Market Segmentation Analysis-Copy1

June 18, 2022

[1]:

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

1 Import the dataset

[2]:

```
dataset = pd.read_csv('C:\\Users\DELL\\Desktop\\Feynn Lab\\mcdonalds.csv')
```

[3]:

```
dataset.head()
```

[3]:

```
  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
```

2 Exploratory data analysis

[4]:

```
dataset.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
```

[5]:
```
dataset.shape
```

[5]:
```
(1453, 15)
```

[6]:
```
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
```

[7]:
```
dataset.columns
```

[7]:
```
Index(['yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap',
       'tasty', 'expensive', 'healthy', 'disgusting', 'Like', 'Age',
       'VisitFrequency', 'Gender'],
      dtype='object')
```

2.1 now convert all catagorical value into integer

[8]:
```
dataset['yummy']=dataset['yummy'].map({'No':0,'Yes':1})
dataset['convenient']=dataset['convenient'].map({'No':0,'Yes':1})
dataset['spicy']=dataset['spicy'].map({'No':0,'Yes':1})
dataset['fattening']=dataset['fattening'].map({'No':0,'Yes':1})
dataset['greasy']=dataset['greasy'].map({'No':0,'Yes':1})
dataset['fast']=dataset['fast'].map({'No':0,'Yes':1})
dataset['cheap']=dataset['cheap'].map({'No':0,'Yes':1})
dataset['tasty']=dataset['tasty'].map({'No':0,'Yes':1})
dataset['expensive']=dataset['expensive'].map({'No':0,'Yes':1})
dataset['healthy']=dataset['healthy'].map({'No':0,'Yes':1})
dataset['disgusting']=dataset['disgusting'].map({'No':0,'Yes':1})
dataset['Gender']=dataset['Gender'].map({'Female':0,'Male':1})
```

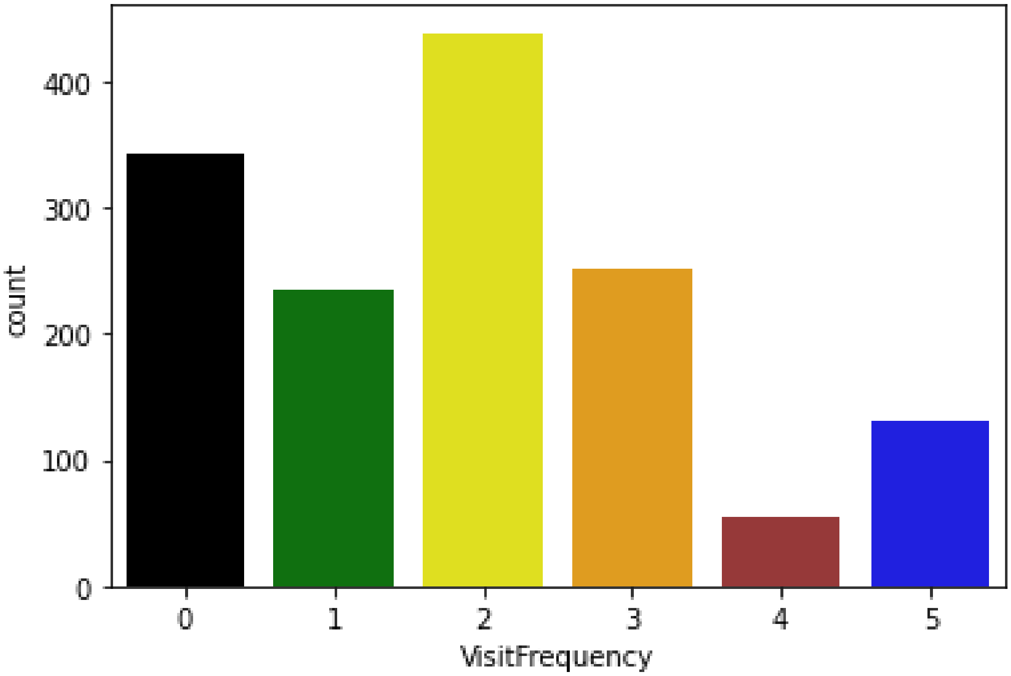
[9]:
```
dataset['VisitFrequency'].unique()
```

[9]:
```
array(['Every three months', 'Once a week', 'Once a month', 'Once a year',
       'More than once a week', 'Never'], dtype=object)
```

[10]:
```
dataset['VisitFrequency']=dataset['VisitFrequency'].map({'Every three months':
 ↪0,'Once a week':1, 'Once a month': 2,
                                                         'Once a year': 3, 'More␣
 ↪than once a week':4, 'Never':5})
```

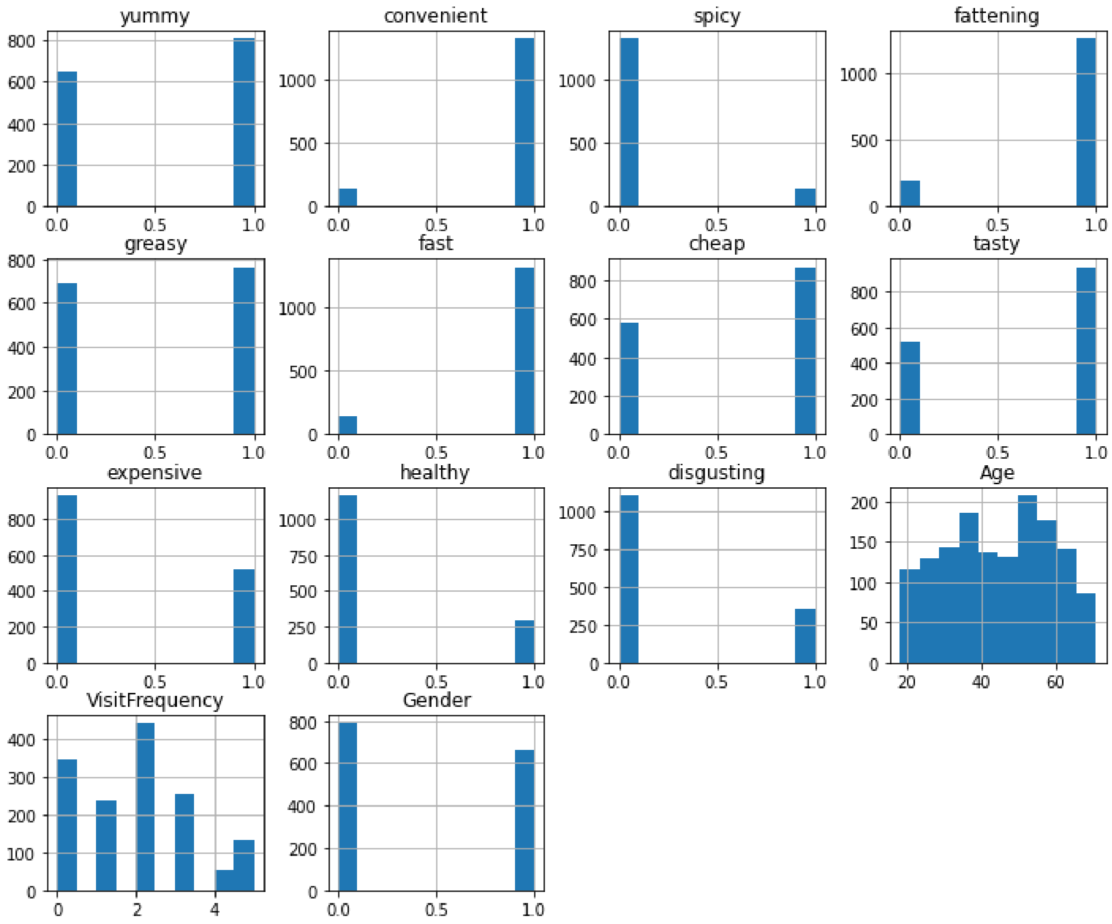
[11]:
```
sns.countplot(x='VisitFrequency' , data= dataset, palette=['black', 'green',␣
 ↪'yellow', 'orange', 'brown','blue'] )
```

[11]:
```
<AxesSubplot:xlabel='VisitFrequency', ylabel='count'>
```



```
[12]: dataset.hist(figsize= [12,10])
```

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



2.2 'Like' columns not require, So drop the column

[13]:
```
dataset.drop('Like', axis = 1, inplace=True)
```

[14]:
```
dataset.head()
```

[14]:
```
   yummy  convenient  spicy  fattening  greasy  fast  cheap  tasty  expensive  \
0      0           1      0          1       0     1      1      0          1
1      1           1      0          1       1     1      1      1          1
2      0           1      1          1       1     1      0      1          1
3      1           1      0          1       1     1      1      1          0
4      0           1      0          1       1     1      1      0          0

   healthy  disgusting  Age  VisitFrequency  Gender
0        0           0   61               0       0
1        0           0   51               0       0
2        1           0   62               0       0
3        0           1   69               1       0
```

```
4        1           0   49                2       1
```

[15]:
```python
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1453 entries, 0 to 1452
Data columns (total 14 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   yummy           1453 non-null   int64
 1   convenient      1453 non-null   int64
 2   spicy           1453 non-null   int64
 3   fattening       1453 non-null   int64
 4   greasy          1453 non-null   int64
 5   fast            1453 non-null   int64
 6   cheap           1453 non-null   int64
 7   tasty           1453 non-null   int64
 8   expensive       1453 non-null   int64
 9   healthy         1453 non-null   int64
 10  disgusting      1453 non-null   int64
 11  Age             1453 non-null   int64
 12  VisitFrequency  1453 non-null   int64
 13  Gender          1453 non-null   int64
dtypes: int64(14)
memory usage: 159.0 KB
```

3 now all the dataset are integer and non- null value

[16]:
```python
dataset.insert(0, 'Gender', dataset.pop('Gender'))
```

[17]:
```python
dataset.head()
```

[17]:
```
   Gender  yummy  convenient  spicy  fattening  greasy  fast  cheap  tasty  \
0       0      0           1      0          1       0     1      1      0
1       0      1           1      0          1       1     1      1      1
2       0      0           1      1          1       1     1      0      1
3       0      1           1      0          1       1     1      1      1
4       1      0           1      0          1       1     1      1      0

   expensive  healthy  disgusting  Age  VisitFrequency
0          1        0           0   61               0
1          1        0           0   51               0
2          1        1           0   62               0
3          0        0           1   69               1
4          0        1           0   49               2
```

```
[18]: X = dataset.iloc[:, :-1]
      y = dataset.iloc[:, -1]
```

```
[19]: X.head()
```

```
[19]:    Gender  yummy  convenient  spicy  fattening  greasy  fast  cheap  tasty  \
      0       0      0           1      0          1       0     1      1      0
      1       0      1           1      0          1       1     1      1      1
      2       0      0           1      1          1       1     1      0      1
      3       0      1           1      0          1       1     1      1      1
      4       1      0           1      0          1       1     1      1      0

         expensive  healthy  disgusting  Age
      0          1        0           0   61
      1          1        0           0   51
      2          1        1           0   62
      3          0        0           1   69
      4          0        1           0   49
```

```
[20]: y
```

```
[20]: 0       0
      1       0
      2       0
      3       1
      4       2
             ..
      1448    3
      1449    1
      1450    2
      1451    0
      1452    0
      Name: VisitFrequency, Length: 1453, dtype: int64
```

4 Spliting the dataset

```
[21]: from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,␣
        ↪random_state=1)
```

5 Feature Scaling

[22]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

[23]:
```
X_train, X_test
```

[23]:
```
(array([[ 1.08069906, -1.09199489,  0.32659863, …, -0.50080667,
          1.71434625, -0.33249598],
        [ 1.08069906,  0.9157552 ,  0.32659863, …, -0.50080667,
         -0.58331274,  0.86738081],
        [ 1.08069906,  0.9157552 ,  0.32659863, …,  1.99677852,
          1.71434625, -0.26191499],
        …,
        [-0.92532699,  0.9157552 ,  0.32659863, …,  1.99677852,
         -0.58331274, -0.12075301],
        [-0.92532699, -1.09199489,  0.32659863, …, -0.50080667,
         -0.58331274,  1.50260969],
        [-0.92532699, -1.09199489,  0.32659863, …, -0.50080667,
          1.71434625,  0.30273291]]),
 array([[ 1.08069906,  0.9157552 ,  0.32659863, …, -0.50080667,
         -0.58331274, -1.03830585],
        [-0.92532699, -1.09199489,  0.32659863, …, -0.50080667,
         -0.58331274, -0.47365795],
        [ 1.08069906, -1.09199489,  0.32659863, …,  1.99677852,
         -0.58331274,  1.64377167],
        …,
        [-0.92532699, -1.09199489,  0.32659863, …, -0.50080667,
         -0.58331274,  0.09098995],
        [ 1.08069906,  0.9157552 ,  0.32659863, …, -0.50080667,
         -0.58331274, -0.61481993],
        [ 1.08069906,  0.9157552 ,  0.32659863, …, -0.50080667,
         -0.58331274, -0.33249598]]))
```

6 Training the Decision Tree Classification model on the Training set

[24]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
classifier.fit(X_train, y_train)
```

[24]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

# Predicting the Test set results

[25]:
```
y_pred = classifier.predict(X_test)
```

7 Making the Confusion Matrix

[27]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[17 13 21 10  3  2]
 [12 10 20  5  2  0]
 [22 28 21  9  0  4]
 [15  4 12 13  1  8]
 [ 4  4  5  0  2  0]
 [ 6  3  3  5  0  7]]
```

[27]:
```
0.24054982817869416
```

8 Training the Random Forest Classification model on the Training dataset

[28]:
```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 10, criterion = 'entropy',␣
  ↪random_state = 0)
classifier.fit(X_train, y_train)
```

[28]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10, random_state=0)
```

[29]:
```
y_pred = classifier.predict(X_test)
```

[30]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[13 12 25 10  3  3]
 [10  8 24  3  2  2]
 [22 19 28  9  3  3]
 [14  4 18  7  0 10]
 [ 2  6  4  2  1  0]
 [ 9  1  3  6  0  5]]
```

[30]:
```
0.21305841924398625
```

9 Training the K-NN model on the Training set

[31]:

```
from sklearn.neighbors import KNeighborsClassifier
k_classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p =␣
  ↪2)
k_classifier.fit(X_train, y_train)
```

[31]:

```
KNeighborsClassifier()
```

[32]:

```
y_pred = k_classifier.predict(X_test)
```

[33]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[27 10 19  6  1  3]
 [12  6 25  5  1  0]
 [30 16 29  6  1  2]
 [19  1 14 14  0  5]
 [ 3  4  8  0  0  0]
 [ 8  1  2  9  0  4]]
```

[33]:

```
0.27491408934707906
```

10 Training the Logistic Regression model on the Training set

[34]:

```
from sklearn.linear_model import LogisticRegression
log_classifier = LogisticRegression(random_state = 0)
log_classifier.fit(X_train, y_train)
```

[34]:

```
LogisticRegression(random_state=0)
```

[35]:

```
y_pred = log_classifier.predict(X_test)
```

[36]:

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
print(cm)
accuracy_score(y_test, y_pred)
```

```
[[12  1 42  9  0  2]
 [ 8  2 37  2  0  0]
 [ 8  2 66  7  0  1]
 [17  0 17 16  0  3]
 [ 0  1 14  0  0  0]
 [ 3  0  3  8  0 10]]
```

```
[36]: 0.3642611683848797
```

```
[ ]:
```