# Reshma Ankush Bhondave.

# Task-01

# Mcdonalds Market Segmentation project code # feyan labs internship (segaments B)

In [139]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
!pip install bioinfokit
!pip install yellowbrick
```

```
Requirement already satisfied: bioinfokit in c:\users\rakesh\anaconda3\lib\si
te-packages (2.1.3)
Requirement already satisfied: textwrap3 in c:\users\rakesh\anaconda3\lib\sit
e-packages (from bioinfokit) (0.9.2)
Requirement already satisfied: matplotlib-venn in c:\users\rakesh\anaconda3\l
ib\site-packages (from bioinfokit) (0.11.9)
Requirement already satisfied: pandas in c:\users\rakesh\anaconda3\lib\site-p
ackages (from bioinfokit) (1.4.2)
Requirement already satisfied: scipy in c:\users\rakesh\anaconda3\lib\site-pa
ckages (from bioinfokit) (1.7.3)
Requirement already satisfied: seaborn in c:\users\rakesh\anaconda3\lib\site-
packages (from bioinfokit) (0.11.2)
Requirement already satisfied: numpy in c:\users\rakesh\anaconda3\lib\site-pa
ckages (from bioinfokit) (1.21.5)
Requirement already satisfied: statsmodels in c:\users\rakesh\anaconda3\lib\s
ite-packages (from bioinfokit) (0.13.2)
Requirement already satisfied: adjustText in c:\users\rakesh\anaconda3\lib\si
te-packages (from bioinfokit) (0.8)
Requirement already satisfied: tabulate in c:\users\rakesh\anaconda3\lib\site
-packages (from bioinfokit) (0.8.9)
Requirement already satisfied: scikit-learn in c:\users\rakesh\anaconda3\lib
\site-packages (from bioinfokit) (1.0.2)
Requirement already satisfied: matplotlib in c:\users\rakesh\anaconda3\lib\si
te-packages (from bioinfokit) (3.5.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\rakesh\anaconda3
\lib\site-packages (from matplotlib->bioinfokit) (4.25.0)
Requirement already satisfied: cycler>=0.10 in c:\users\rakesh\anaconda3\lib
\site-packages (from matplotlib->bioinfokit) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\rakesh\anaconda3
\lib\site-packages (from matplotlib->bioinfokit) (1.3.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\rakesh\anaconda3\lib
\site-packages (from matplotlib->bioinfokit) (9.0.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\rakesh\anaconda3
\lib\site-packages (from matplotlib->bioinfokit) (3.0.4)
Requirement already satisfied: packaging>=20.0 in c:\users\rakesh\anaconda3\l
ib\site-packages (from matplotlib->bioinfokit) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\rakesh\anacon
da3\lib\site-packages (from matplotlib->bioinfokit) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\rakesh\anaconda3\lib\site
-packages (from python-dateutil>=2.7->matplotlib->bioinfokit) (1.16.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\rakesh\anaconda3\lib
\site-packages (from pandas->bioinfokit) (2021.3)
Requirement already satisfied: joblib>=0.11 in c:\users\rakesh\anaconda3\lib
\site-packages (from scikit-learn->bioinfokit) (1.1.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\rakesh\anacon
da3\lib\site-packages (from scikit-learn->bioinfokit) (2.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\rakesh\anaconda3\lib
\site-packages (from statsmodels->bioinfokit) (0.5.2)
```

```
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, sta
tus=None)) after connection broken by 'NewConnectionError('<pip._vendor.urlli
b3.connection.HTTPSConnection object at 0x000001E4D3986C40>: Failed to establ
ish a new connection: [Errno 11001] getaddrinfo failed')': /simple/yellowbric
k/
WARNING: Retrying (Retry(total=3, connect=None, read=None, redirect=None, sta
tus=None)) after connection broken by 'NewConnectionError('<pip._vendor.urlli
b3.connection.HTTPSConnection object at 0x000001E4D398C700>: Failed to establ
ish a new connection: [Errno 11001] getaddrinfo failed')': /simple/yellowbric
k/
WARNING: Retrying (Retry(total=2, connect=None, read=None, redirect=None, sta
tus=None)) after connection broken by 'NewConnectionError('<pip._vendor.urlli
b3.connection.HTTPSConnection object at 0x000001E4D398C100>: Failed to establ
ish a new connection: [Errno 11001] getaddrinfo failed')': /simple/yellowbric
k/
WARNING: Retrying (Retry(total=1, connect=None, read=None, redirect=None, sta
tus=None)) after connection broken by 'NewConnectionError('<pip._vendor.urlli
b3.connection.HTTPSConnection object at 0x000001E4D398C940>: Failed to establ
ish a new connection: [Errno 11001] getaddrinfo failed')': /simple/yellowbric
k/
WARNING: Retrying (Retry(total=0, connect=None, read=None, redirect=None, sta
tus=None)) after connection broken by 'NewConnectionError('<pip._vendor.urlli
b3.connection.HTTPSConnection object at 0x000001E4D398CAF0>: Failed to establ
ish a new connection: [Errno 11001] getaddrinfo failed')': /simple/yellowbric
k/
ERROR: Could not find a version that satisfies the requirement yellowbrick (f
rom versions: none)
ERROR: No matching distribution found for yellowbrick
```

In [140]:
```
# read data
df=pd.read_csv("C://Users/Rakesh/Downloads/mcdonalds.csv")
```

In [141]:
```
df
```

Out[141]:

| | yummy | convenient | spicy | fattening | greasy | fast | cheap | tasty | expensive | healthy | disgu |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | No | Yes | No | Yes | No | Yes | Yes | No | Yes | No | |
| **1** | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | No | |
| **2** | No | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | |
| **3** | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | No | No | |
| **4** | No | Yes | No | Yes | Yes | Yes | Yes | No | No | Yes | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1448** | No | Yes | No | Yes | Yes | No | No | No | Yes | No | |
| **1449** | Yes | Yes | No | Yes | No | No | Yes | Yes | No | Yes | |
| **1450** | Yes | Yes | No | Yes | No | Yes | No | Yes | Yes | No | |
| **1451** | Yes | Yes | No | No | No | Yes | Yes | Yes | No | Yes | |
| **1452** | No | Yes | No | Yes | Yes | No | No | No | Yes | No | |

1453 rows × 15 columns

In [142]:
```
df.head()
```

Out[142]:

| | yummy | convenient | spicy | fattening | greasy | fast | cheap | tasty | expensive | healthy | disgusti |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | No | Yes | No | Yes | No | Yes | Yes | No | Yes | No | I |
| **1** | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | Yes | No | I |
| **2** | No | Yes | Yes | Yes | Yes | Yes | No | Yes | Yes | Yes | I |
| **3** | Yes | Yes | No | Yes | Yes | Yes | Yes | Yes | No | No | Y |
| **4** | No | Yes | No | Yes | Yes | Yes | Yes | No | No | Yes | I |

In [143]:
```
df.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
```

In [144]:
```
print(pd.isnull(df).sum())
```

```
yummy             0
convenient        0
spicy             0
fattening         0
greasy            0
fast              0
cheap             0
tasty             0
expensive         0
healthy           0
disgusting        0
Like              0
Age               0
VisitFrequency    0
Gender            0
dtype: int64
```

In [145]:
```
df.describe()
```

Out[145]:

| | Age |
|---|---|
| **count** | 1453.000000 |
| **mean** | 44.604955 |
| **std** | 14.221178 |
| **min** | 18.000000 |
| **25%** | 33.000000 |
| **50%** | 45.000000 |
| **75%** | 57.000000 |
| **max** | 71.000000 |

In [146]:
```
df.dtypes
```

Out[146]:
```
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 [147]:
```
df.shape
```

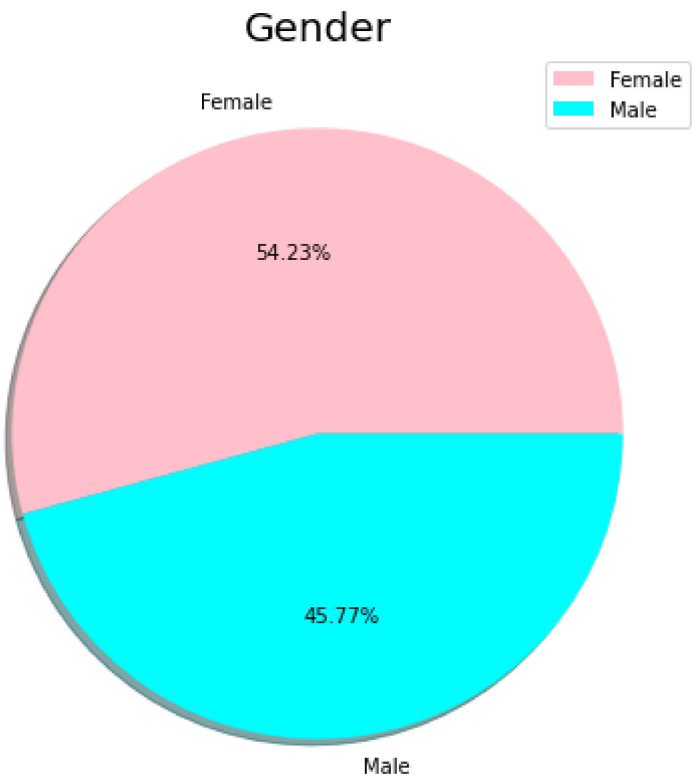
Out[147]:
```
(1453, 15)
```

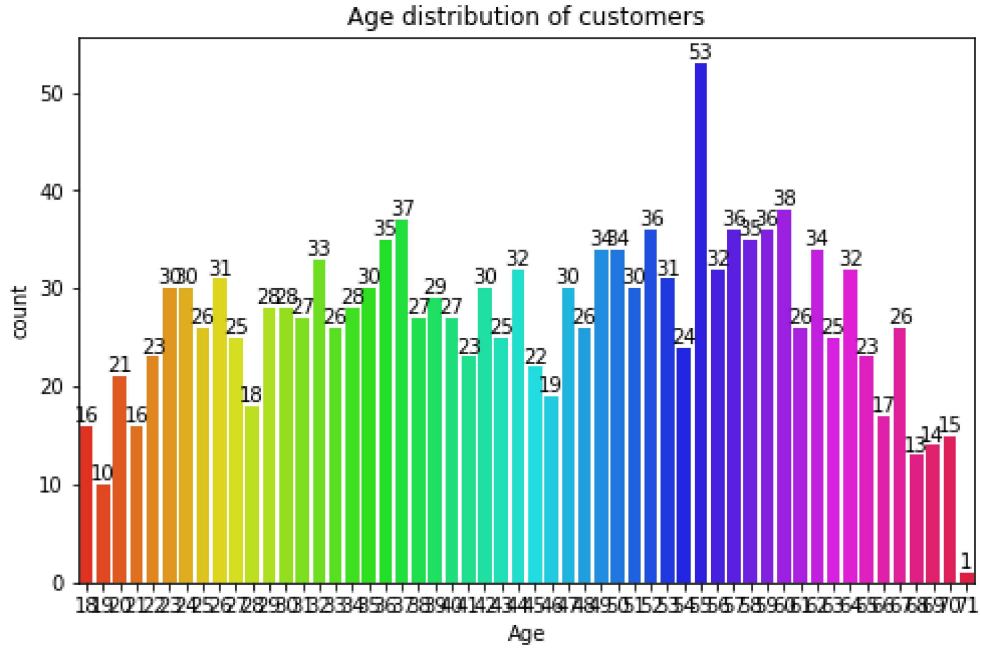
In [148]:
```
#checking for the count of below variables
df['Gender'].value_counts()
df['VisitFrequency'].value_counts()
df['Like'].value_counts()
```

Out[148]:
```
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
```

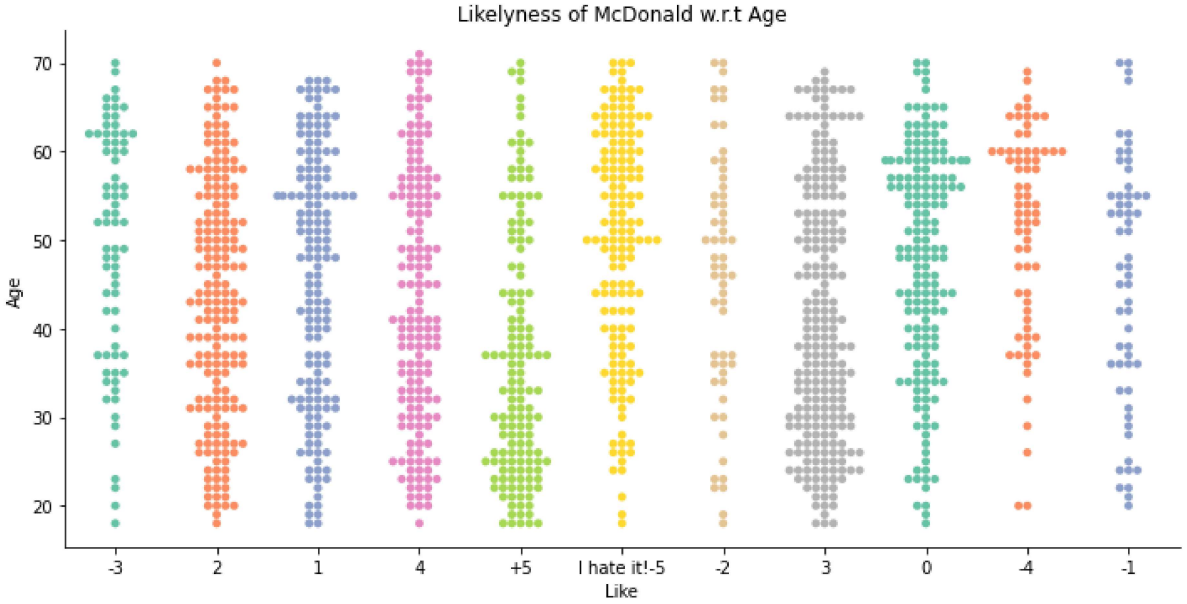
# EXPLORING DATA

In [149]:
```
# customer segmentation - based on socio-demographs(Age & Gender)

# Gender
labels=['Female','Male']
size = df['Gender'].value_counts()
colors=['pink','cyan']
explod=[0,0.1]
plt.rcParams['figure.figsize']=(7,7)
plt.pie(size,colors=colors, labels=labels,shadow=True,autopct='%.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']=(8,5)
f=sns.countplot(x=df['Age'],palette='hsv')
f.bar_label(f.containers[0])
plt.title('Age distribution of customers')
plt.show()
#mcdonalds recieves morecustomers of age between 50-60 and 35-40.
```






In [150]:
```
#customers segementation -based on pyschographic segementation

#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='Set
plt.title('Likelyness of McDonald w.r.t Age')
plt.show()
```



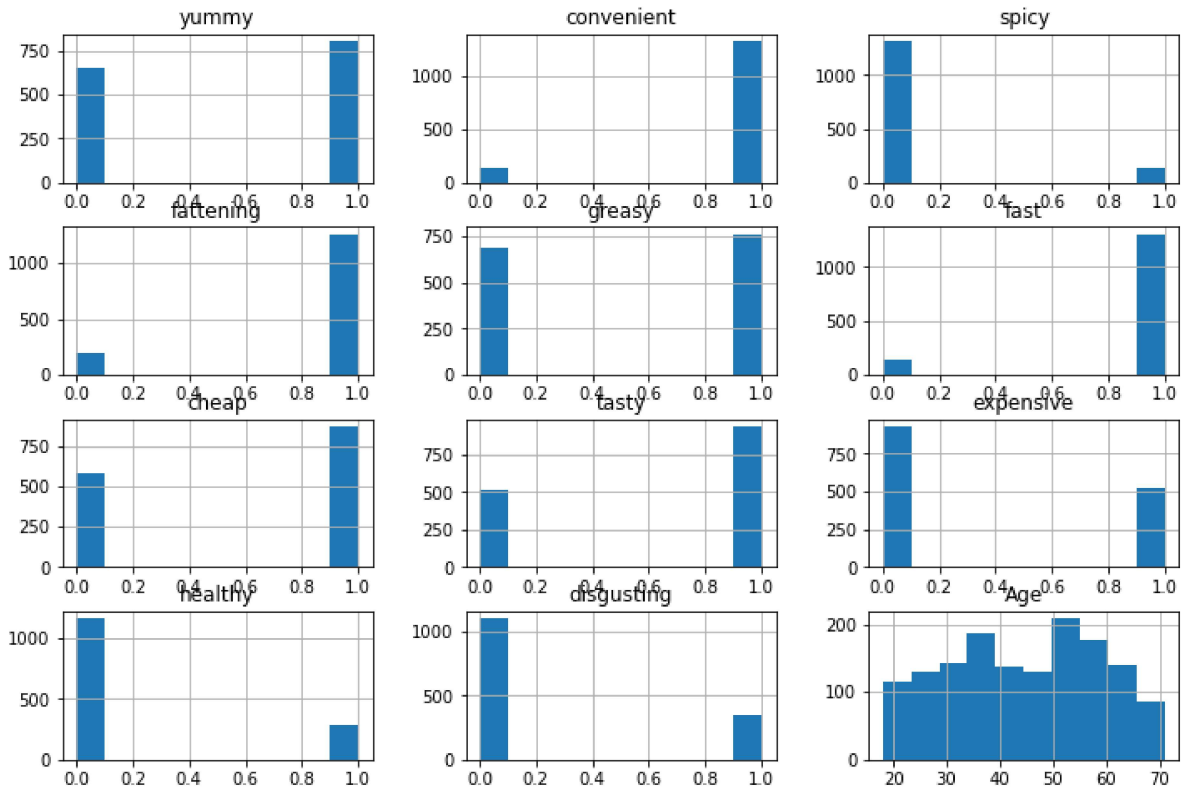
In [151]:
```
# labels encoding for categorical - converting 11 cols with yes/no

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',

for i in cat:
    labelling(i)

df
```

Out[151]:

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

1453 rows × 15 columns

In [152]:
```
# Histogram of the each attributes
plt.rcParams['figure.figsize']=(12,8)
df.hist()
plt.show()
```




In [153]:
```
#Considering only first 11 attributes
df_eleven =df.loc[:,cat]
df_eleven
```

Out[153]:

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

1453 rows × 11 columns

In [154]:
```
#considering only the 11 cols and converting it into array
x=df.loc[:,cat].values
x
```

Out[154]:
```
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]])
```

In [155]:
```
#princopale 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[155]:

| | pc1 | pc2 | pc3 | pc4 | pc5 | pc6 | pc7 | pc8 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.425367 | -0.219079 | 0.663255 | -0.401300 | 0.201705 | -0.389767 | -0.211982 | 0.163235 | 0.18 |
| **1** | -0.218638 | 0.388190 | -0.730827 | -0.094724 | 0.044669 | -0.086596 | -0.095877 | -0.034756 | 0.11 |
| **2** | 0.375415 | 0.730435 | -0.122040 | 0.692262 | 0.839643 | -0.687406 | 0.583112 | 0.364379 | -0.32 |
| **3** | -0.172926 | -0.352752 | -0.843795 | 0.206998 | -0.681415 | -0.036133 | -0.054284 | -0.231477 | -0.02 |
| **4** | 0.187057 | -0.807610 | 0.028537 | 0.548332 | 0.854074 | -0.097305 | -0.457043 | 0.171758 | -0.07 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **1448** | 1.550242 | 0.275031 | -0.013737 | 0.200604 | -0.145063 | 0.306575 | -0.075308 | 0.345552 | -0.13 |
| **1449** | -0.957339 | 0.014308 | 0.303843 | 0.444350 | -0.133690 | 0.381804 | -0.326432 | 0.878047 | -0.30 |
| **1450** | -0.185894 | 1.062662 | 0.220857 | -0.467643 | -0.187757 | -0.192703 | -0.091597 | -0.036576 | 0.03 |
| **1451** | -1.182064 | -0.038570 | 0.561561 | 0.701126 | 0.047645 | 0.193687 | -0.027335 | -0.339374 | 0.02 |
| **1452** | 1.550242 | 0.275031 | -0.013737 | 0.200604 | -0.145063 | 0.306575 | -0.075308 | 0.345552 | -0.13 |

1453 rows × 11 columns

In [156]:
```
#Proportion of variance(from pc1 to pc11)
pca.explained_variance_ratio_
```

Out[156]:
```
array([0.29944723, 0.19279721, 0.13304535, 0.08309578, 0.05948052,
       0.05029956, 0.0438491 , 0.03954779, 0.0367609 , 0.03235329,
       0.02932326])
```

In [157]:
```
np.cumsum(pca.explained_variance_ratio_)
```

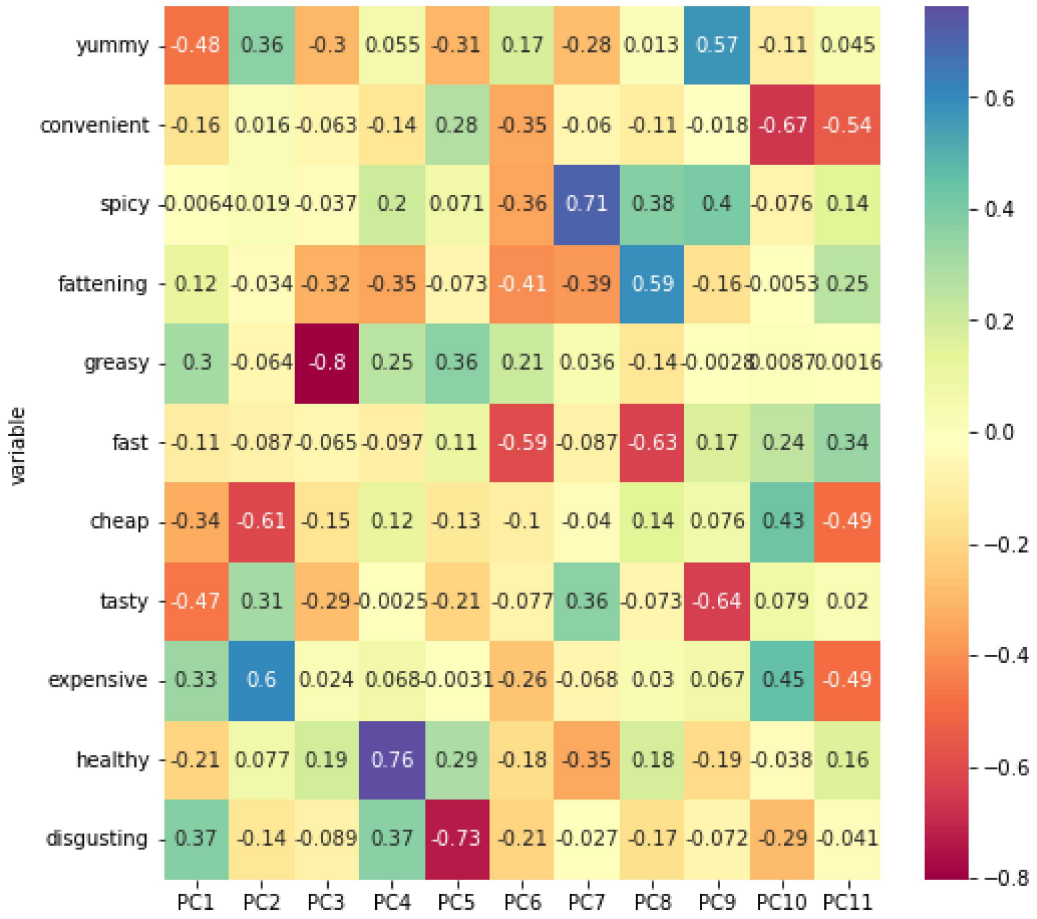
Out[157]:
```
array([0.29944723, 0.49224445, 0.6252898 , 0.70838558, 0.7678661 ,
       0.81816566, 0.86201476, 0.90156255, 0.93832345, 0.97067674,
       1.        ])
```

In [158]:
```
# correlation cofficinet between original variables and the component

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"] = df_eleven.columns.values
loadings_df=loadings_df.set_index('variable')
loadings_df
```

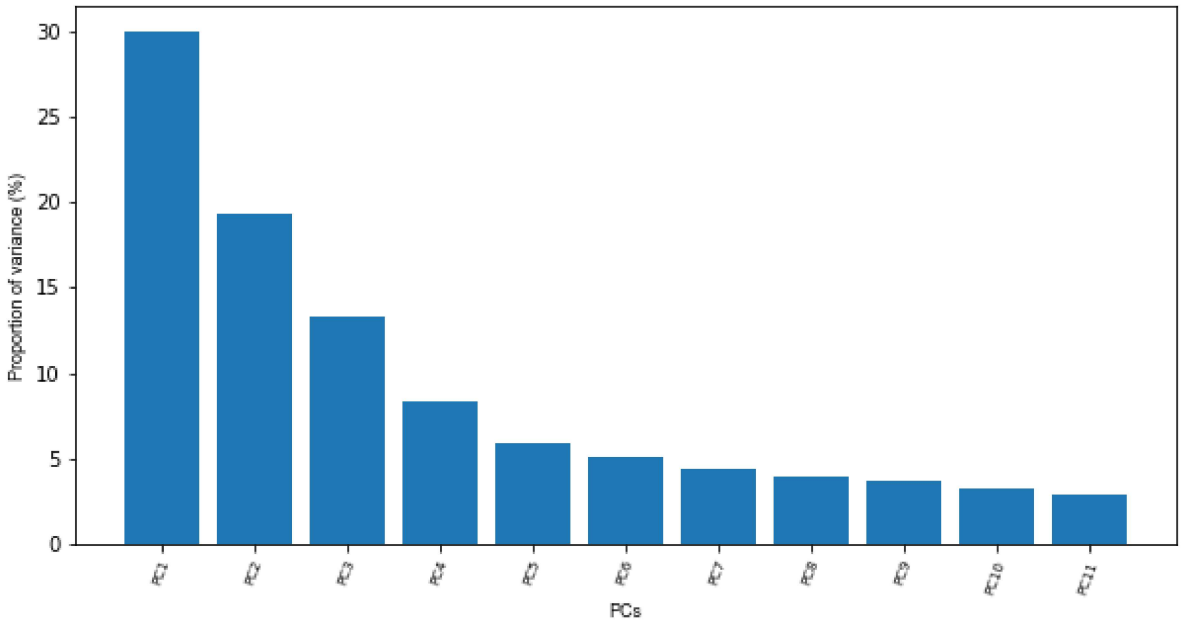
Out[158]:

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

In [159]:
```
# correlation matrix plot for loadings
plt.rcParams['figure.figsize']=(8,8)
ax=sns.heatmap(loadings_df,annot=True,cmap='Spectral')
plt.show()
```



In [160]:
```
# Scree plot(Elbow test)-PCA
from bioinfokit.visuz import cluster
cluster.screeplot(obj=[pc_list,pca.explained_variance_ratio_],show=True,dim=(1
```




In [ ]:
```
```

# Extracting Segments

In [161]:
```
# Extracting segments

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

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Input In [161], in <cell line: 5>()
      1 # Extracting segments
      2
      3 # using k-means clustering analysis
      4 from sklearn.cluster import KMeans
----> 5 from yellowbrick.cluster import  KElbowVisualizer
      6 model=KMeans()
      7 visulizer=KElbowVisualizer(model, k(1,12)).fit(df_eleven)

ModuleNotFoundError: No module named 'yellowbrick'
```

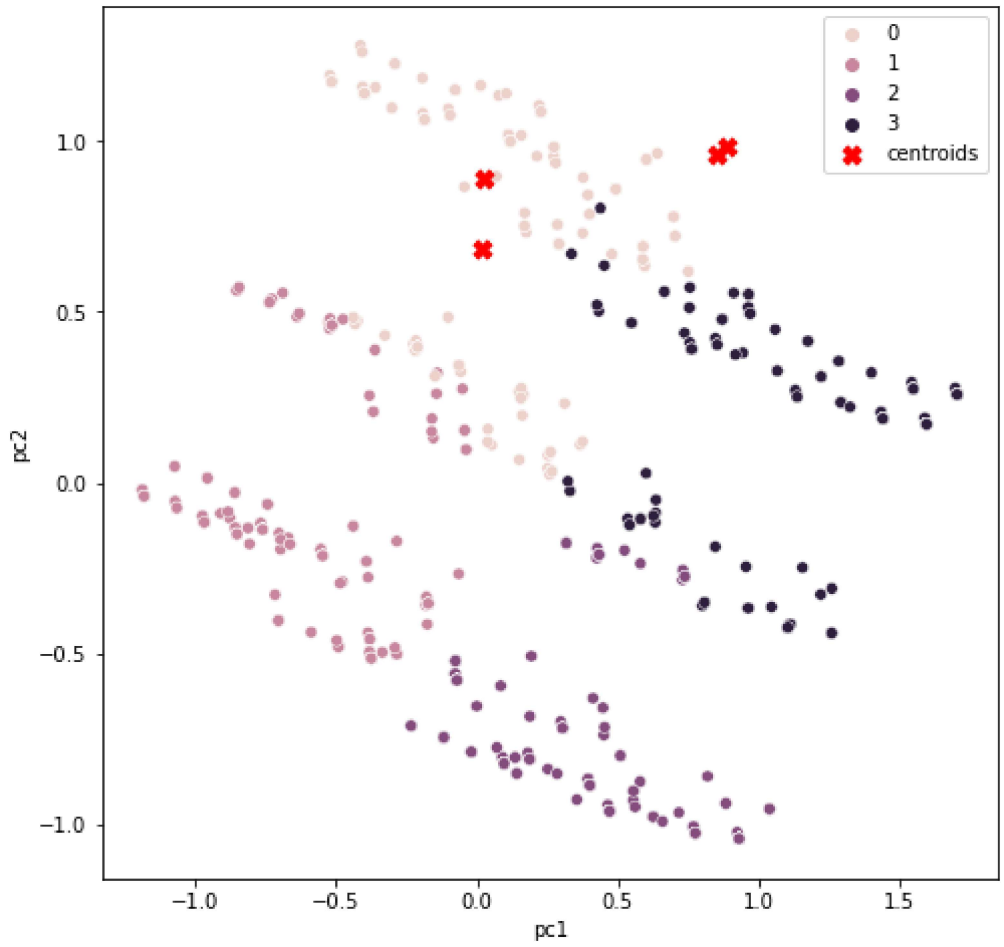
In [162]:
```
# K-means clustering
kmeans = KMeans(n_clusters=4,init='k-means++',random_state=0).fit(df_eleven)
df['cluster_num']=kmeans.labels_
#adding to df
print(kmeans.labels_)
#Labels assined for each data point
print(kmeans.inertia_)
#gives within-cluster sum of squares.
print(kmeans.n_iter_)
#numer of iteration that k-means algorithm runs to get a minimum within-cluste
print(kmeans.cluster_centers_)
#Location of the centroids on each cluster.
```

```
[2 0 0 ... 0 1 3]
1603.0604440558945
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]]
```

In [163]:
```
# To see each cluster size
from collections import Counter
Counter(kmeans.labels_)
```

Out[163]:
```
Counter({2: 304, 0: 323, 1: 580, 3: 246})
```

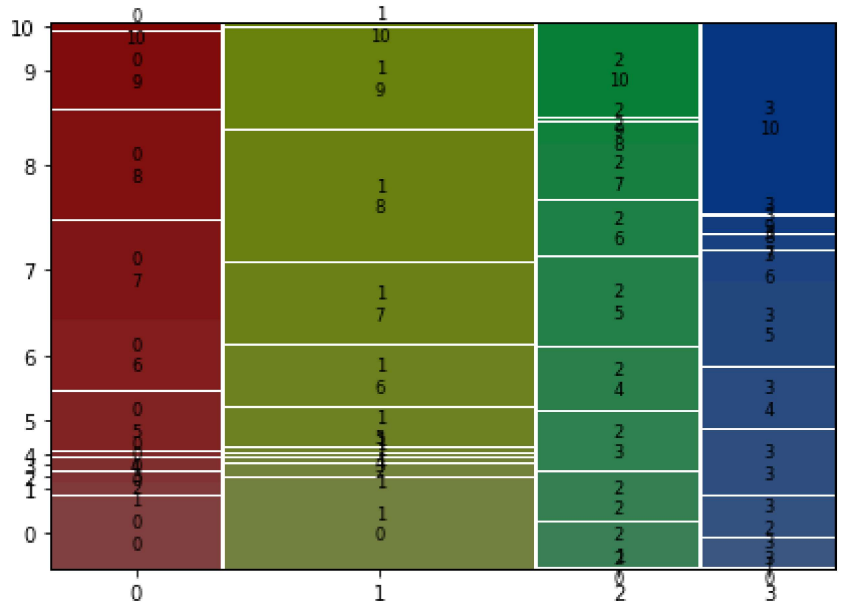
In [164]:
```
# Visualtization clusters
sns.scatterplot(data=pf,x="pc1",y="pc2",hue=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:,0],kmeans.cluster_centers_[:,1],marker="
plt.legend()
plt.show()
```




# DESCRIBING SEGEMENTS

In [ ]:
```
```

In [165]:
```
# MOSTAIC PLOT
plt.rcParams['figure.figsize']=(7,5)
mosaic(crosstab.stack())
plt.show()
```

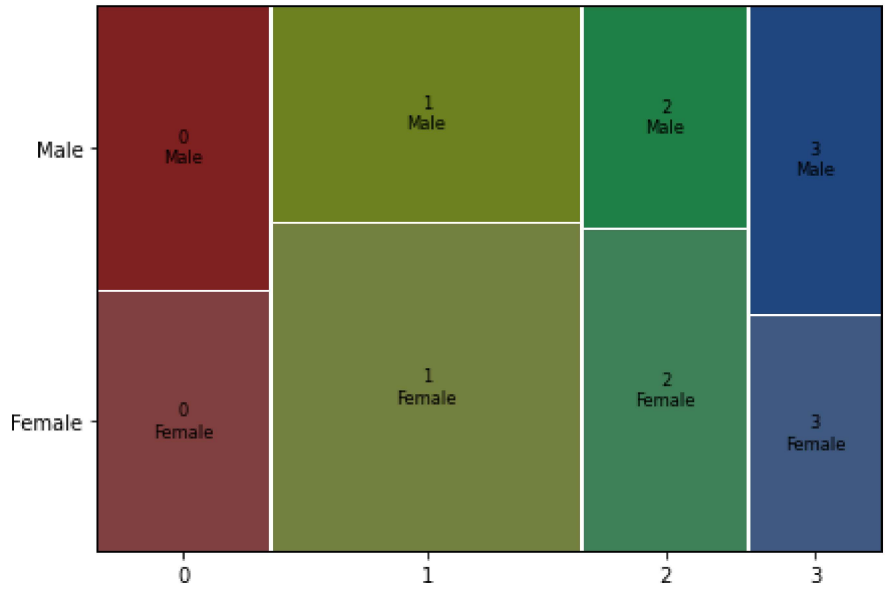



In [166]:
```
#Mosaic plot gender vs segment
crosstab_gender=pd.crosstab(df['cluster_num'],df['Gender'])
crosstab_gender
```

Out[166]:

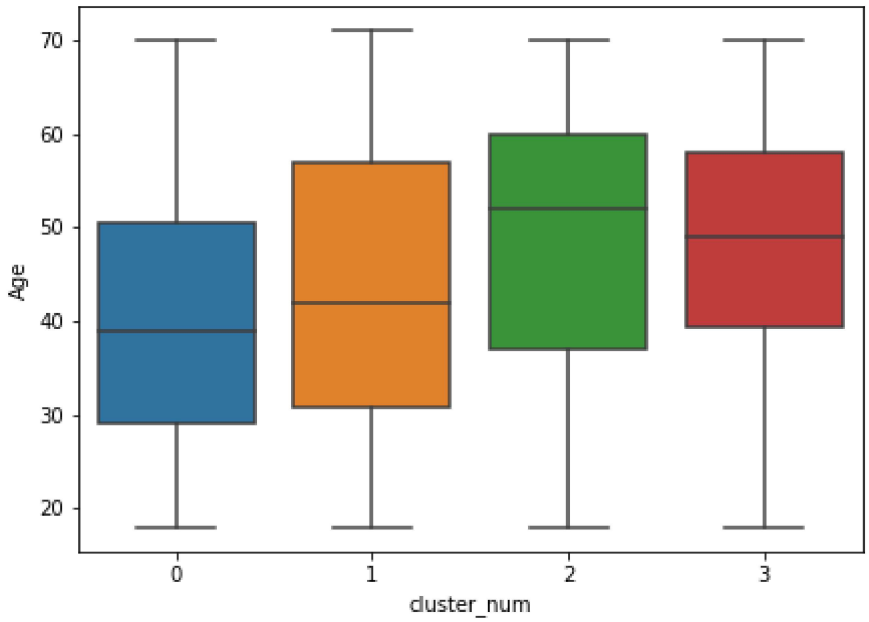
| Gender | Female | Male |
|---|---|---|
| cluster_num | | |
| **0** | 154 | 169 |
| **1** | 349 | 231 |
| **2** | 179 | 125 |
| **3** | 106 | 140 |

In [167]:
```
plt.rcParams['figure.figsize']=(7,5)
mosaic(crosstab_gender.stack())
plt.show()
```




In [168]:
```
# box plot for age

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

Out[168]:
```
<AxesSubplot:xlabel='cluster_num', ylabel='Age'>
```




# Selecting target segment

In [169]:
```
#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[169]:

| | cluster_num | VisitFrequency |
|---|---|---|
| **0** | 0 | 2.547988 |
| **1** | 1 | 2.584483 |
| **2** | 2 | 2.822368 |
| **3** | 3 | 2.654472 |

In [170]:
```
# Like
df['Like']=LabelEncoder().fit_transform(df['Like'])
Like=df.groupby('cluster_num')['Like'].mean()
Like=Like.to_frame().reset_index()
Like
```

Out[170]:

| | cluster_num | Like |
|---|---|---|
| **0** | 0 | 5.897833 |
| **1** | 1 | 5.974138 |
| **2** | 2 | 5.391447 |
| **3** | 3 | 6.211382 |

In [171]:
```
# Gender
df['Gender']=LabelEncoder().fit_transform(df['Gender'])
Gender=df.groupby('cluster_num')['Gender'].mean()
Gender=Gender.to_frame().reset_index()
Gender
```

Out[171]:

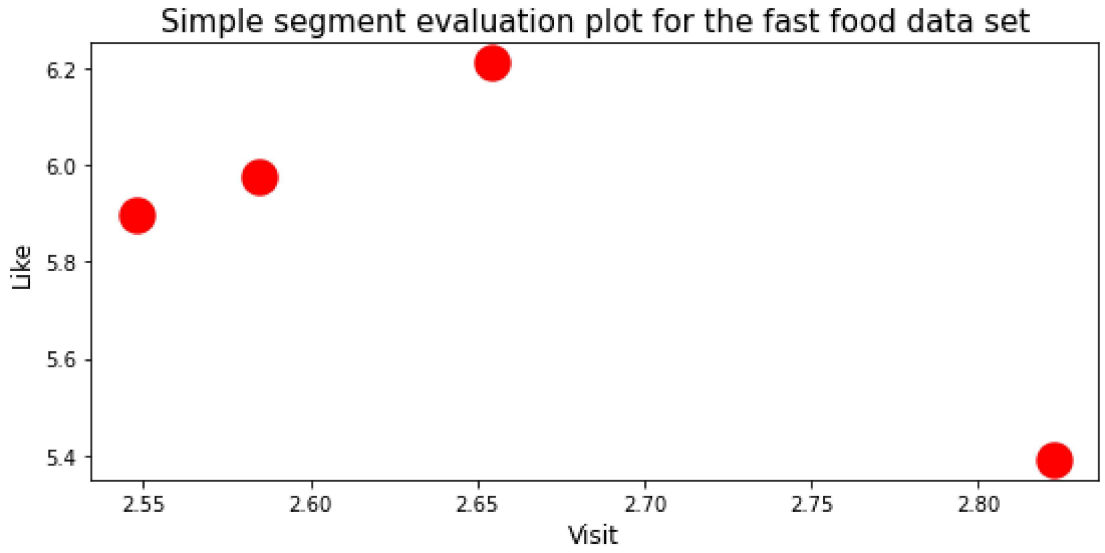
| | cluster_num | Gender |
|---|---|---|
| **0** | 0 | 0.523220 |
| **1** | 1 | 0.398276 |
| **2** | 2 | 0.411184 |
| **3** | 3 | 0.569106 |

In [172]:
```
segment=Gender.merge (Like,on='cluster_num',how='left').merge(visit,on='cluste
segment
```

Out[172]:

| | cluster_num | Gender | Like | VisitFrequency |
|---|---|---|---|---|
| **0** | 0 | 0.523220 | 5.897833 | 2.547988 |
| **1** | 1 | 0.398276 | 5.974138 | 2.584483 |
| **2** | 2 | 0.411184 | 5.391447 | 2.822368 |
| **3** | 3 | 0.569106 | 6.211382 | 2.654472 |

In [173]:
```
# Target segmensts'r
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
plt.xlabel("Visit",fontsize=12)
plt.ylabel("Like",fontsize=12)
plt.show()
```




In [ ]:

In [ ]:

In [ ]: