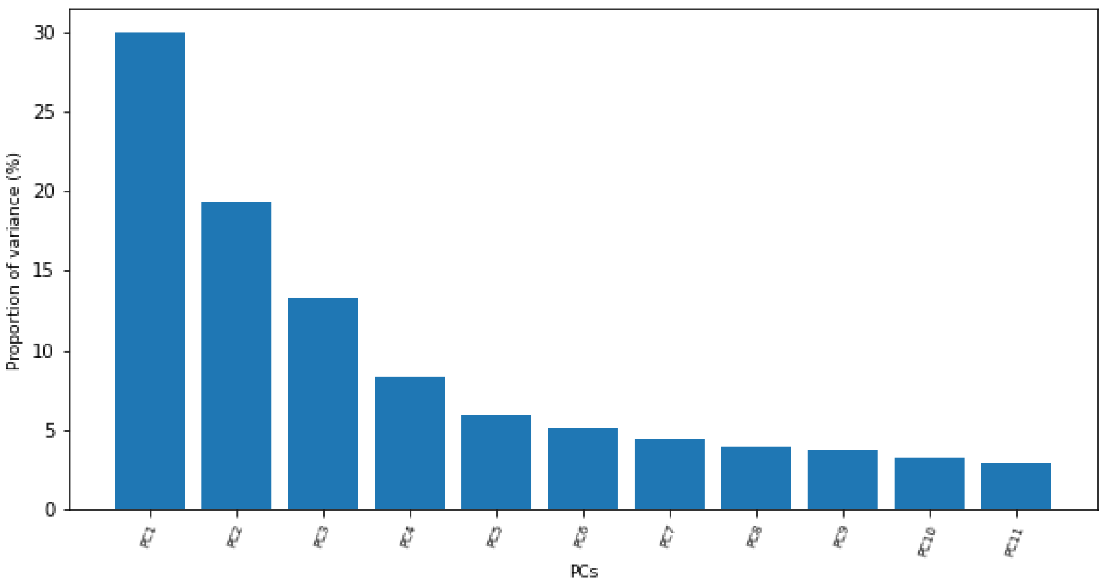


In [24]:

```
!pip install bioinfokit
#Scree plot (Elbow test)- PCA
from bioinfokit.visuz import cluster
cluster.screeplot(obj=[pc_list, pca.explained_variance_ratio_],show=True,dim=(10,5))
```

```
Collecting bioinfokit
  Downloading bioinfokit-2.1.3.tar.gz (87 kB)
     |████████████████████████████████| 87 kB 3.1 MB/s
  Preparing metadata (setup.py) ... done
Requirement already satisfied: pandas in /opt/conda/lib/python3.7/site-packages (from bio
infokit) (1.3.4)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from bioi
nfokit) (1.19.5)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-packages (from
bioinfokit) (3.5.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from bioi
nfokit) (1.7.3)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.7/site-packages (fr
om bioinfokit) (0.23.2)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.7/site-packages (from bi
oinfokit) (0.11.2)
Requirement already satisfied: matplotlib-venn in /opt/conda/lib/python3.7/site-packages
(from bioinfokit) (0.11.6)
```

```
Requirement already satisfied: tabulate in /opt/conda/lib/python3.7/site-packages (from b
ioinfokit) (0.8.9)
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.7/site-packages (fro
m bioinfokit) (0.12.2)
Requirement already satisfied: textwrap3 in /opt/conda/lib/python3.7/site-packages (from
bioinfokit) (0.9.2)
Collecting adjustText
  Downloading adjustText-0.8-py3-none-any.whl (9.1 kB)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.7/site-pack
ages (from matplotlib->bioinfokit) (2.8.0)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (fr
om matplotlib->bioinfokit) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (f
rom matplotlib->bioinfokit) (8.2.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packages
(from matplotlib->bioinfokit) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/site-package
s (from matplotlib->bioinfokit) (4.28.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-package
s (from matplotlib->bioinfokit) (1.3.2)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/site-packages
(from matplotlib->bioinfokit) (3.0.6)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (fr
om pandas->bioinfokit) (2021.3)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-packages (fr
om scikit-learn->bioinfokit) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site-pack
ages (from scikit-learn->bioinfokit) (3.0.0)
Requirement already satisfied: patsy>=0.5 in /opt/conda/lib/python3.7/site-packages (from
statsmodels->bioinfokit) (0.5.2)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (from patsy>
=0.5->statsmodels->bioinfokit) (1.16.0)
Building wheels for collected packages: bioinfokit
  Building wheel for bioinfokit (setup.py) ... done
  Created wheel for bioinfokit: filename=bioinfokit-2.1.3-py3-none-any.whl size=59071 sha
256=998128878ea450c154b335d45dd0d55a6396cf987a62f93151c2ba4f7d42b9a7
  Stored in directory: /root/.cache/pip/wheels/4e/24/52/f964c59a354a5cb0fc1f51cf7c13c74cd
8450b5b9f78833d7c
Successfully built bioinfokit
Installing collected packages: adjustText, bioinfokit
Successfully installed adjustText-0.8 bioinfokit-2.1.3
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting
behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
```

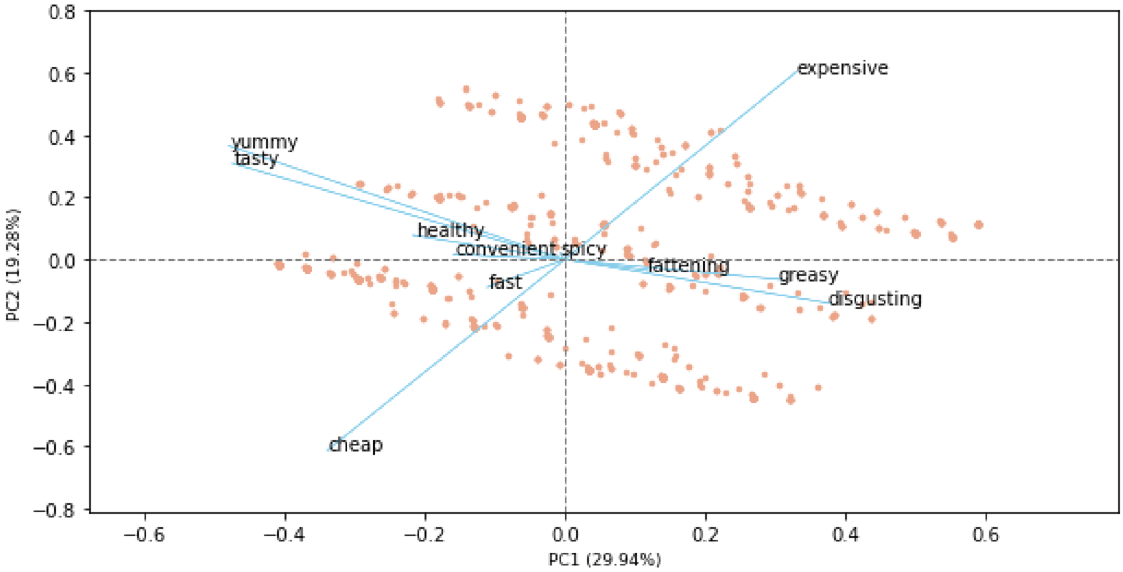


In [25]:

```
# get PC scores
pca_scores = PCA().fit_transform(x)

# get 2D biplot
```

```
cluster.biplot(cscore=pca_scores, loadings=loadings, labels=df.columns.values, var1=roun
d(pca.explained_variance_ratio_[0]*100, 2),
    var2=round(pca.explained_variance_ratio_[1]*100, 2),show=True,dim=(10,5))
```

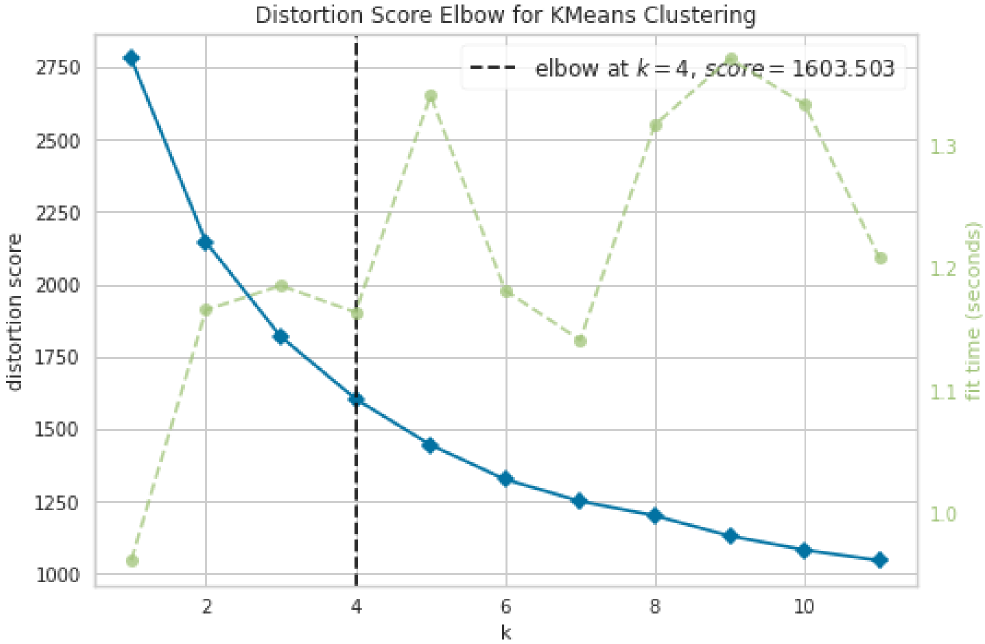


EXTRACTING SEGMENTS

In [26]:

```
#Extracting segments

#Using k-means clustering analysis
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,12)).fit(df_eleven)
visualizer.show()
```



Out[26]:

```
<AxesSubplot:title={'center':'Distortion Score Elbow for KMeans Clustering'}, xlabel='k',
ylabel='distortion score'>
```

In [27]:

```
#K-means clustering

kmeans = KMeans(n_clusters=4, init='k-means++', random_state=0).fit(df_eleven)
df['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.
```

```
[3 0 0 ... 0 1 2]
1603.5029251924634
9
[[0.87619048 0.96825397 0.13650794 0.90793651 0.60952381 0.85714286
  0.11111111 0.93015873 0.91428571 0.20952381 0.10793651]
 [0.88793103 0.98103448 0.0862069  0.79482759 0.32931034 0.96034483
  0.92241379 0.97586207 0.01724138 0.32068966 0.04310345]
 [0.0173913  0.64782609 0.07826087 0.90869565 0.70434783 0.73043478
  0.07391304 0.09565217 0.95217391 0.06521739 0.7        ]
 [0.02439024 0.90243902 0.07621951 0.92682927 0.67073171 0.95426829
  0.86280488 0.16768293 0.00914634 0.06707317 0.4054878 ]]
```

In [28]:

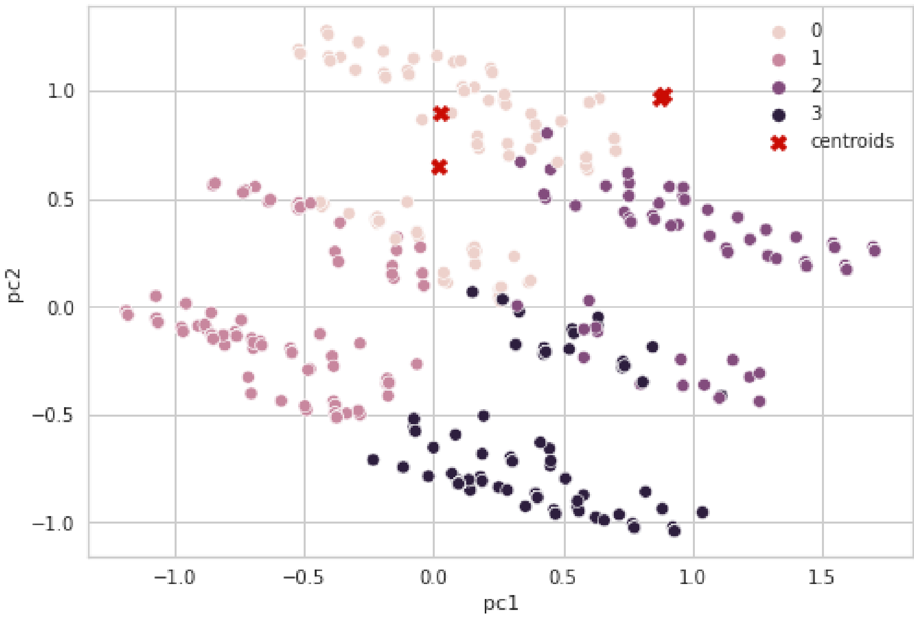
```
#To see each cluster size
from collections import Counter
Counter(kmeans.labels_)
```

Out[28]:

```
Counter({3: 328, 0: 315, 1: 580, 2: 230})
```

In [29]:

```
#Visulazing clusters
sns.scatterplot(data=pf, x="pc1", y="pc2", hue=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1],
            marker="X", c="r", s=80, label="centroids")
plt.legend()
plt.show()
```



DESCRIBING SEGMENTS

In [30]:

```
#DESCRIBING SEGMENTS

from statsmodels.graphics.mosaicplot import mosaic
from itertools import product

crosstab =pd.crosstab(df['cluster_num'],df['Like'])
#Reordering cols
crosstab = crosstab[['-5','-4','-3','-2','-1','0','+1','+2','+3','+4','+5']]
crosstab
```

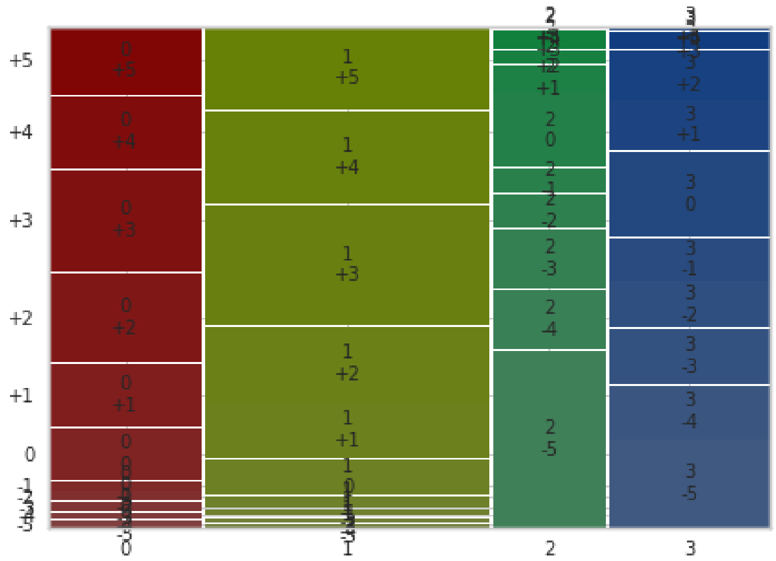
Out[30]:

Out[30]:

Like	-5	-4	-3	-2	-1	0	+1	+2	+3	+4	+5
cluster_num											
0	5	3	6	6	6	33	41	58	66	47	44
1	4	4	2	6	13	43	65	90	143	111	99
2	84	28	28	16	11	35	13	6	9	0	0
3	59	36	37	31	28	58	33	33	11	2	0

In [31]:

```
#MOSAIC PLOT
plt.rcParams['figure.figsize'] = (7,5)
mosaic(crosstab.stack())
plt.show()
```



In [32]:

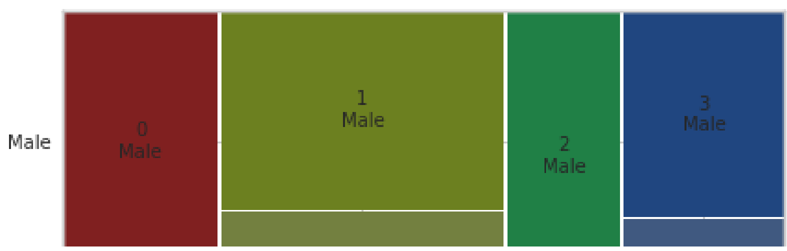
```
#Mosaic plot gender vs segment
crosstab_gender =pd.crosstab(df['cluster_num'],df['Gender'])
crosstab_gender
```

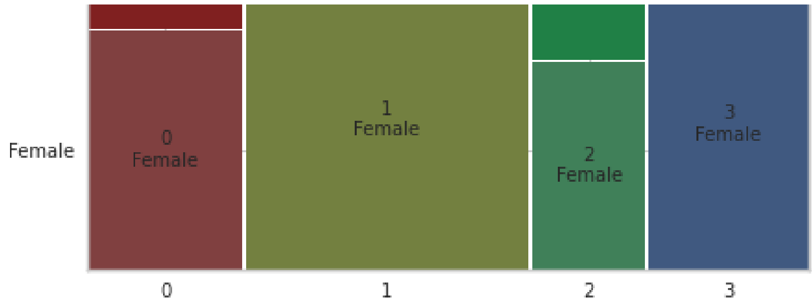
Out[32]:

Gender	Female	Male
cluster_num		
0	151	164
1	349	231
2	96	134
3	192	136

In [33]:

```
plt.rcParams['figure.figsize'] = (7,5)
mosaic(crosstab_gender.stack())
plt.show()
```





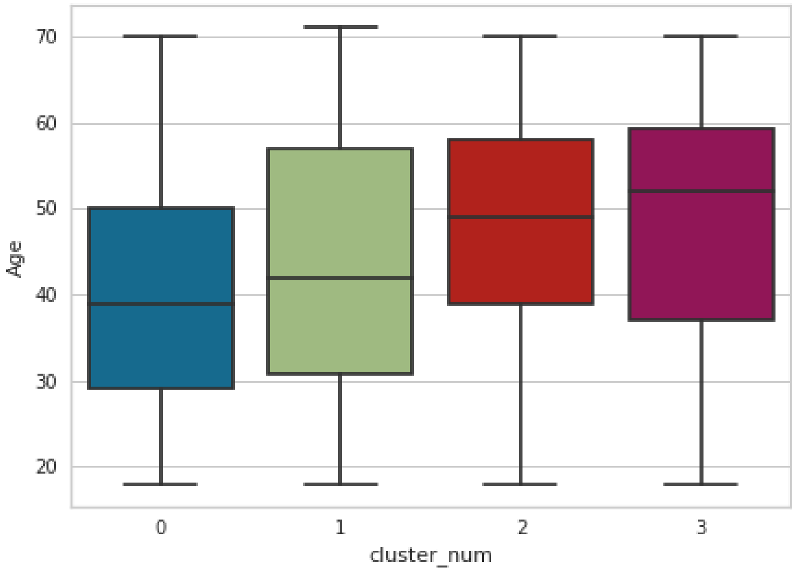
In [34]:

```
#box plot for age

sns.boxplot(x="cluster_num", y="Age", data=df)
```

Out[34]:

```
<AxesSubplot:xlabel='cluster_num', ylabel='Age'>
```



Selecting target segment

In [35]:

```
#Calculating the mean
#Visit frequency
df['VisitFrequency'] = LabelEncoder().fit_transform(df['VisitFrequency'])
visit = df.groupby('cluster_num')['VisitFrequency'].mean()
visit = visit.to_frame().reset_index()
visit
```

Out[35]:

	cluster_num	VisitFrequency
0	0	2.542857
1	1	2.584483
2	2	2.686957
3	3	2.789634

In [36]:

```
#Like
df['Like'] = LabelEncoder().fit_transform(df['Like'])
Like = df.groupby('cluster_num')['Like'].mean()
Like = Like.to_frame().reset_index()
Like
```

Out[36]:

	cluster_num	Like
0	0	3.219048
1	1	2.962069
2	2	7.395652
3	3	6.234756

In [37]:

```
#Gender
df['Gender'] = LabelEncoder().fit_transform(df['Gender'])
Gender = df.groupby('cluster_num')['Gender'].mean()
Gender = Gender.to_frame().reset_index()
Gender
```

Out[37]:

	cluster_num	Gender
0	0	0.520635
1	1	0.398276
2	2	0.582609
3	3	0.414634

In [38]:

```
segment = Gender.merge(Like, on='cluster_num', how='left').merge(visit, on='cluster_num'
, how='left')
segment
```

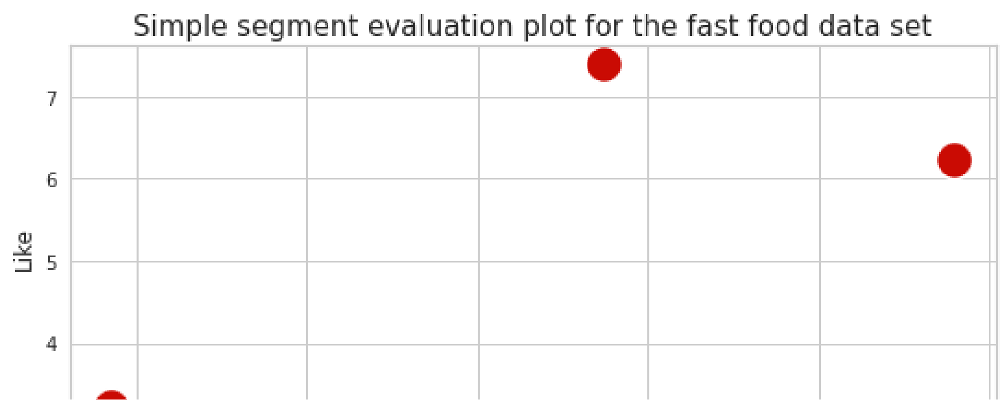
Out[38]:

	cluster_num	Gender	Like	VisitFrequency
0	0	0.520635	3.219048	2.542857
1	1	0.398276	2.962069	2.584483
2	2	0.582609	7.395652	2.686957
3	3	0.414634	6.234756	2.789634

In [39]:

```
#Target segments

plt.figure(figsize = (9,4))
sns.scatterplot(x = "VisitFrequency", y = "Like",data=segment,s=400, color="r")
plt.title("Simple segment evaluation plot for the fast food data set",
          fontsize = 15)
plt.xlabel("Visit", fontsize = 12)
plt.ylabel("Like", fontsize = 12)
plt.show()
```



In [ ]: