McDonalds Market segmentation case study

November 12, 2023

- Importing Libraries

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

- Getting Data

[2]:
```
df = pd.read_csv("C:/Users/91902/Downloads/McDonalds Case␣
 ↪Study-20231110T070929Z-001/McDonalds Case Study/mcdonalds.csv")
df.head()
```

[2]:
```
  yummy convenient spicy fattening greasy fast cheap tasty expensive healthy
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

  disgusting Like  Age      VisitFrequency  Gender
0         No   -3   61  Every three months  Female
1         No   +2   51  Every three months  Female
2         No   +1   62  Every three months  Female
3        Yes   +4   69         Once a week  Female
4         No   +2   49        Once a month    Male
```

- There are no missing values.

[3]:
```
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
```

[4]:

```python
df.isna().sum()
```

[4]:

```
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
```

Data Observations - Mean age of customers is 45. - Min age is 18, while the maximum is 71.

Data Processing

[5]:

```python
category = []
for i in df.columns:
  if df[i].dtype=='O':
    category.append(i)


for i in category:
  print('Distribution of',i)
  print(df[i].value_counts())
  print('-'*60)
```

```
Distribution of yummy
yummy
Yes    803
No     650
Name: count, dtype: int64
------------------------------------------------------------
Distribution of convenient
convenient
Yes    1319
No      134
Name: count, dtype: int64
------------------------------------------------------------
Distribution of spicy
spicy
No     1317
Yes     136
Name: count, dtype: int64
------------------------------------------------------------
Distribution of fattening
fattening
Yes    1260
No      193
Name: count, dtype: int64
------------------------------------------------------------
Distribution of greasy
greasy
Yes    765
No     688
Name: count, dtype: int64
------------------------------------------------------------
Distribution of fast
fast
Yes    1308
No      145
Name: count, dtype: int64
------------------------------------------------------------
Distribution of cheap
cheap
Yes    870
No     583
Name: count, dtype: int64
------------------------------------------------------------
Distribution of tasty
tasty
Yes    936
No     517
Name: count, dtype: int64
------------------------------------------------------------
```

```
Distribution of expensive
expensive
No     933
Yes    520
Name: count, dtype: int64
------------------------------------------------------------
Distribution of healthy
healthy
No     1164
Yes     289
Name: count, dtype: int64
------------------------------------------------------------
Distribution of disgusting
disgusting
No     1100
Yes     353
Name: count, dtype: int64
------------------------------------------------------------
Distribution of Like
Like
+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: count, dtype: int64
------------------------------------------------------------
Distribution of VisitFrequency
VisitFrequency
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: count, dtype: int64
------------------------------------------------------------
Distribution of Gender
Gender
Female    788
Male      665
Name: count, dtype: int64
```

```
------------------------------------------------------------
```

Observations * Majority of the customers visits once a month * +3 is given my most of the customers * 60% customers Found the food yummy * Approx 90 percent doesn't found convinent and spicy * Most of the customers found the service fast and cheap * A few customers found the food disgusting * Majority customers are Female customers

[6]:
```
df['Age'].value_counts().sort_values()
```

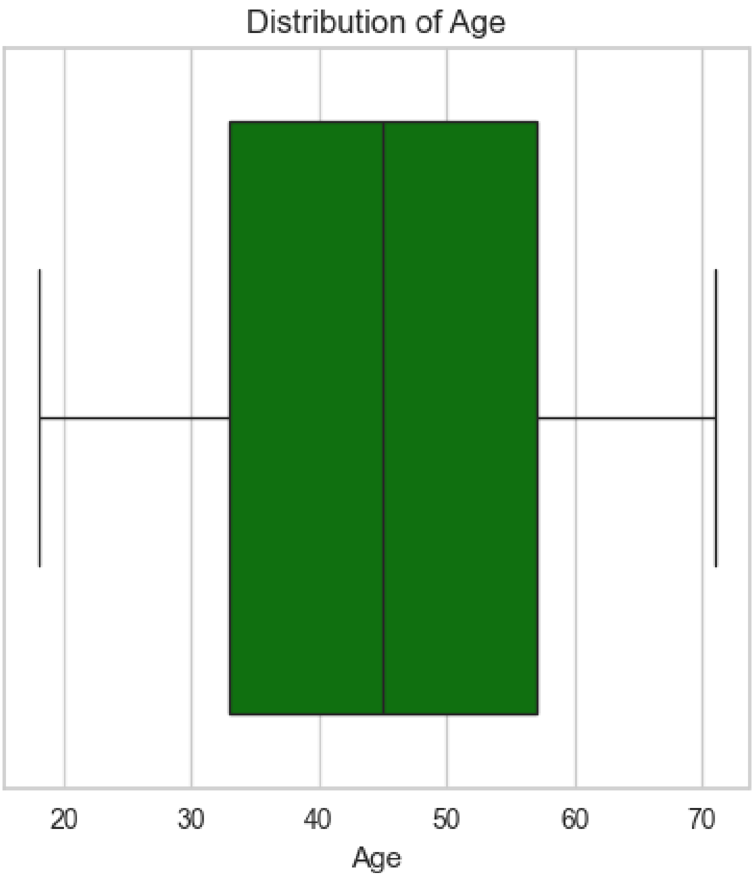
[6]:
```
Age
71     1
19    10
68    13
69    14
70    15
18    16
21    16
66    17
28    18
46    19
20    21
45    22
41    23
65    23
22    23
54    24
63    25
27    25
43    25
48    26
67    26
61    26
33    26
25    26
38    27
31    27
40    27
30    28
29    28
34    28
39    29
23    30
42    30
47    30
51    30
35    30
24    30
26    31
```

```
53    31
44    32
64    32
56    32
32    33
50    34
62    34
49    34
36    35
58    35
52    36
57    36
59    36
37    37
60    38
55    53
Name: count, dtype: int64
```

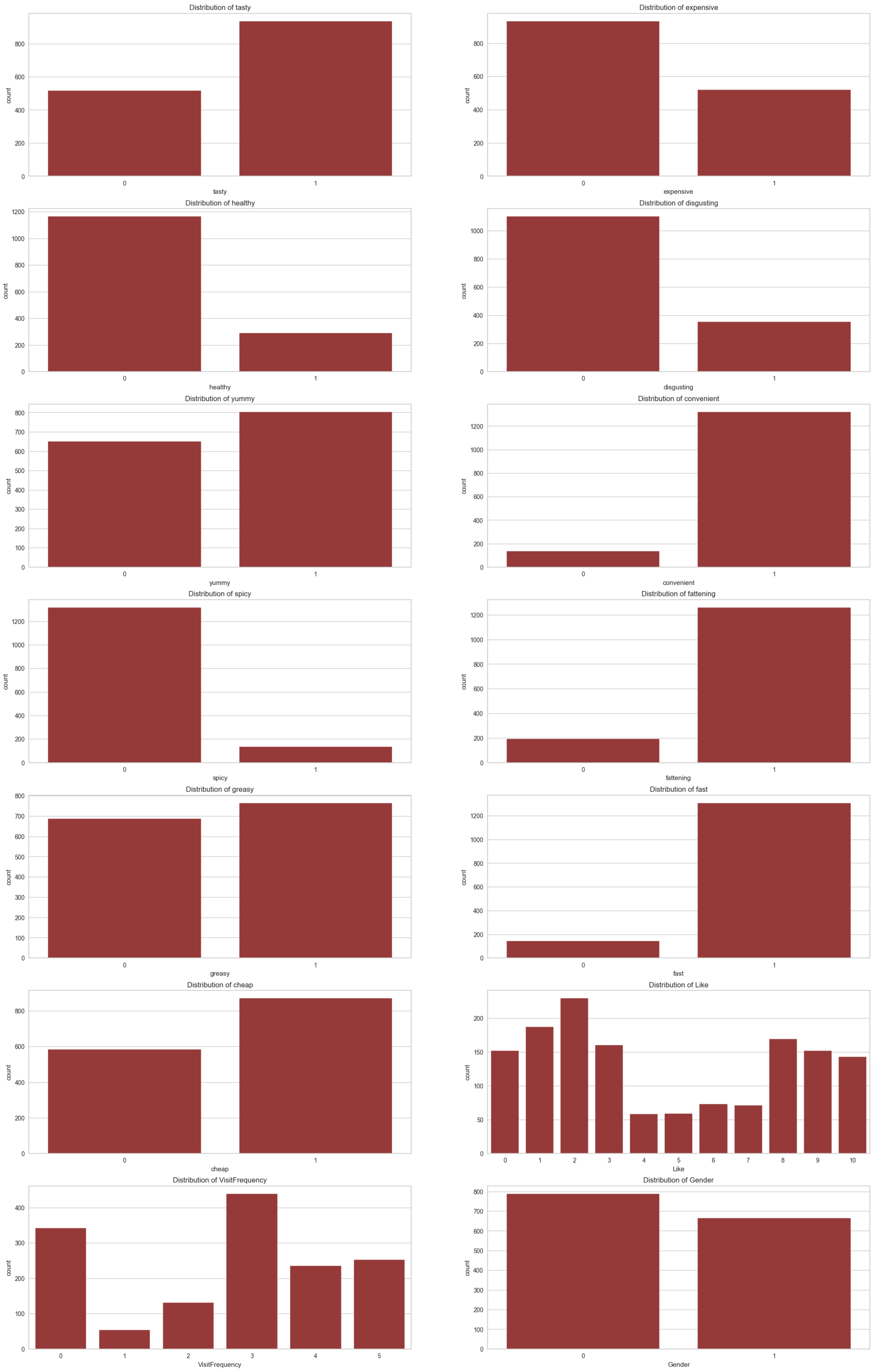
Observations - Majority of the customers aged between 36-49. - Only 11% of customers belong to the adult age category.

Data Visualization

[51]:
```
sns.set_style('whitegrid')
plt.figure(figsize=(5,5))
sns.set_palette('coolwarm')
sns.boxplot(x=df['Age'],color = 'green')
plt.title('Distribution of Age')
plt.show()
```



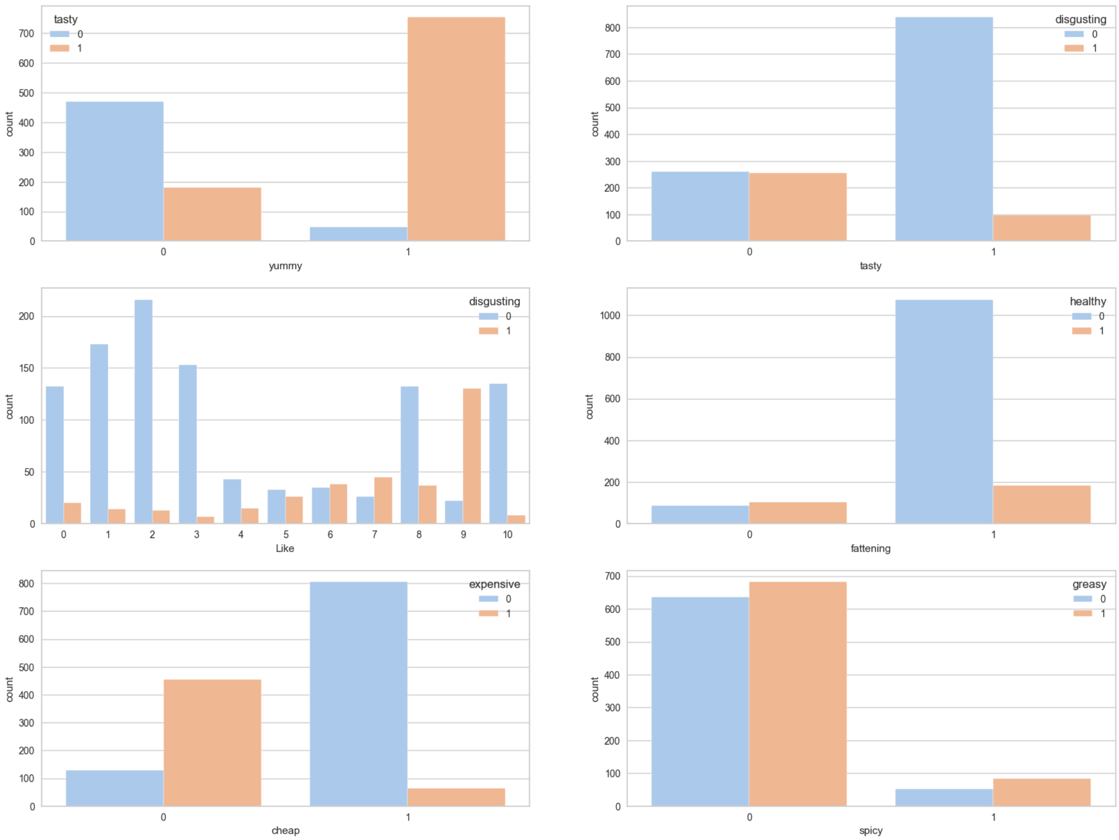
[33]:
```python
fig,([ax0,ax1],[ax2,ax3],[ax4,ax5],[ax6,ax7],[ax8,ax9],[ax10,ax11],[ax12,ax13]) = plt.subplots(ncols=2,nrows=7,figsize=(25,40))

ax = [ax4,ax5,ax6,ax7,ax8,ax9,ax10,ax0,ax1,ax2,ax3,ax11,ax12,ax13]
for i in range(0,14):
  sns.countplot(data=df,x=category[i],ax=ax[i], color='brown')
  ax[i].set_title('Distribution of '+category[i])
```


Distribution of tasty
count
0
200
400
600
800
0
1
tasty
Distribution of expensive
count
0
200
400
600
800
0
1
expensive
Distribution of healthy
count
0
200
400
600
800
1000
1200
0
1
healthy
Distribution of disgusting
count
0
200
400
600
800
1000
0
1
disgusting
Distribution of yummy
count
0
100
200
300
400
500
600
700
800
0
1
yummy
Distribution of convenient
count
0
200
400
600
800
1000
1200
0
1
convenient
Distribution of spicy
count
0
200
400
600
800
1000
1200
0
1
spicy
Distribution of fattening
count
0
200
400
600
800
1000
1200
0
1
fattening
Distribution of greasy
count
0
100
200
300
400
500
600
700
800
0
1
greasy
Distribution of fast
count
0
200
400
600
800
1000
1200
0
1
fast
Distribution of cheap
count
0
200
400
600
800
0
1
cheap
Distribution of Like
count
0
50
100
150
200
0
1
2
3
4
5
6
7
8
9
10
Like
Distribution of VisitFrequency
count
0
100
200
300
400
0
1
2
3
4
5
VisitFrequency
Distribution of Gender
count
0
100
200
300
400
500
600
700
800
0
1
Gender

Observations * There are many customers who have never visited once * Majority of the customers visits once a month * +3 and +2 is given by approx 30 percent the customers * 60% customers Found the food yummy * Approx 90 percent doesn't found convinent and spicy * Most of the customers found the service fast and cheap * A few customers found the food disgusting * Majority customers are Female customers * A big group of customers said the food is fatty

```
[34]: sns.set_palette('pastel')
      fig,([ax0,ax1],[ax2,ax3],[ax4,ax5]) = plt.
       ↪subplots(nrows=3,ncols=2,figsize=(20,15))
      sns.countplot(x=df['cheap'],hue=df['expensive'],ax=ax4)
      sns.countplot(x=df['yummy'],hue=df['tasty'],ax=ax0)
      sns.countplot(x=df['fattening'],hue=df['healthy'],ax=ax3)
      sns.countplot(x=df['spicy'],hue=df['greasy'],ax=ax5)
      sns.countplot(x=df['tasty'],hue=df['disgusting'],ax=ax1)
      sns.countplot(hue=df['disgusting'],x=df['Like'],ax=ax2)
```

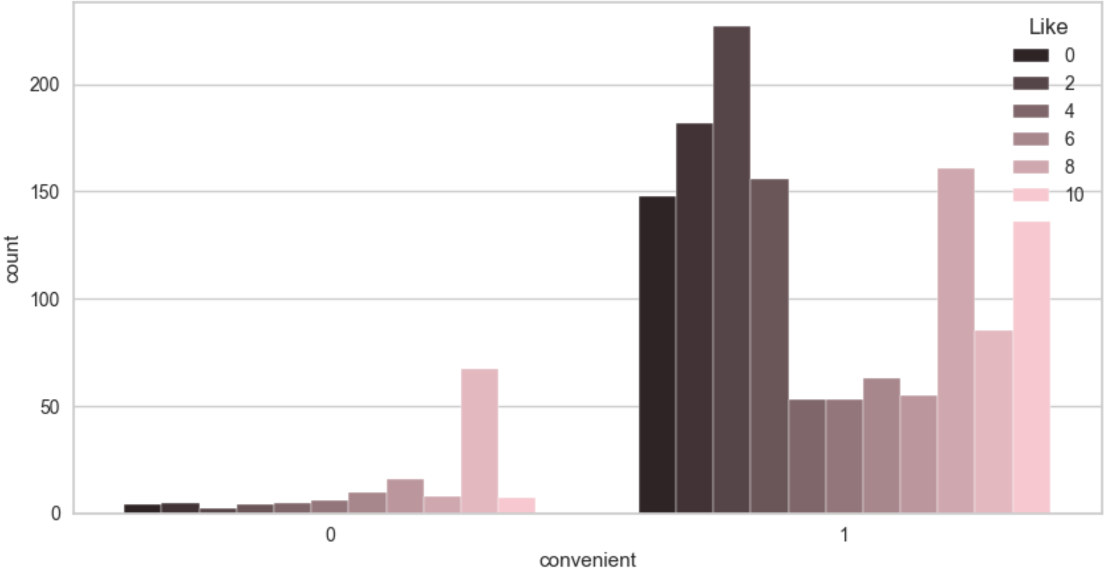
```
[34]: <Axes: xlabel='Like', ylabel='count'>
```

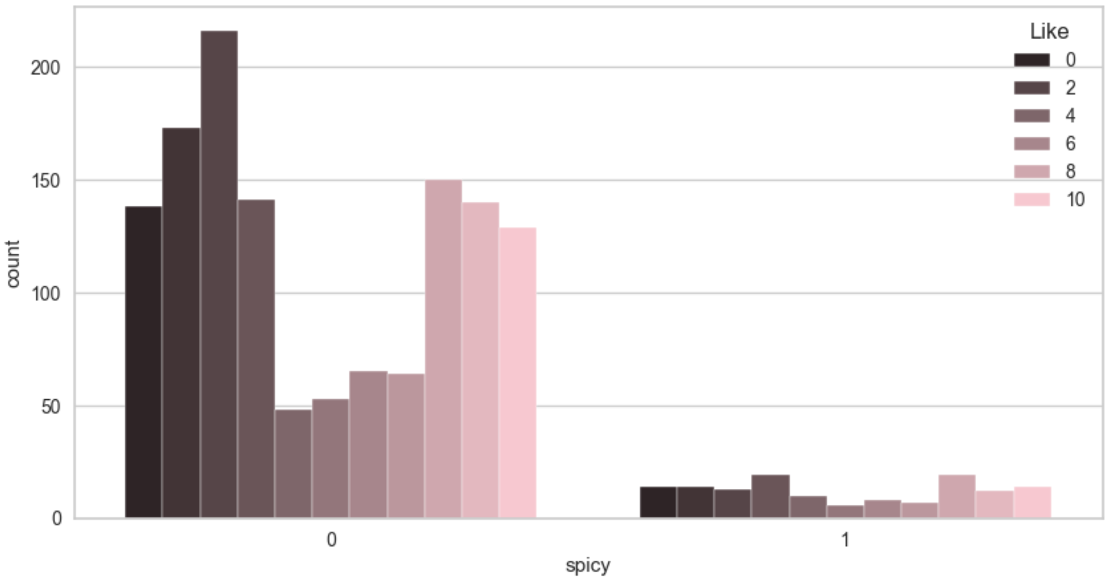


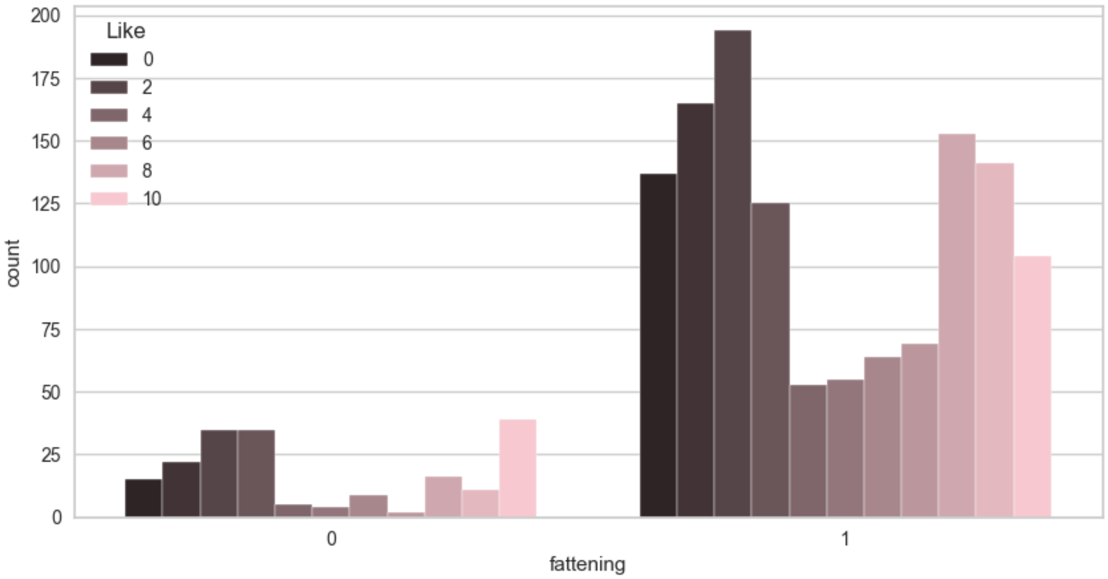
Observations

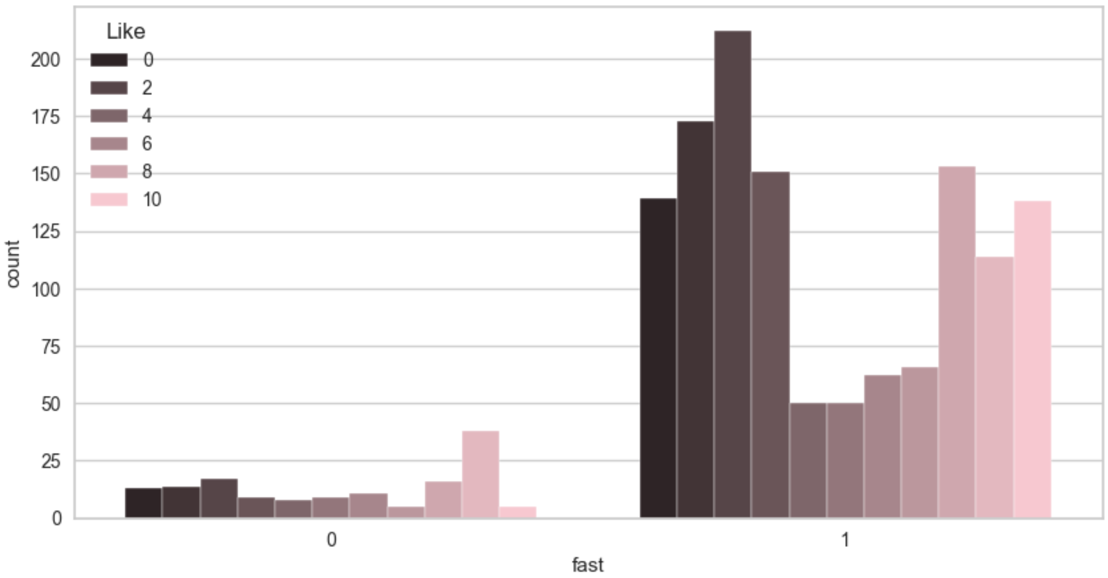
- From the plot it can be seen data have alot of discrepencies
- **yummy** and **tasty** are a kind of same can remove either of one
- Some of the customers rate the food tasty as well as disgusting and vice-versa, needs to check the data
- same error can be seen in **cheap,expensive,disgusting,Likes,fattening,healthy**
- **spicy** and **grease** are highly correlated, can remove either of them
- Needs to check the data for discrepency and if needs to remove the values than we'll

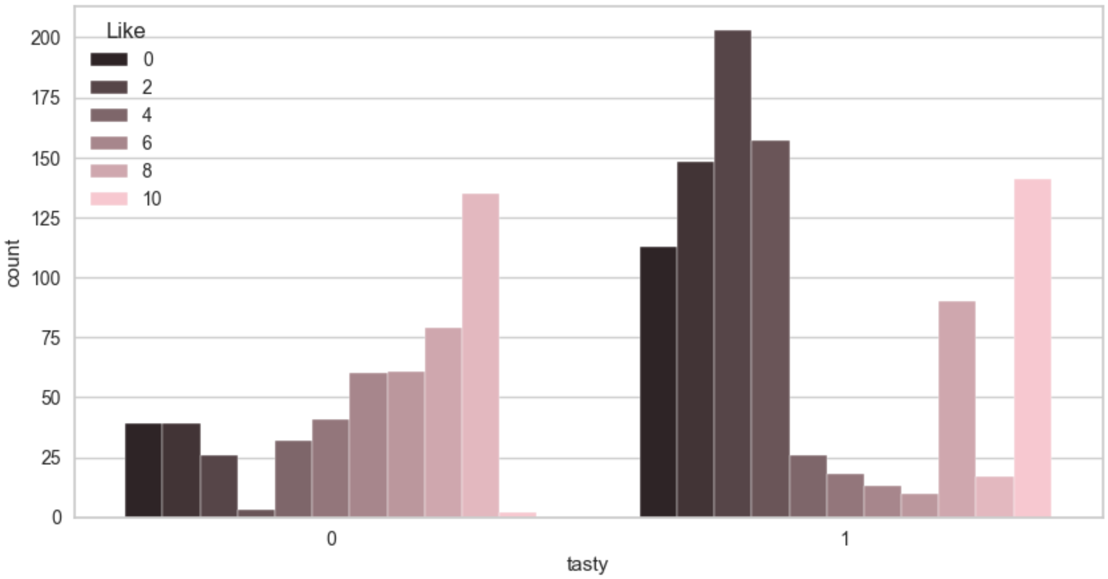
```
[38]: sns.set_style('whitegrid')
      for i in df.drop(['Like','yummy','cheap','healthy','greasy','Age'],axis=1).
       ↪columns:
        plt.figure(figsize=(10,5))
        sns.countplot(x=df[i],hue=df['Like'] ,color= "pink")
        plt.show()
```

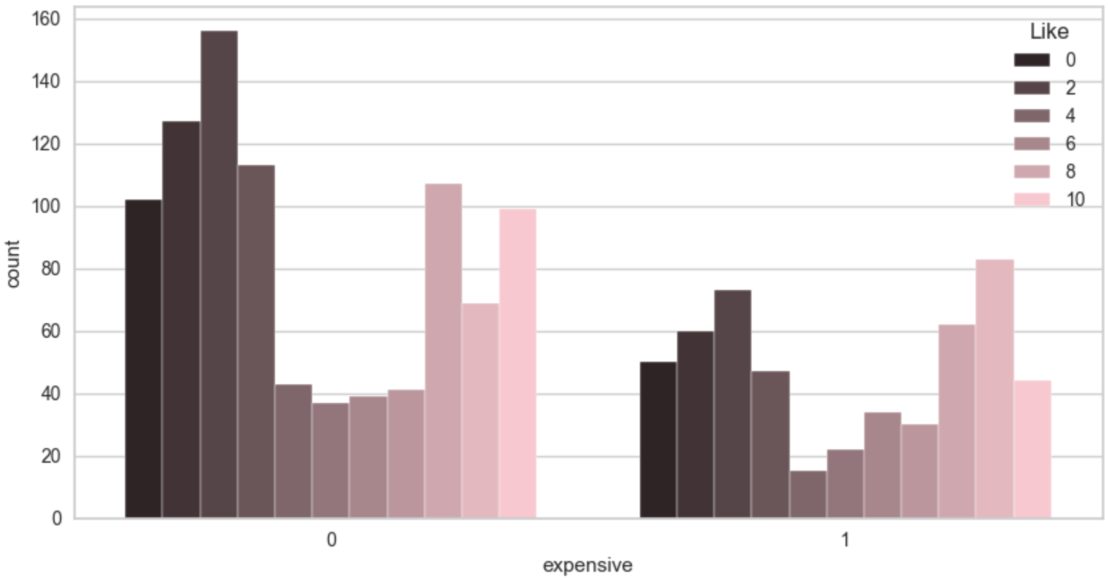


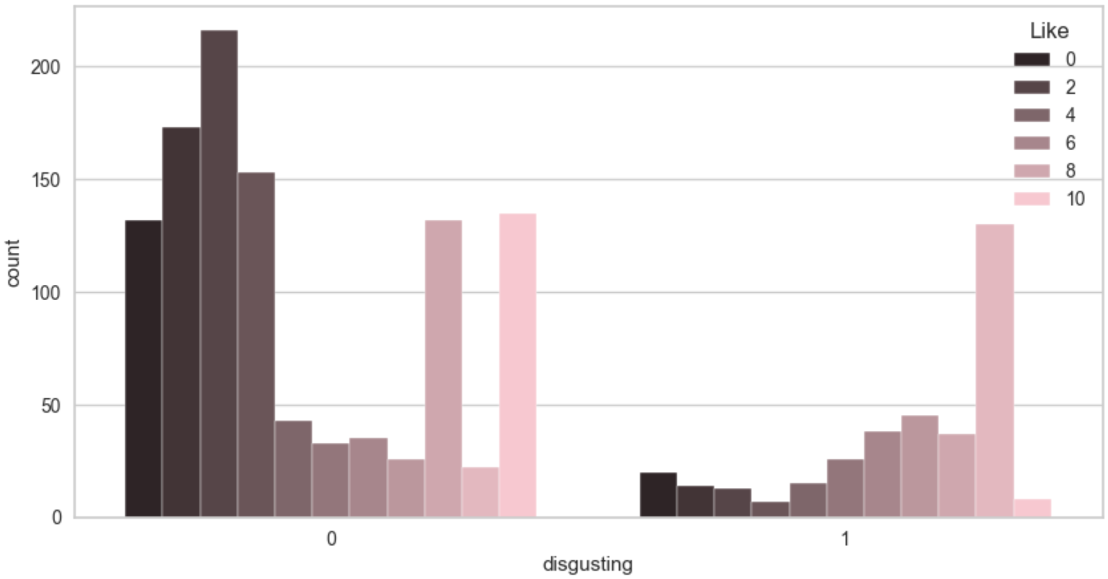

Like
0
2
4
6
8
10
count
0
50
100
150
200
0
1
spicy

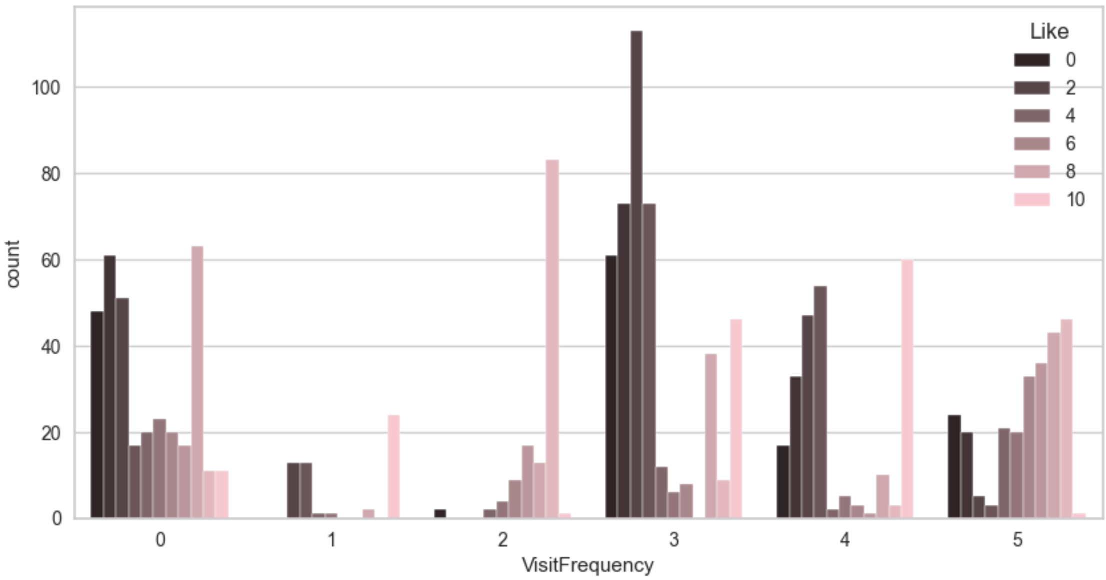

Like
0
2
4
6
8
10
count
0
25
50
75
100
125
150
175
200
0
1
fattening

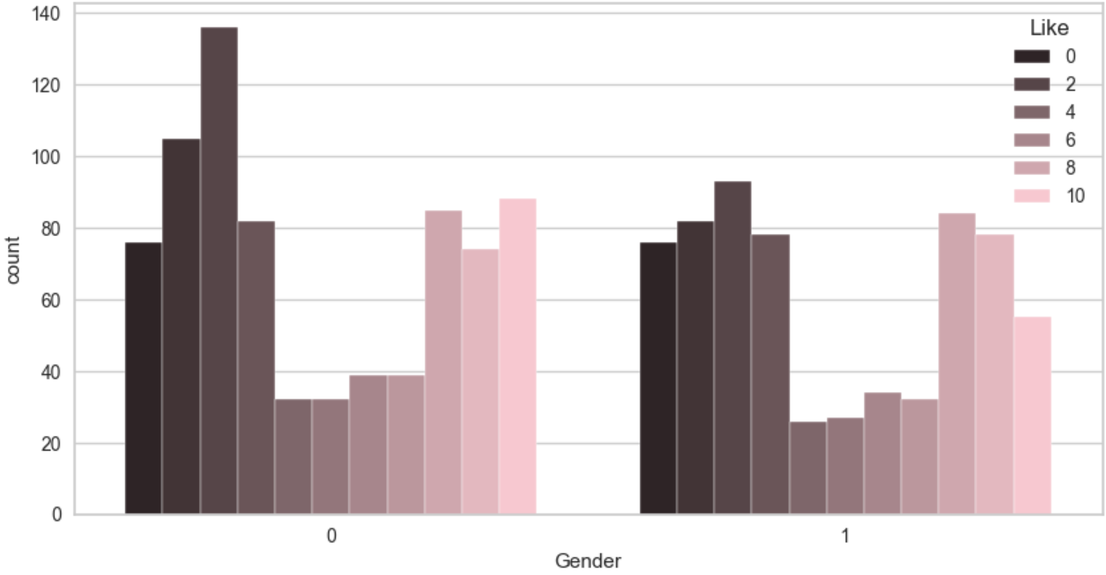

Like
0
2
4
6
8
10
count
200
175
150
125
100
75
50
25
0
0
1
fast

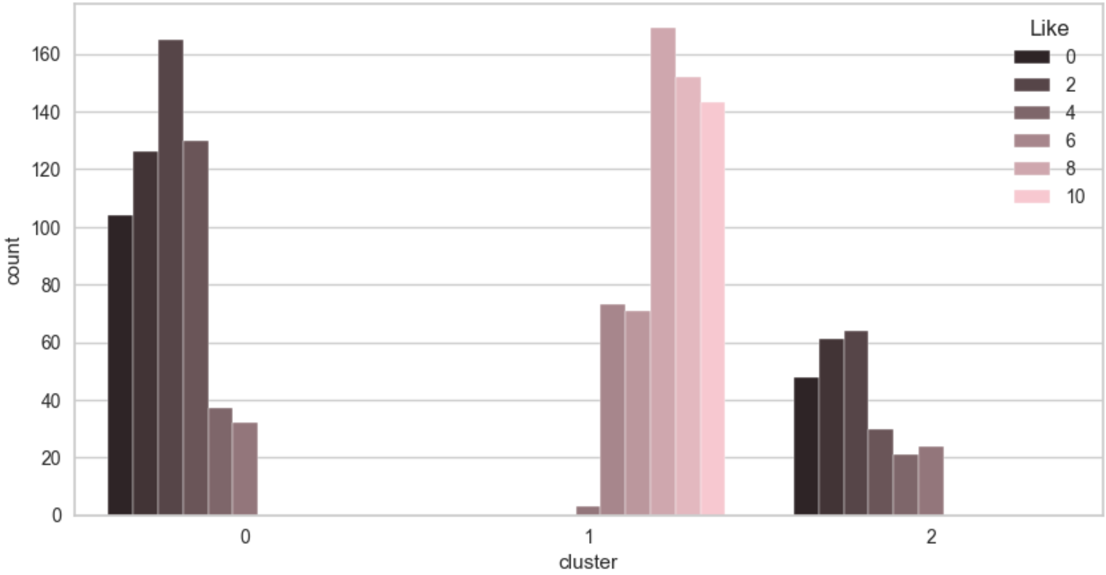

Like
0
2
4
6
8
10
count
200
175
150
125
100
75
50
25
0
0
1
tasty


count
0
20
40
60
80
100
120
140
160
Like
0
2
4
6
8
10
0
1
expensive


count
0
50
100
150
200
Like
0
2
4
6
8
10
0
1
disgusting


Like
0
2
4
6
8
10
count
0
20
40
60
80
100
0
1
2
3
4
5
VisitFrequency


Like
0
2
4
6
8
10
count
0
20
40
60
80
100
120
140
0
1
Gender



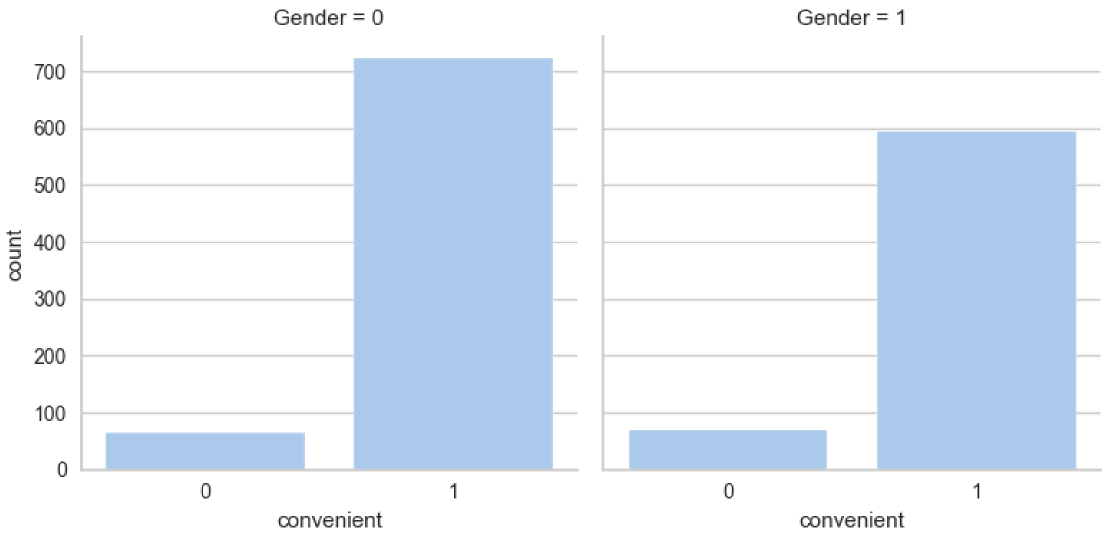
Observations - Customers finding food ‘inconvenient’ tend to rate it lower. - Most customers who disliked the food gave a rating of ‘I hate it! -5’.

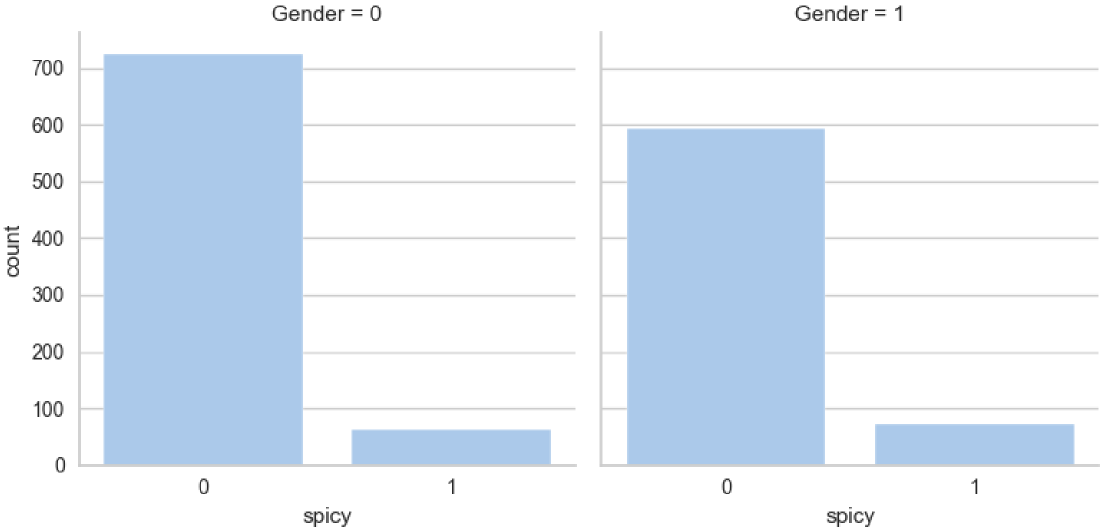
[11]:

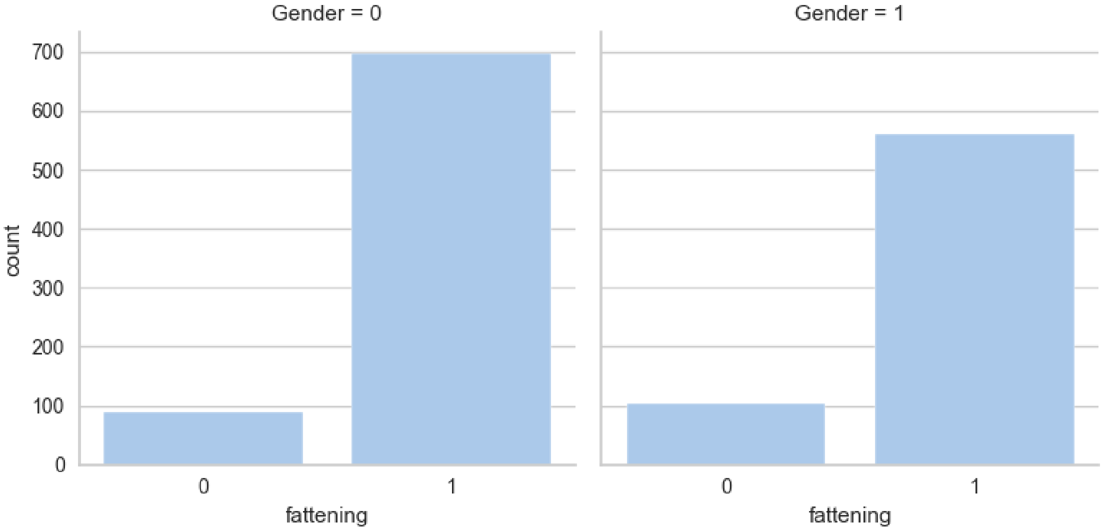
```
import warnings
warnings.filterwarnings("ignore")
```

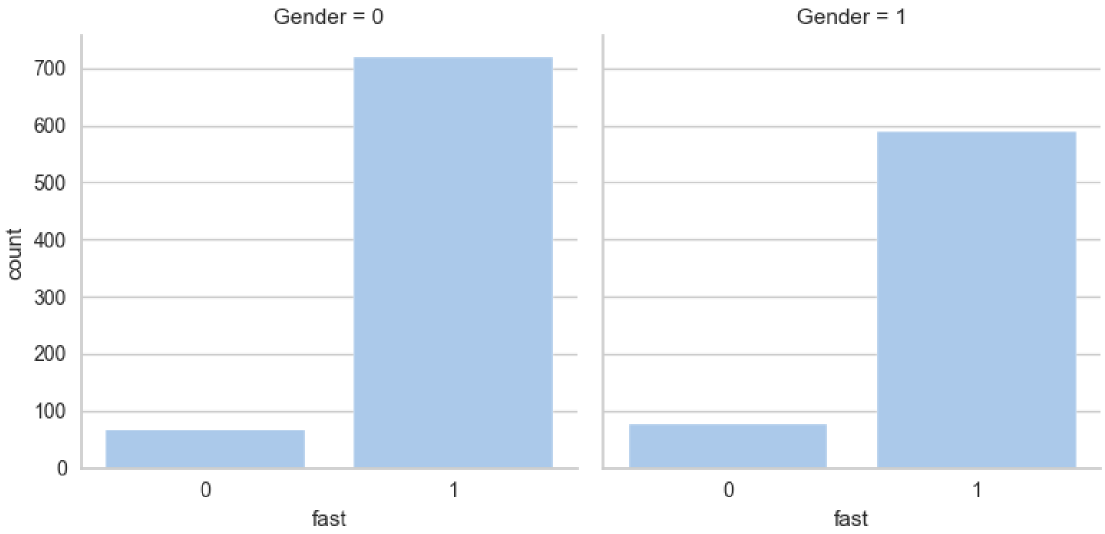
[45]:

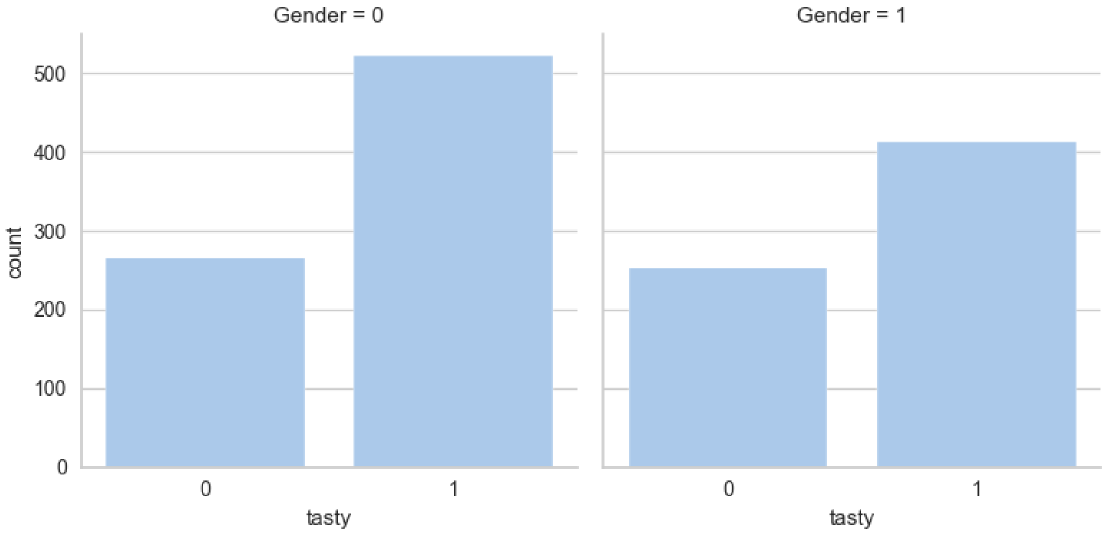
```
sns.set_palette('pastel')
for i in df.
 ↪drop(['Gender','yummy','cheap','healthy','greasy','Age','VisitFrequency'],axis=1):
 ↪
  grid = sns.FacetGrid(df,height=4,col='Gender')
  grid = grid.map(sns.countplot,i)
```

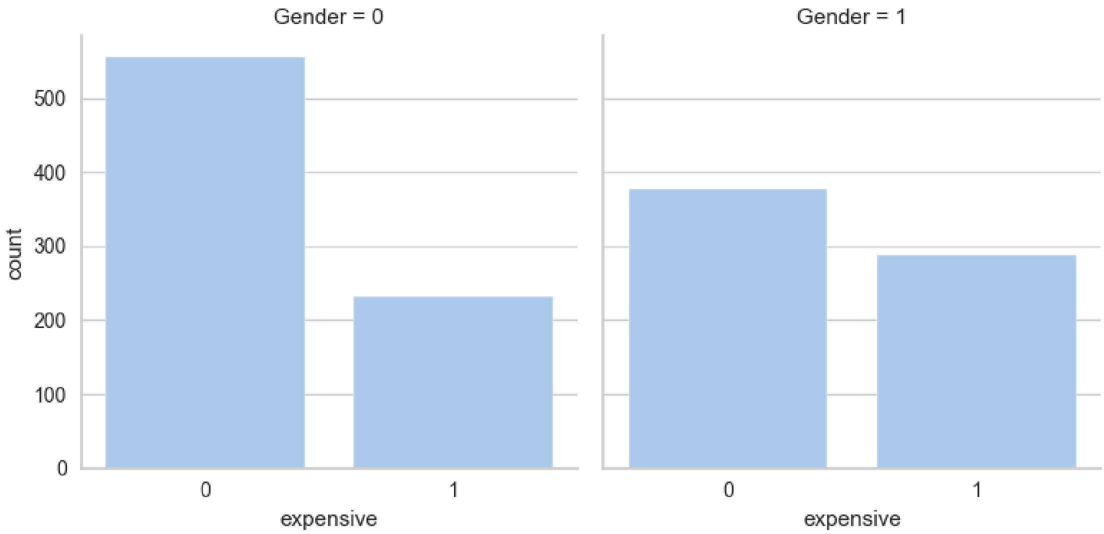


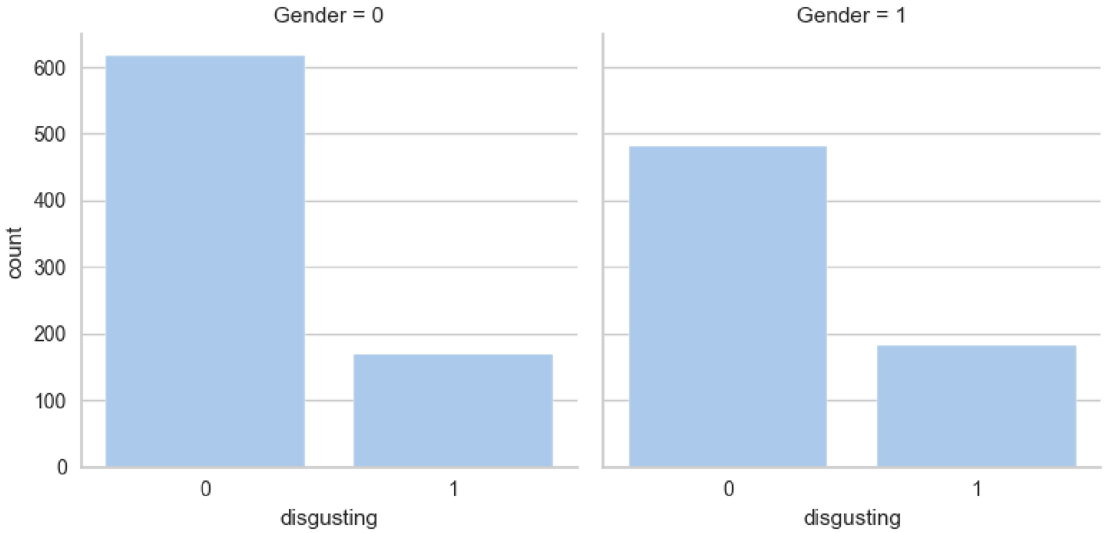

Gender = 0
Gender = 1
count
700
600
500
400
300
200
100
0
0
1
spicy
0
1
spicy

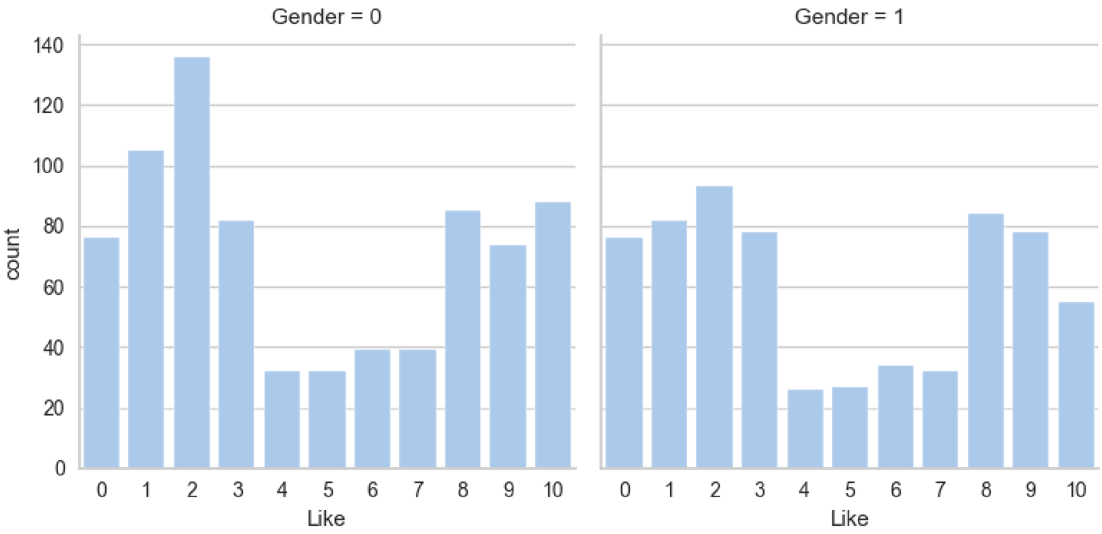

Gender = 0
Gender = 1
count
700
600
500
400
300
200
100
0
0
1
fattening
0
1
fattening

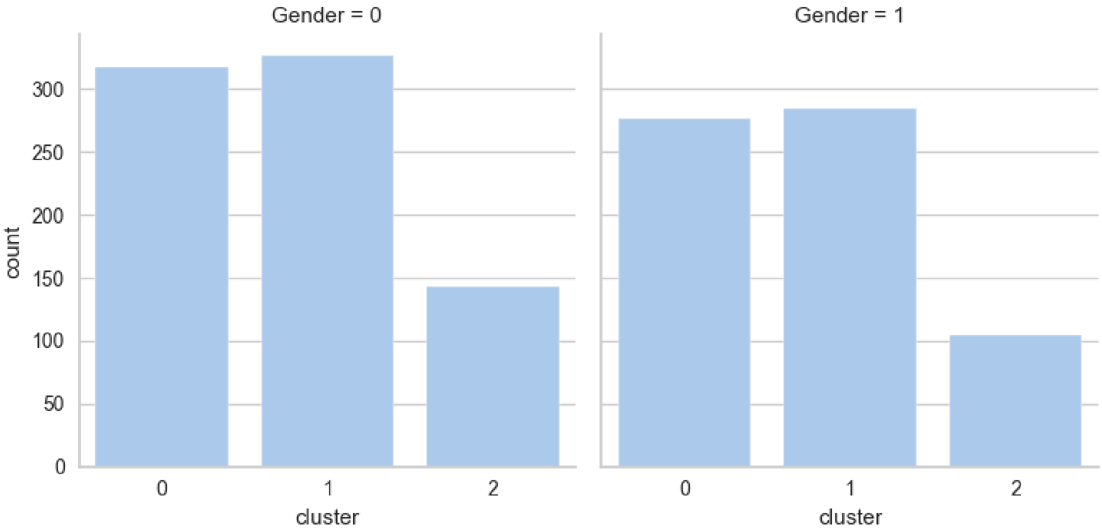

Gender = 0
Gender = 1
count
700
600
500
400
300
200
100
0
0
1
fast
0
1
fast


Gender = 0
Gender = 1
count
500
400
300
200
100
0
0
1
tasty
0
1
tasty


Gender = 0
Gender = 1
count
0
100
200
300
400
500
0
1
expensive
0
1
expensive

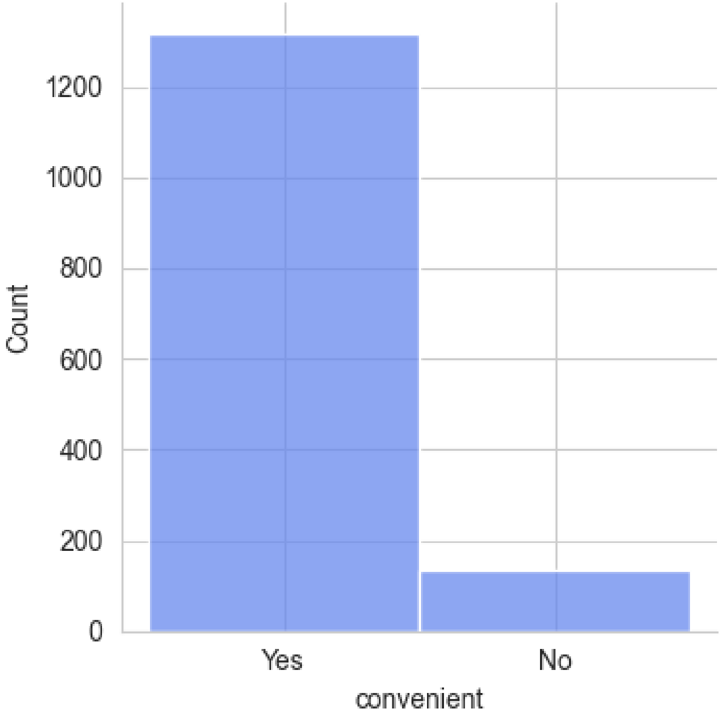

Gender = 0
Gender = 1
count
0
100
200
300
400
500
600
0
1
disgusting
0
1
disgusting



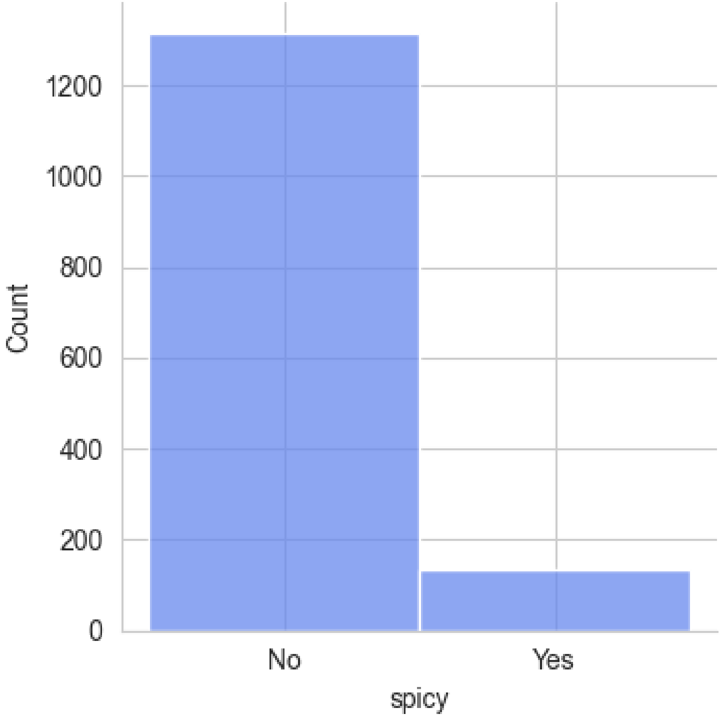


Observations - Female customers find the food less convenient compared to male customers. - Majority of female customers find the food expensive, while males don't. - Both male and female customers are distributed almost equally.

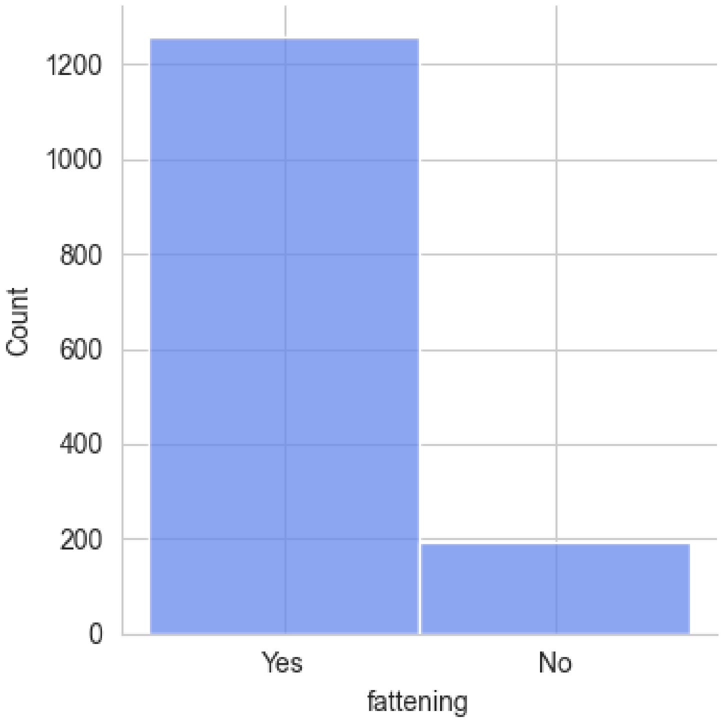
[13]:
```
sns.set_palette('coolwarm')
for i in df.
 ↪drop(['yummy','cheap','healthy','greasy','Age','VisitFrequency'],axis=1):
  grid = sns.FacetGrid(df,height=4)
  grid = grid.map(sns.histplot,i,bins=30)
```

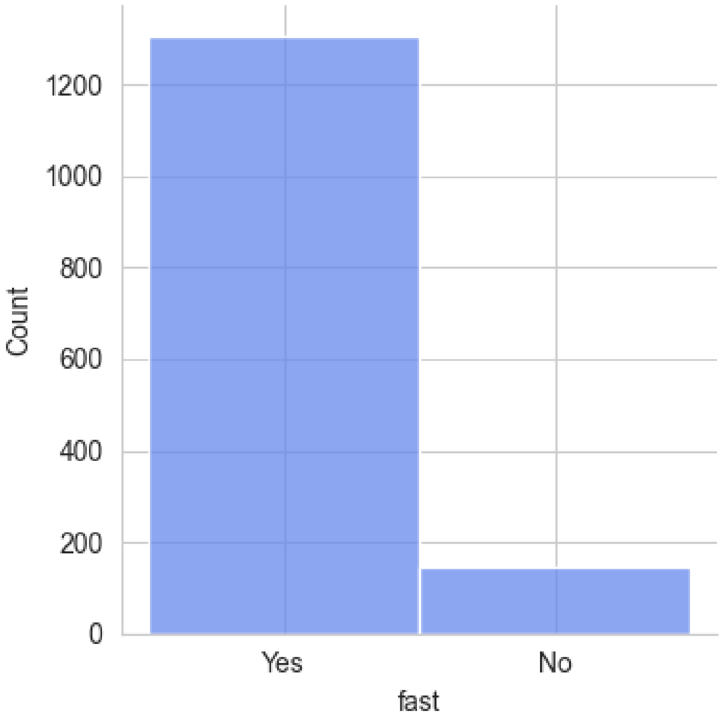


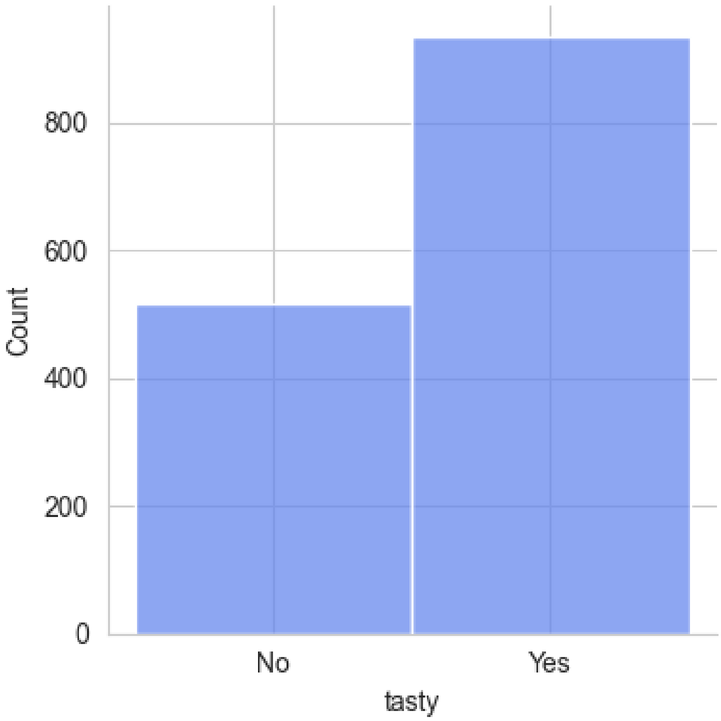
Count
0
200
400
600
800
1000
1200
Yes
No
convenient

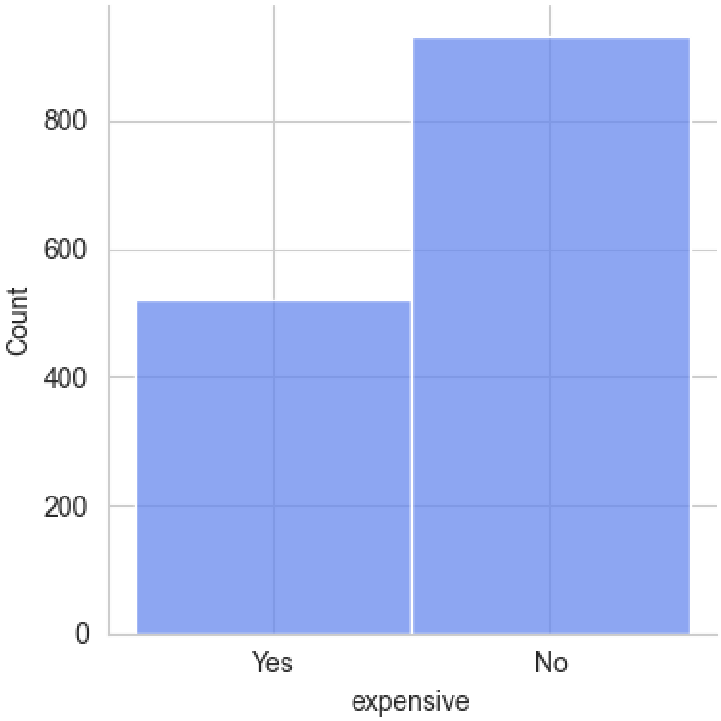


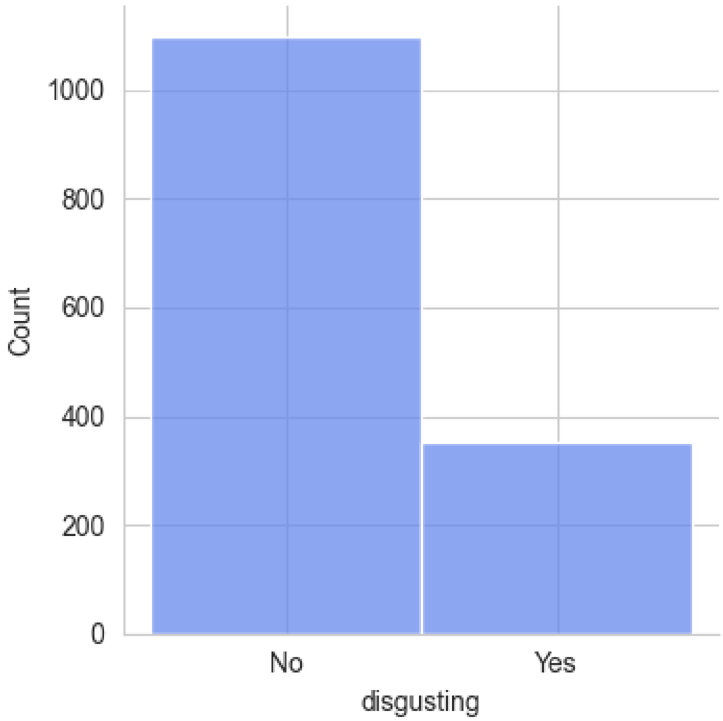
Count
0
200
400
600
800
1000
1200
No
Yes
spicy

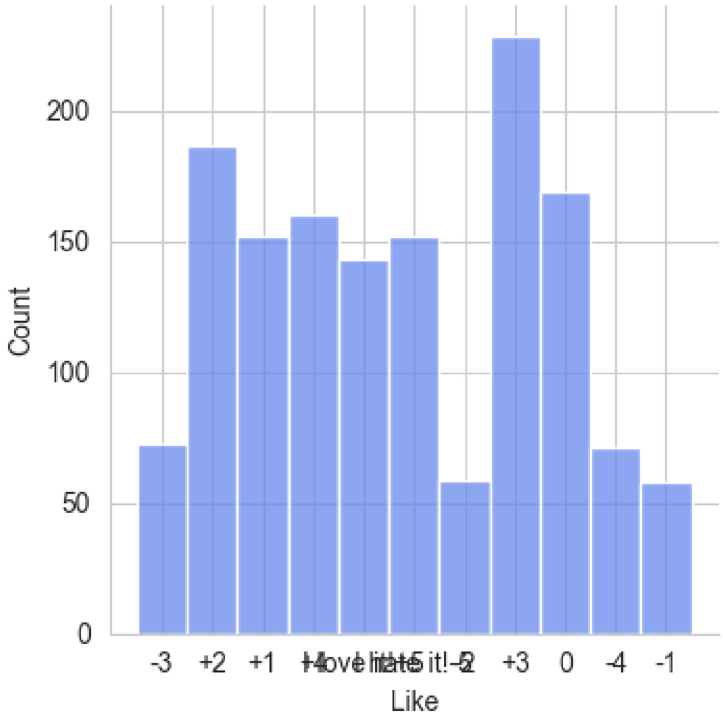

Count
1200
1000
800
600
400
200
0
Yes
No
fattening

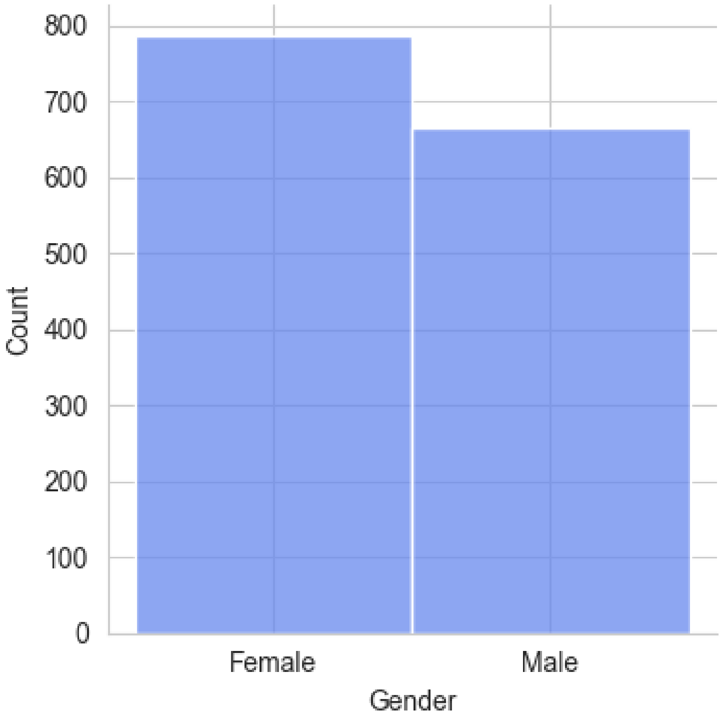

Count
1200
1000
800
600
400
200
0
Yes
No
fast


Count
0
200
400
600
800
No
Yes
tasty


Count
0
200
400
600
800
Yes
No
expensive


Count
1000
800
600
400
200
0
No
Yes
disgusting


Count
200
150
100
50
0
-3
+2
+1
+4
I love it!
+5
I hate it!
-2
+3
0
-4
-1
Like



Data Preprocessing

[14]:
```
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
```

[15]:
```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
for i in df.columns:
    if i != 'Age':
        df[i] = label_encoder.fit_transform(df[i])
```

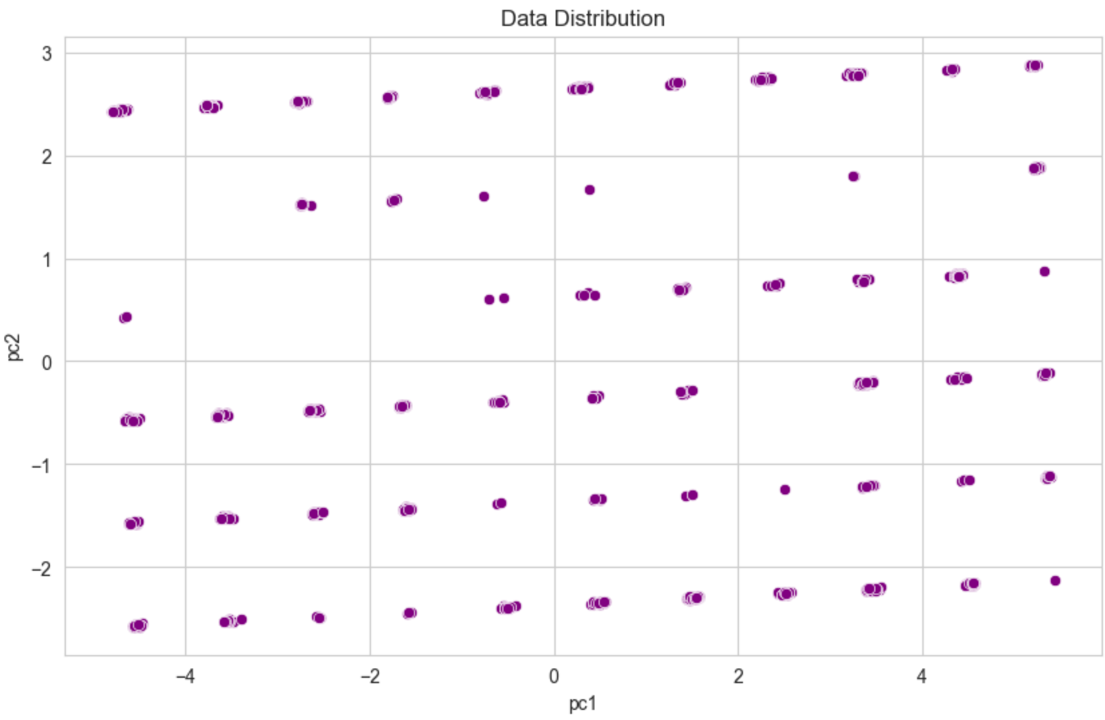
Observations * yummy is correlated with like and tasty * expensive with cheap * like is correlated with visitfrequency

Extract Segments

[16]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=14)
data = pca.fit_transform(df.drop(['Age'], axis=1))
pc = pd.DataFrame(data=data, columns=['pc1', 'pc2', 'pc3', 'pc4', 'pc5', 'pc6',␣
 ↪'pc7', 'pc8', 'pc9', 'pc10', 'pc11', 'pc12', 'pc13', 'pc14'])
```

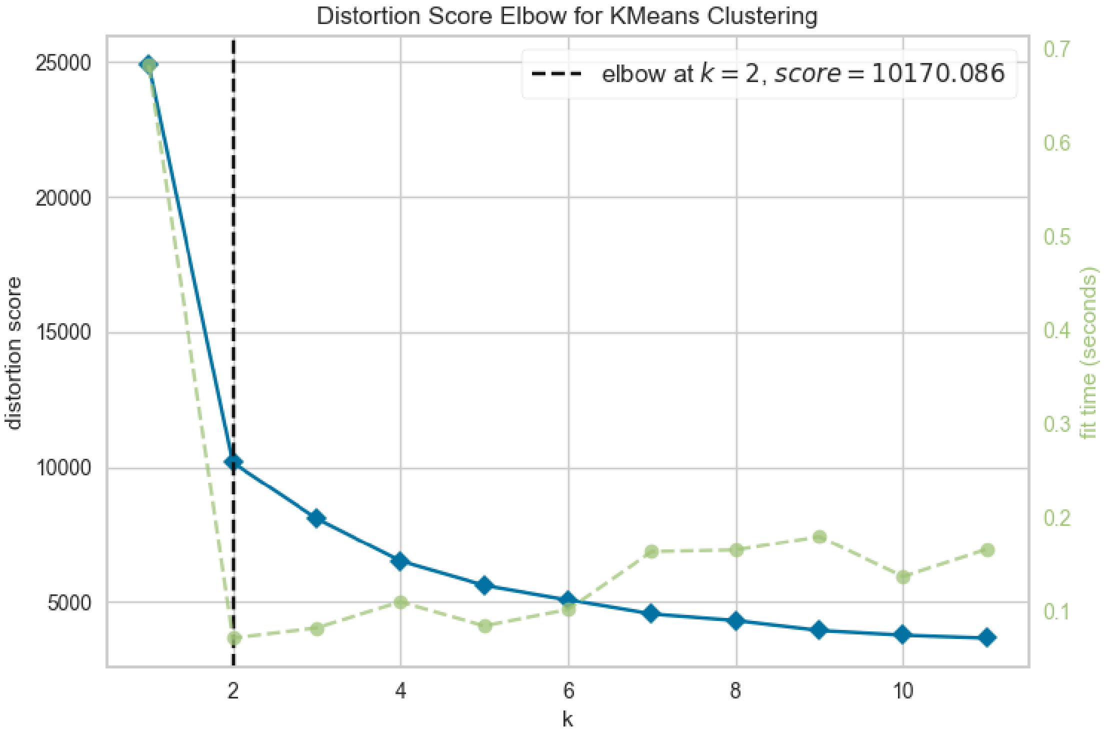
[17]:
```
plt.figure(figsize=(10,6))
sns.scatterplot(data=pc, x='pc1', y='pc2', color='purple')
plt.title('Data Distribution')
```

[17]:
```
Text(0.5, 1.0, 'Data Distribution')
```



Observation * Choosing 3 as the value of 'k' for clustering.

[18]:
```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
kmeans = KMeans()
visualizer = KElbowVisualizer(kmeans, k=(1, 12)).fit(pc)
visualizer.show()
```



[18]:
```
<Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'},
xlabel='k', ylabel='distortion score'>
```

[19]:
```
kmeans = KMeans(n_clusters=3)
kmeans.fit(pc)
```

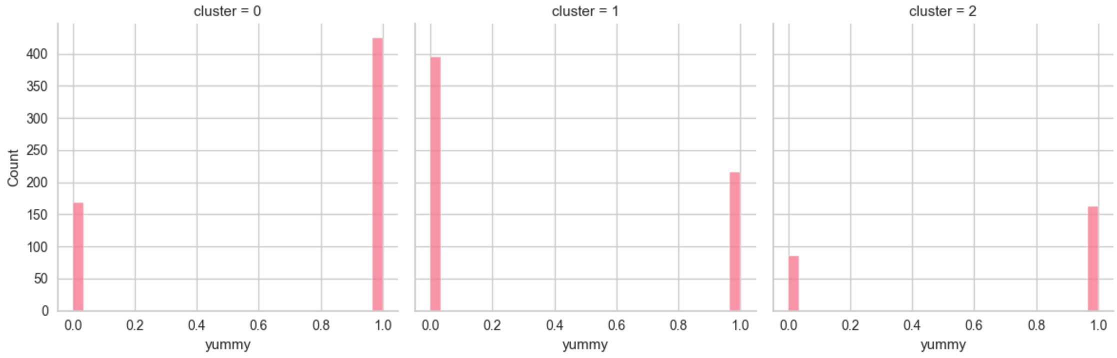
[19]:
```
KMeans(n_clusters=3)
```

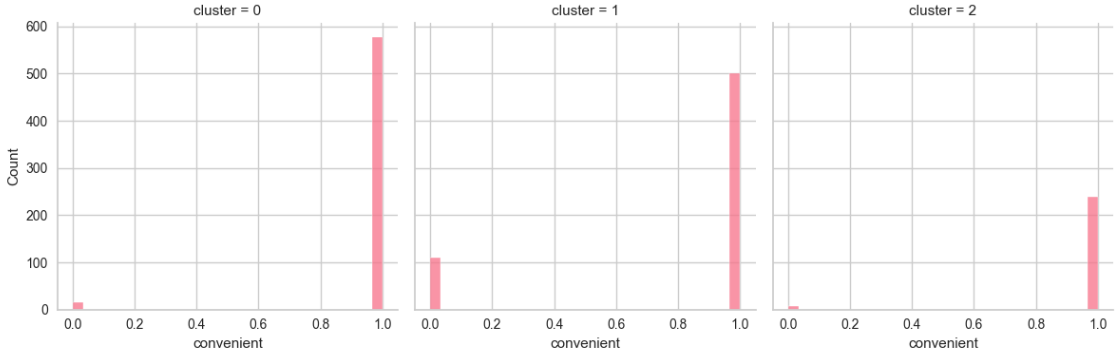
[20]:
```
np.random.seed(42)
preds = kmeans.predict(pc)
```

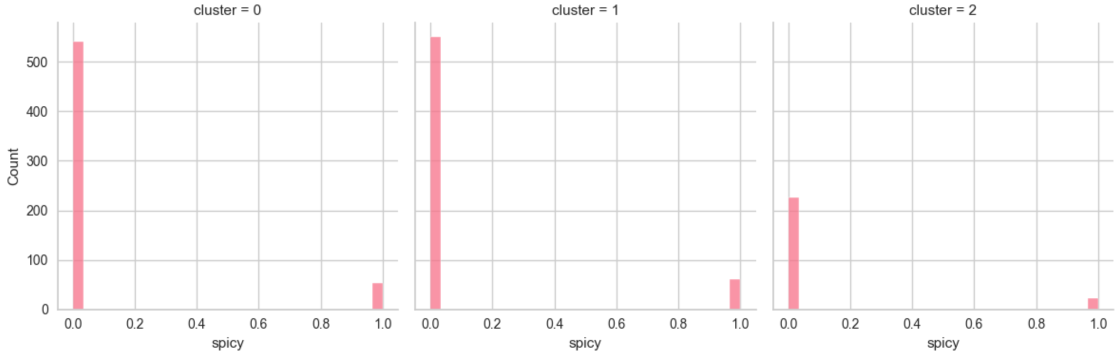
Observations * Maximum customers belong to cluster 0. * Approximately 25 percent of the customers are in cluster 1.

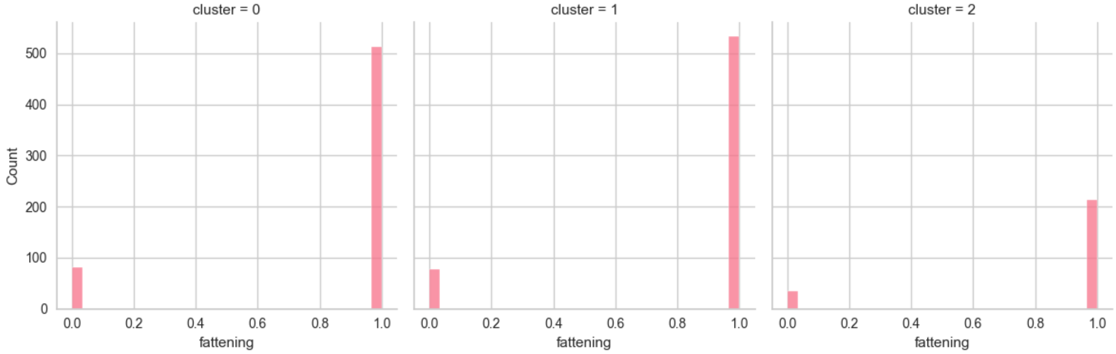
Observations * Most customers belong to cluster 0. * Approximately 25% of customers are categorized into cluster 1.

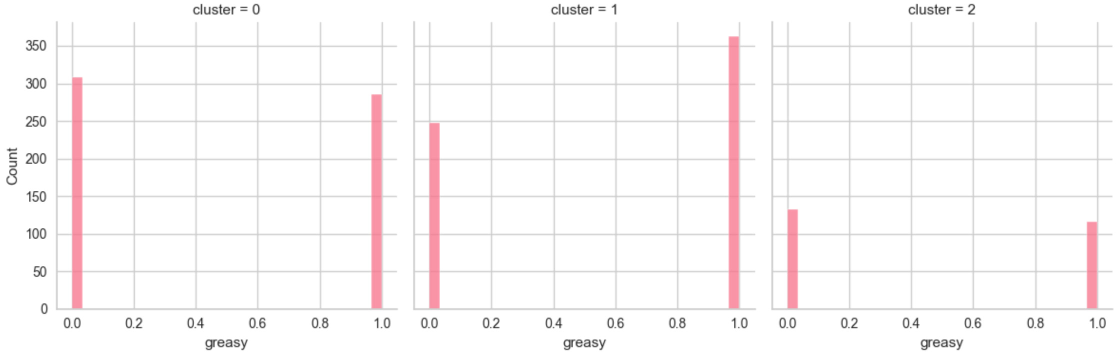
```
[57]: sns.set_palette('husl')
      for i in df.drop(['cluster'], axis=1):
          grid = sns.FacetGrid(df, height=4, col='cluster')
          grid = grid.map(sns.histplot, i, bins=30)
```

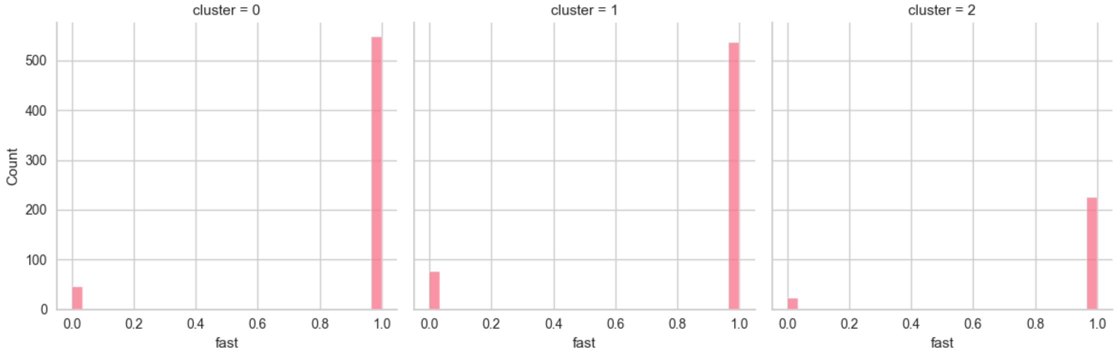


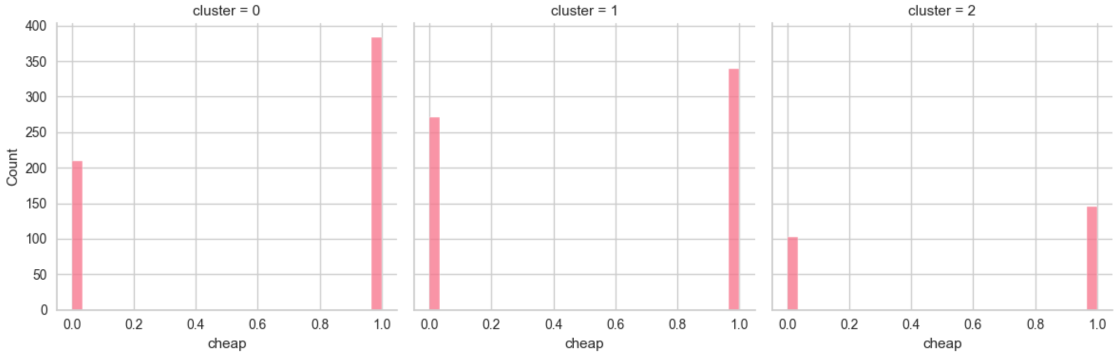


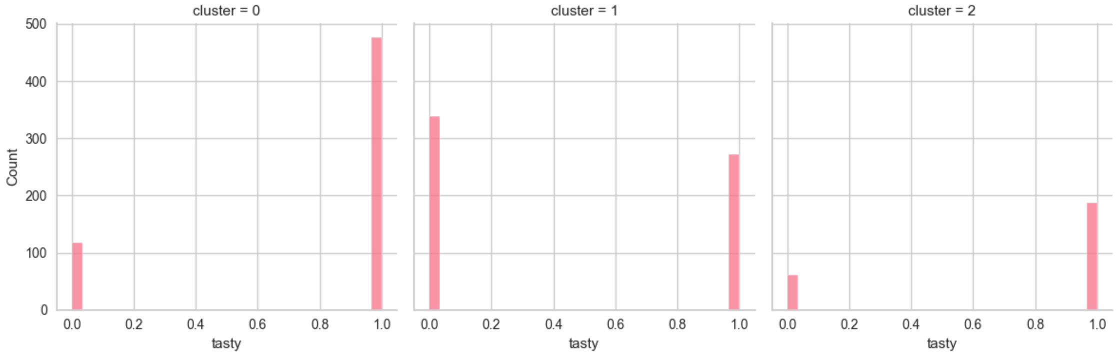


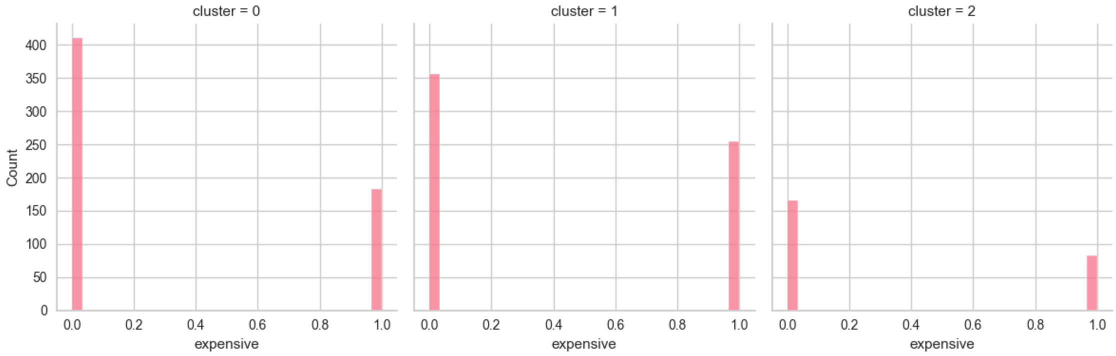

cluster = 0
cluster = 1
cluster = 2
Count
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
fattening
fattening
fattening

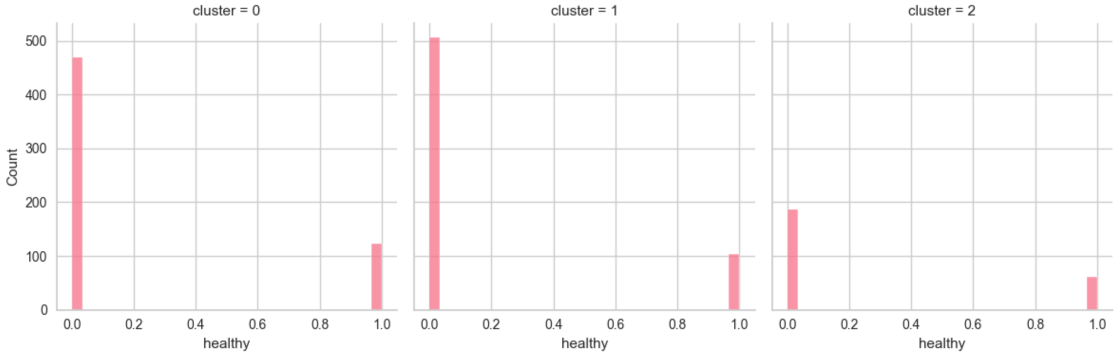

cluster = 0
cluster = 1
cluster = 2
Count
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0
greasy
greasy
greasy

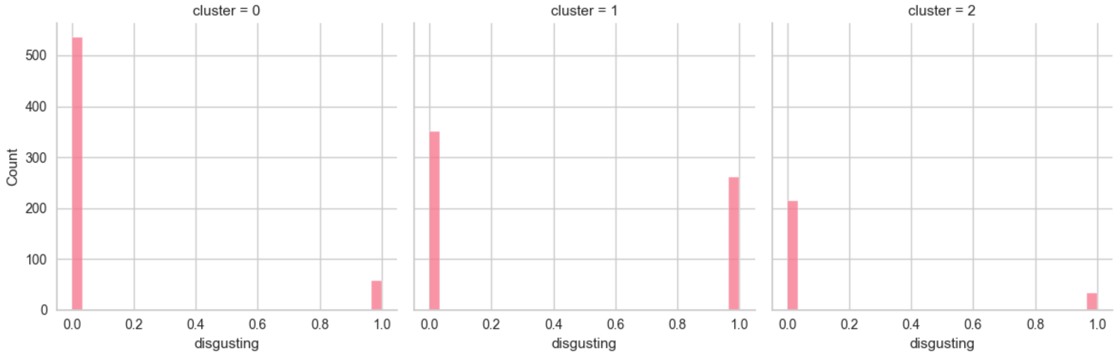

cluster = 0
cluster = 1
cluster = 2
Count
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
fast
fast
fast

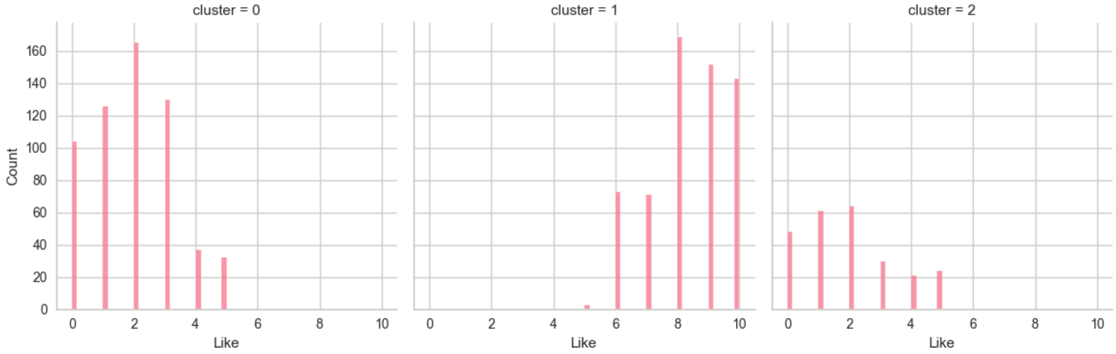

cluster = 0
cluster = 1
cluster = 2
Count
0
50
100
150
200
250
300
350
400
0.0
0.2
0.4
0.6
0.8
1.0
cheap
cheap
cheap

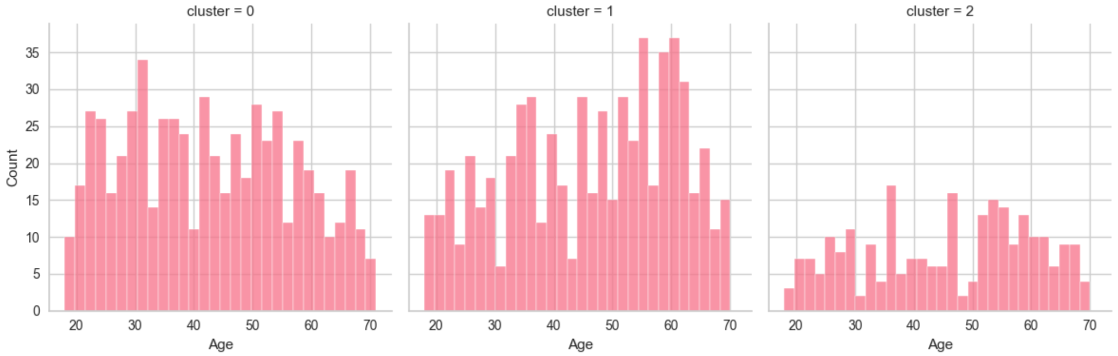

cluster = 0
cluster = 1
cluster = 2
Count
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
tasty
tasty
tasty

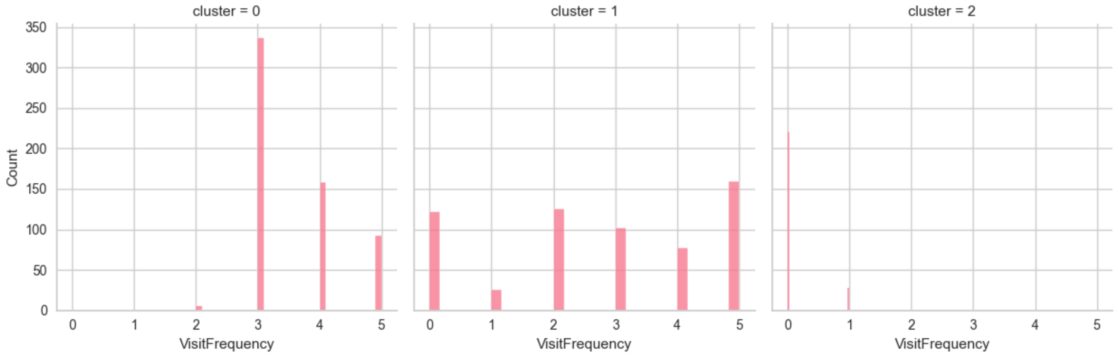

cluster = 0
cluster = 1
cluster = 2
Count
0
50
100
150
200
250
300
350
400
0.0
0.2
0.4
0.6
0.8
1.0
expensive
expensive
expensive

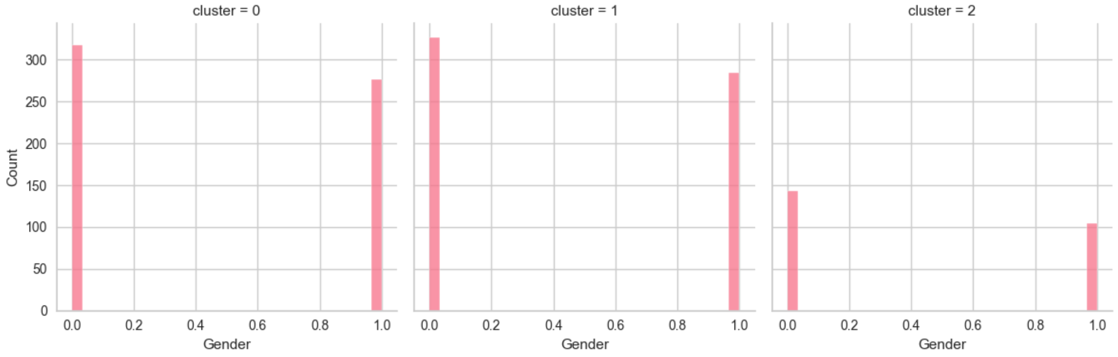

cluster = 0
cluster = 1
cluster = 2
Count
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
healthy
healthy
healthy


cluster = 0
cluster = 1
cluster = 2
Count
0
100
200
300
400
500
0.0
0.2
0.4
0.6
0.8
1.0
disgusting
disgusting
disgusting


cluster = 0
cluster = 1
cluster = 2
Count
0
20
40
60
80
100
120
140
160
0
2
4
6
8
10
Like
Like
Like



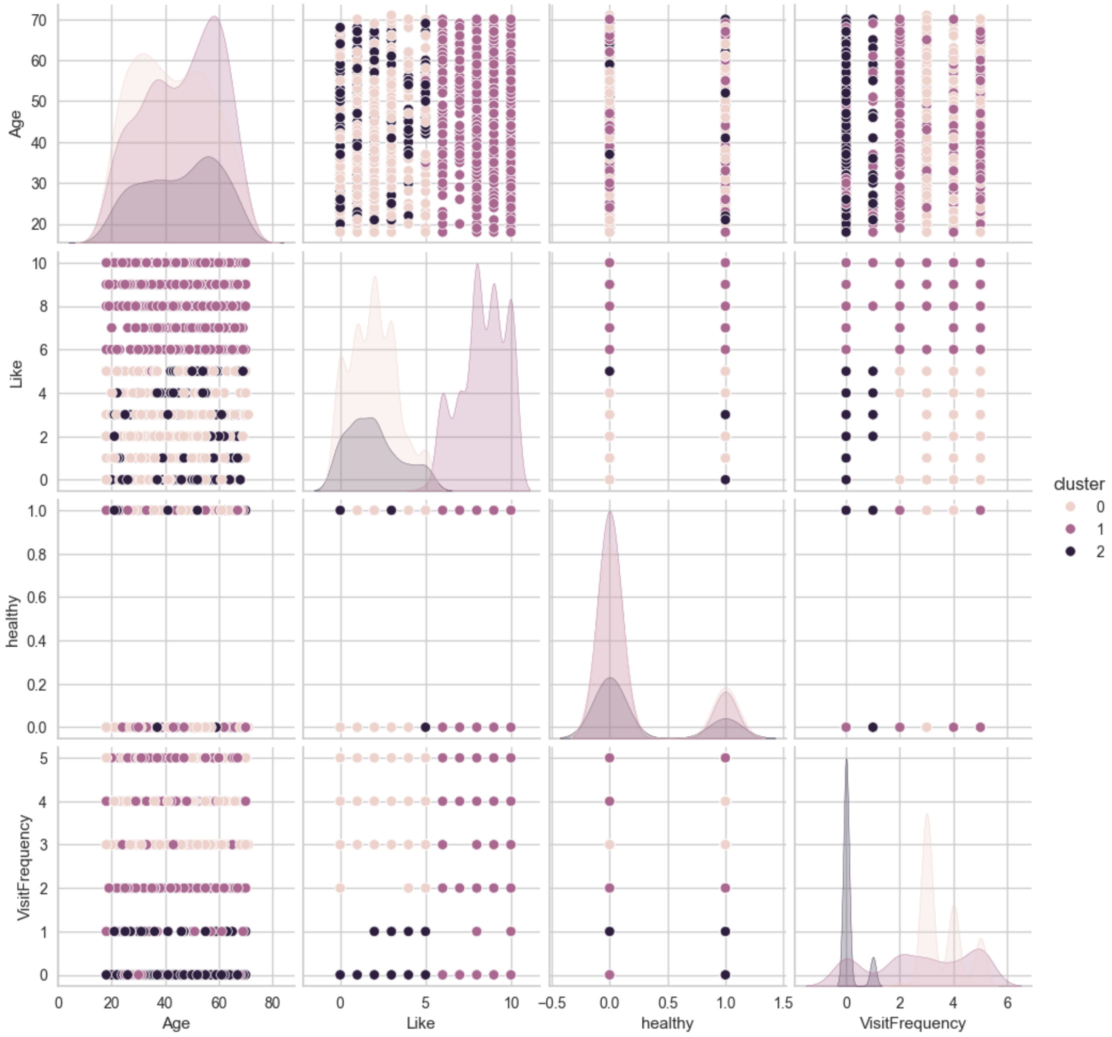




Observations * cluster 0 contains most of the customers who voted for not yummy where as in cluster 1 customers mostly voted yummy * same is for tasty, cluster 0 customers almost doesn't find the food tasty * customers belonging to cluster 1 doesn't find the food convienent * Like is distributed with in intervals > * Like -5 to -2 belongs to cluster 0 > * +2 to +5 belongs to cluster 1 >* -2 to +2 belongs to cluster 2 * cluster 0 doesn't contain customers visited more than once in a month * cluster 1 does not contain who have never visited the store * most of the customers of cluster 2 have not visited more than once in a week

```
[50]: df_1 = df[['Age', 'Like', 'cluster','healthy','VisitFrequency']]
      sns.pairplot(data=df_1, hue='cluster')
```

```
[50]: <seaborn.axisgrid.PairGrid at 0x2616b816790>
```



Classification

```
[25]: from sklearn.model_selection import train_test_split

      x = df.drop(['cluster'], axis=1)
      y = df['cluster']

      x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
       ↪random_state=42, stratify=y)
```

[26]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

[27]:
```
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(x_train, y_train)
preds = clf.predict(x_test)
```

[59]:
```
from sklearn.ensemble import GradientBoostingClassifier
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
    max_depth=1, random_state=0).fit(x_train, y_train)
clf.score(x_test, y_test)
xgboostPred = clf.predict(x_test)
```

[28]:
```
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪ConfusionMatrixDisplay

print(classification_report(y_test, preds))
```

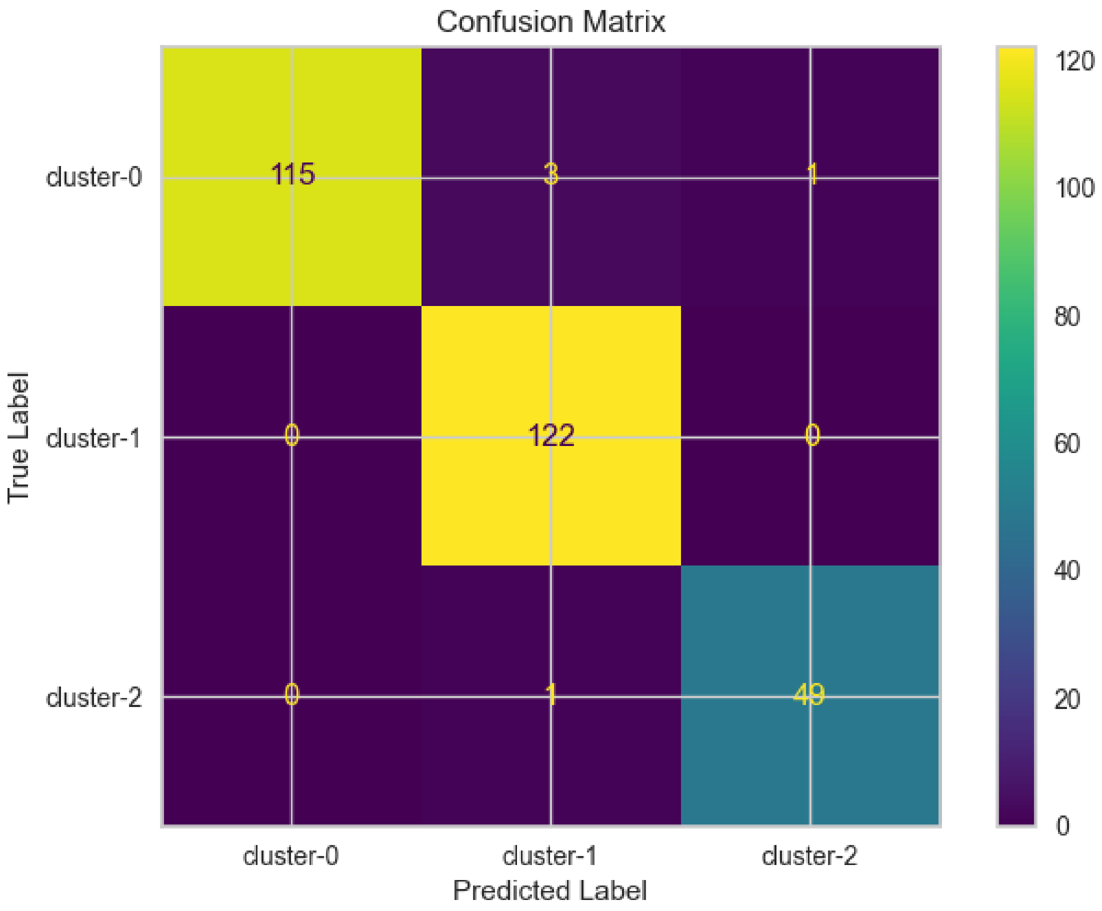
```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98       119
           1       0.97      1.00      0.98       122
           2       0.98      0.98      0.98        50

    accuracy                           0.98       291
   macro avg       0.98      0.98      0.98       291
weighted avg       0.98      0.98      0.98       291
```

[60]:
```
print(classification_report(y_test, xgboostPred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       119
           1       0.98      0.99      0.98       122
           2       1.00      1.00      1.00        50

    accuracy                           0.99       291
   macro avg       0.99      0.99      0.99       291
weighted avg       0.99      0.99      0.99       291
```

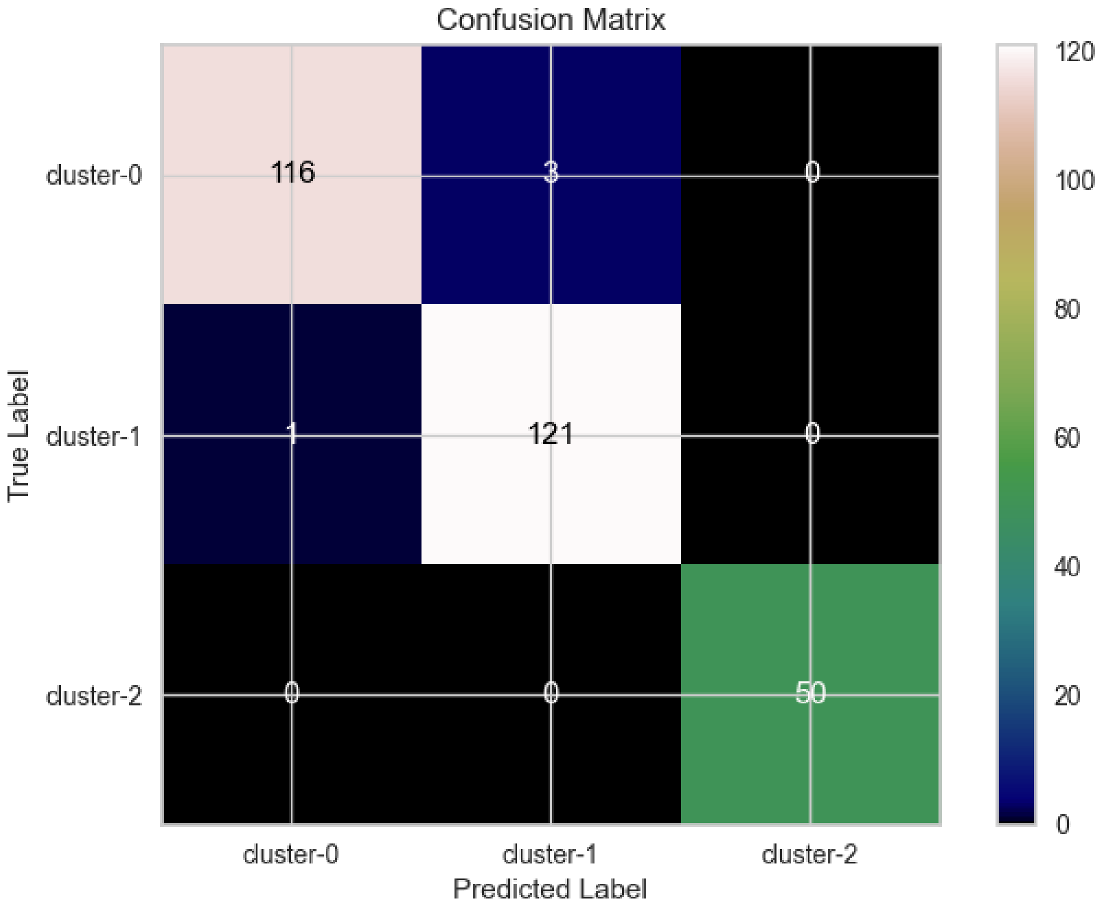
```
[52]: cm = confusion_matrix(y_test, preds, labels=[0, 1, 2])
      disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["cluster-0",␣
        ↪"cluster-1", 'cluster-2'])
      disp.plot(colorbar=True)
      plt.xlabel('Predicted Label')
      plt.ylabel('True Label')
      plt.title('Confusion Matrix')
```

```
[52]: Text(0.5, 1.0, 'Confusion Matrix')
```



```
[66]: cm = confusion_matrix(y_test, xgboostPred, labels=[0, 1, 2])
      disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=["cluster-0",␣
        ↪"cluster-1", 'cluster-2'])
      disp.plot(cmap='gist_earth',colorbar=True)
      plt.xlabel('Predicted Label')
      plt.ylabel('True Label')
      plt.title('Confusion Matrix')
```

```
[66]: Text(0.5, 1.0, 'Confusion Matrix')
```



Observations * The classification model performs well, showing no signs of parameter tuning requirement. * xgboost Model performs well than any other Model With High Accuracy. So we can use this Model for Market Segmentation(Case Study Mcdonalds dataset)