# Market Segmentation Analysis using ML

# Introduction

## Project Overview

In this project, we collect data of fast food restaurant reviews and predict the age group of the customer based on the data using machine learning algorithms.

## Purpose

The purpose of this project is to match the genuine needs and desires of consumers with the offers of suppliers particularly suited to satisfy those needs and desires. This matching process benefits consumers and suppliers, and drives an organization’s marketing planning process.

# Literature Survey

## Existing Problem

The Market Segmentation Analysis Using ML aims to analyze the spending behaviour of customers and identify opportunities for growth.

## References

Market Segmentation is the process of dividing customers into groups based on their shared characteristics, such as spending habits, location, or industry. This can be a valuable tool for wholesale businesses to better understand their customers and tailor their marketing and sales strategies accordingly.

There is a growing body of literature on wholesale customer segmentation. A 2019 study by the Aberdeen Group found that businesses that use customer segmentation are more likely to achieve their revenue and profit goals than those that do not. The study also found that businesses that use customer segmentation are better able to:

- Target their marketing campaigns more effectively
- Develop products and services that meet the needs of their customers
- Increase customer satisfaction and retention

There are a number of different ways to segmentcustomers. Some common methods include:

• **Geographic segmentation:** This involves dividing customers into groups based on their location. This can be a useful way to target customers with local marketing campaigns or to tailor product offerings to meet the needs of customers in different regions.

• **Demographic segmentation:** This involves dividing customers into groups based on their age, gender, income, or other demographic characteristics. This can be a useful way to target customers with specific products or services.

• **Behavioral segmentation:** This involves dividing customers into groups based on their buying habits, such as the products they purchase, the frequency of their purchases, or the amount they spend. This can be a useful way to identify customers who are most likely to respond to a particular marketing campaign or to develop new products or services that meet the needs of these customers.

Wholesale customer segmentation can be a valuable tool for businesses of all sizes. By understanding their customers and their needs, businesses can better tailor their marketing and sales strategies to achieve their goals.

## Problem Statement Definition

The Market Segmentation Analysis Using ML aims to analyze the spending behaviour of customers and identify opportunities for growth. The data set consists of opinions and characteristics ('yummy', 'convenient', 'spicy', 'fattening', 'greasy', 'fast', 'cheap', 'tasty', 'expensive', 'healthy', 'disgusting', 'Like', 'VisitFrequency', 'Gender') of the customer.

By using k-means clustering and logistic regression, we predict the age group of the customer. Basically, we are performing demographic segmentation on our audience.

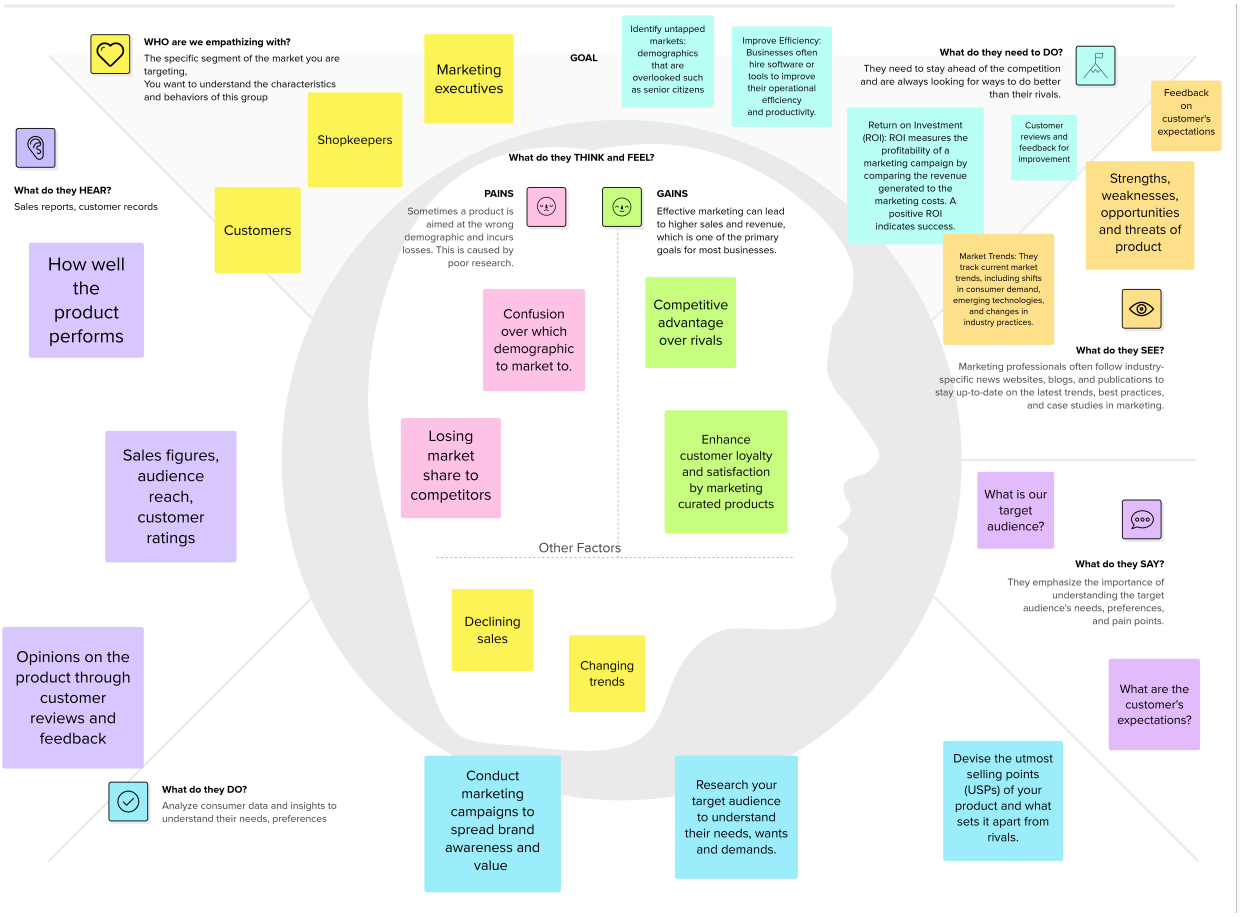
The project seeks to group customers with similar spending patterns together. By identifying customer segments with distinct spending behaviours, the project aims to provide insights on how businesses can tailor their marketing strategies and product offerings to better serve each customer segment. The project also aims to identify opportunities for growth, such as which

products or product categories are underrepresented among customers, and which segments may be receptive to new product offerings.

Overall, the project seeks to provide valuable insights for wholesale businesses on how to optimize their operations and increase customer satisfaction and retention.

# Ideation and Proposed Solution

## Empathy Map Canvas



# Ideation and Brainstorming

1

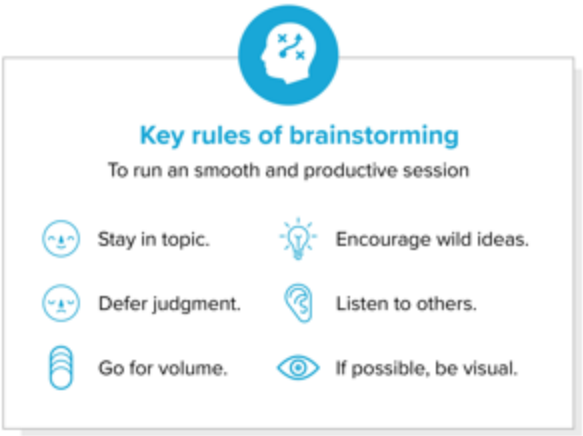
## Define your problem statement

The Market Segmentation Analysis Using ML aims to analyze the spending behaviour of customers and identify opportunities for growth

5 minutes

PROBLEM

**The Market Segmentation Analysis Using ML aims to analyze the spending behaviour of customers and identify opportunities for growth**



## 2 Brainstorm

Write down any ideas that come to mind that address your problem statement.

⏱ 10 minutes

**TIP**
You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

### Snvithi

Group customers into demographics based on shared characteristics like age and gender

Segment customers based on the frequency of spending.

Provide discounts and offers for customers who spend frequently to build brand loyalty.

### Pranav

Social Media Monitoring: Analyze social media data to understand how customer sentiment and social interactions influence spending behavior

Competitor Analysis: Use ML to gather and analyze data on competitor pricing, promotions, and customer behavior to find gaps and opportunities.

Dynamic Pricing: Implement dynamic pricing strategies using ML to adjust prices based on real-time demand and customer behavior.

Customer Lifetime Value (CLV) Prediction: Predict CLV for individual customers using ML. This can guide marketing and sales efforts towards high-value customers.

Churn Prediction: Use ML to predict customer churn. Identifying customers at risk of leaving can help businesses take proactive measures to retain the m.

Collaborative Filtering: Collaborative filtering algorithms can be employed to identify patterns in spending behavior based on the behavior of similar customers.

Customer Segmentation: Utilize ML algorithms to segment your customer base based on their spending behavior..

Anomaly Detection: Employ anomaly detection algorithms to identify irregular spending patterns. This can help in fraud detection or recognizing sudden changes in customer behavior that might indicate new opportunities.

Predictive Analytics: Develop predictive models using ML to forecast future spending behavior. This can help businesses anticipate customer needs and tailor marketing strategies accordingly.

### Prashant

Cluster customers based on their spending behavior. Consider features such as purchase history, demographics, and social media activity.

Develop targeted marketing campaigns and product offerings for each segment,

Develop strategies to retain at-risk customers, such as offering them discounts, personalized recommendations, or other incentives.

Recommend products and services to customers and increase sales.

Use machine learning algorithms to identify trends in customer spending and use this info. to identify new markets for growth.

Predict how likely each customer is to churn, based on their past behavior and other factors.

### Vivek

Use machine learning algorithms to identify trends in customer spending and use this info. to identify new markets for growth.

ML algorithms can analyze historical sales data and external factors to forecast future demand accurately, enabling businesses to optimize inventory management and supply chain operations.

Use ML algorithms to identify potential cross-selling and upselling opportunities by analyzing customer behavior and purchase history to suggest relevant products or services.

Use machine learning algorithms to understand the needs and wants of your customers. You can develop products and services that they are more likely to purchase.

ML can help identify high-value customers who contribute the most to revenue and profit, allowing businesses to focus on building strong relationships with them.

machine learning algorithms can be used to develop targeted marketing campaigns and product development initiatives to identify popular products and services by analyzing customer spending data.

ML help in customers behave and what they prefer, you can find and fix issues in their experience with your company. Making these improvements can lead to happier customers who are more likely to stick around.

## 3 Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

⏱ 20 minutes

**TIP**
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

### Segmentation

Customer Segmentation: Utilize ML algorithms to segment your customer base based on their spending behavior..

ML can help identify high-value customers who contribute the most to revenue and profit, allowing businesses to focus on building strong relationships with them.

Group customers into demographics based on shared characteristics like age and gender

Segment customers based on the frequency of spending.

Cluster customers based on their spending behavior. Consider features such as purchase history, demographics, and social media activity.

Develop targeted marketing campaigns and product offerings for each segment,

### Analysis

Social Media Monitoring: Analyze social media data to understand how customer sentiment and social interactions influence spending behavior

Anomaly Detection: Employ anomaly detection algorithms to identify irregular spending patterns. This can help in fraud detection or recognizing sudden changes in customer behavior that might indicate new opportunities.

Predictive Analytics: Develop predictive models using ML to forecast future spending behavior. This can help businesses anticipate customer needs and tailor marketing strategies accordingly.

ML algorithms can analyze historical sales data and external factors to forecast future demand accurately, enabling businesses to optimize inventory management and supply chain operations.

Dynamic Pricing: Implement dynamic pricing strategies using ML to adjust prices based on real-time demand and customer behavior.

Collaborative Filtering: Collaborative filtering algorithms can be employed to identify patterns in spending behavior based on the behavior of similar customers.

### Building brand value

Provide discounts and offers for customers who spend frequently to build brand loyalty.

Customer Lifetime Value (CLV) Prediction: Predict CLV for individual customers using ML. This can guide marketing and sales efforts towards high-value customers.

Recommend products and services to customers and increase sales.

ML help in customers behave and what they prefer, you can find and fix issues in their experience with your company. Making these improvements can lead to happier customers who are more likely to stick around.

ML can help identify high-value customers who contribute the most to revenue and profit, allowing businesses to focus on building strong relationships with them.

ML algorithms can be used to develop targeted marketing campaigns and product development initiatives to identify popular products and services by analyzing customer spending data.

### Risk management

Churn Prediction: Use ML to predict customer churn. Identifying customers at risk of leaving can help businesses take proactive measures to retain the m.

Predict how likely each customer is to churn, based on their past behavior and other factors.

Competitor Analysis: Use ML to gather and analyze data on competitor pricing, promotions, and customer behavior to find gaps and opportunities.

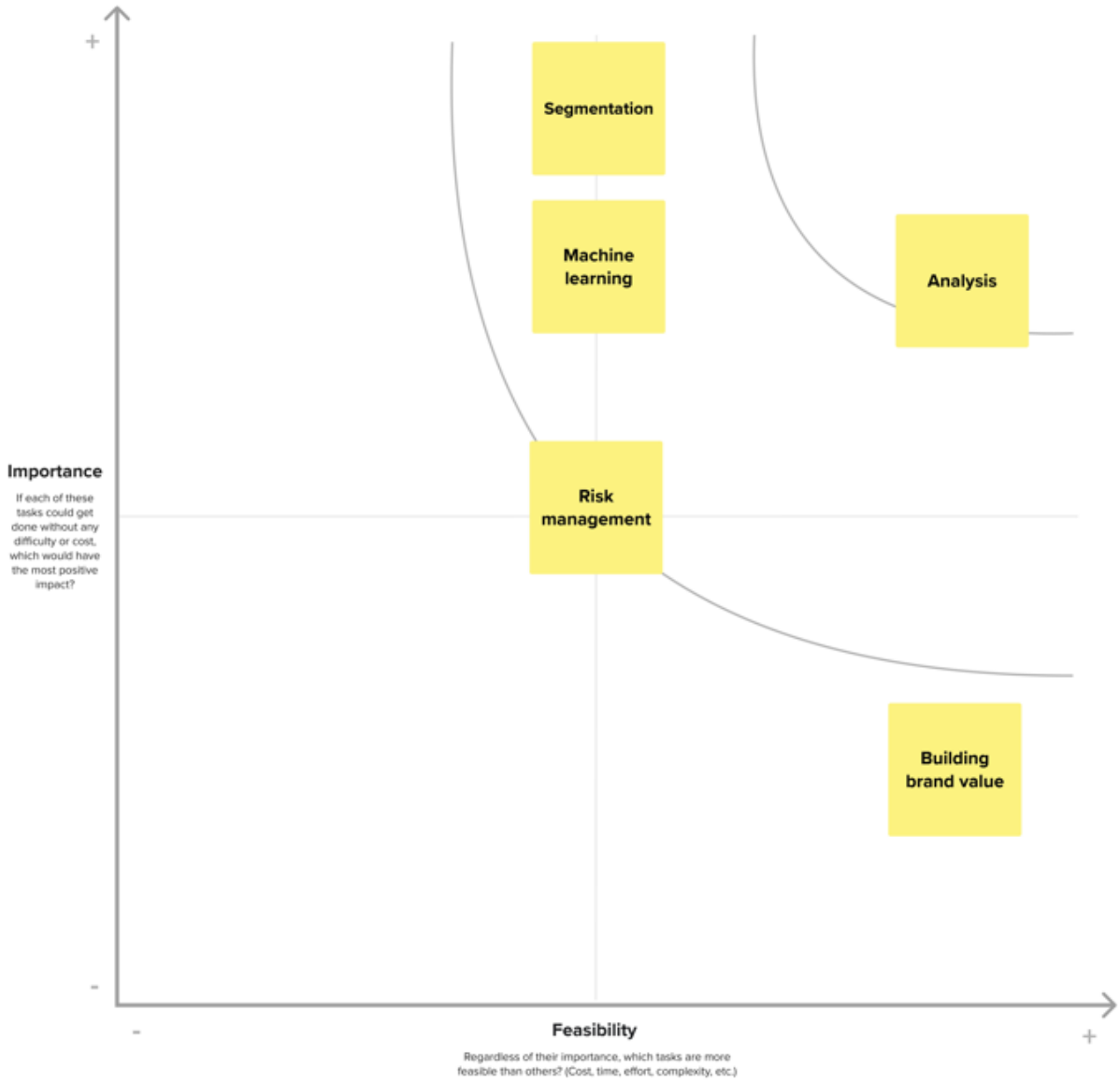
Develop strategies to retain at-risk customers, such as offering them discounts, personalized recommendations, or other incentives.

### Machine learning

Use ML algorithms to identify potential cross-selling and upselling opportunities by analyzing customer behavior and purchase history to suggest relevant products or services.

Use machine learning algorithms to understand the needs and wants of your customers. You can develop products and services that they are more likely to purchase.

Use machine learning algorithms to identify trends in customer spending and use this info. to identify new markets for growth.




# Requirements Analysis

## Functional Requirements

- Text values must be converted to numeric values before they are input to the ML model

- The output must display the correct age group which the predicted cluster represents.

- The application must indicate all of the accepted values and conditions for each input variable.
- If an invalid value is entered, the application should return an exception which informs the user of the correct values and conditions.

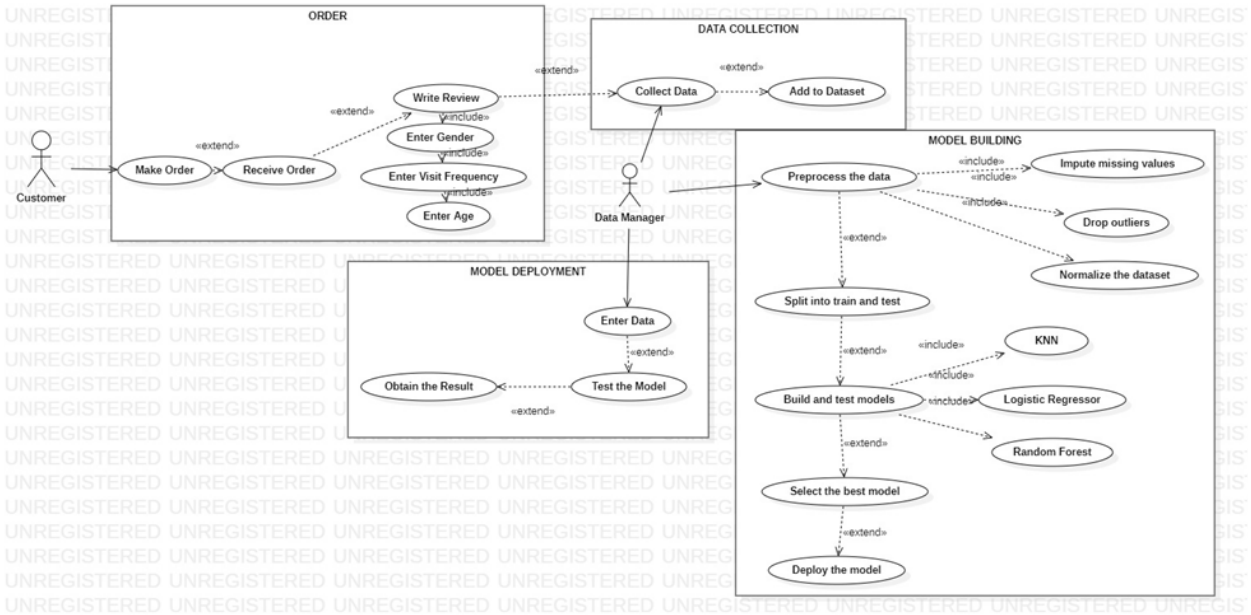
## Non-functional Requirements

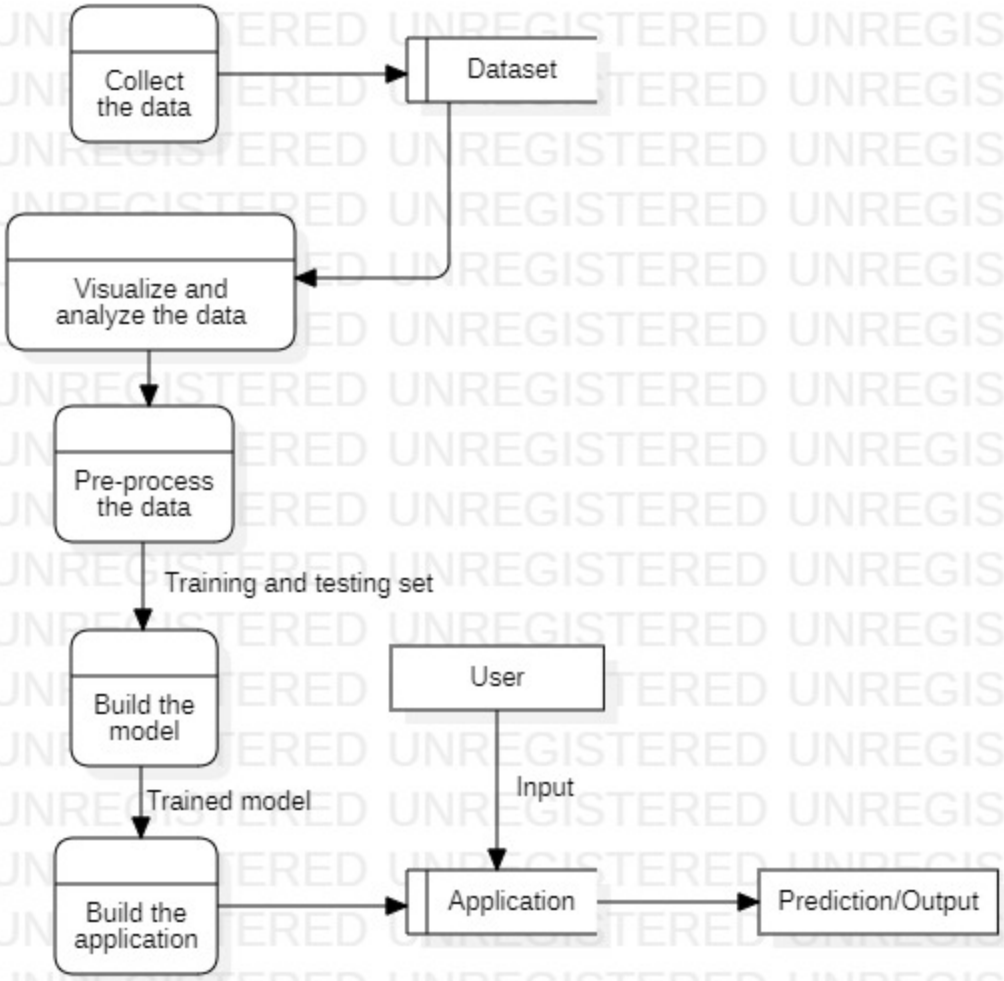
- High processing power to accomodate the heavy computation of the clustering and logistic regression algorithms.

- Very high model accuracy score (>90%) to prevent customers from being assigned to the wrong segment (cluster)

# Project Design

## Data Flow Diagrams and User Stories



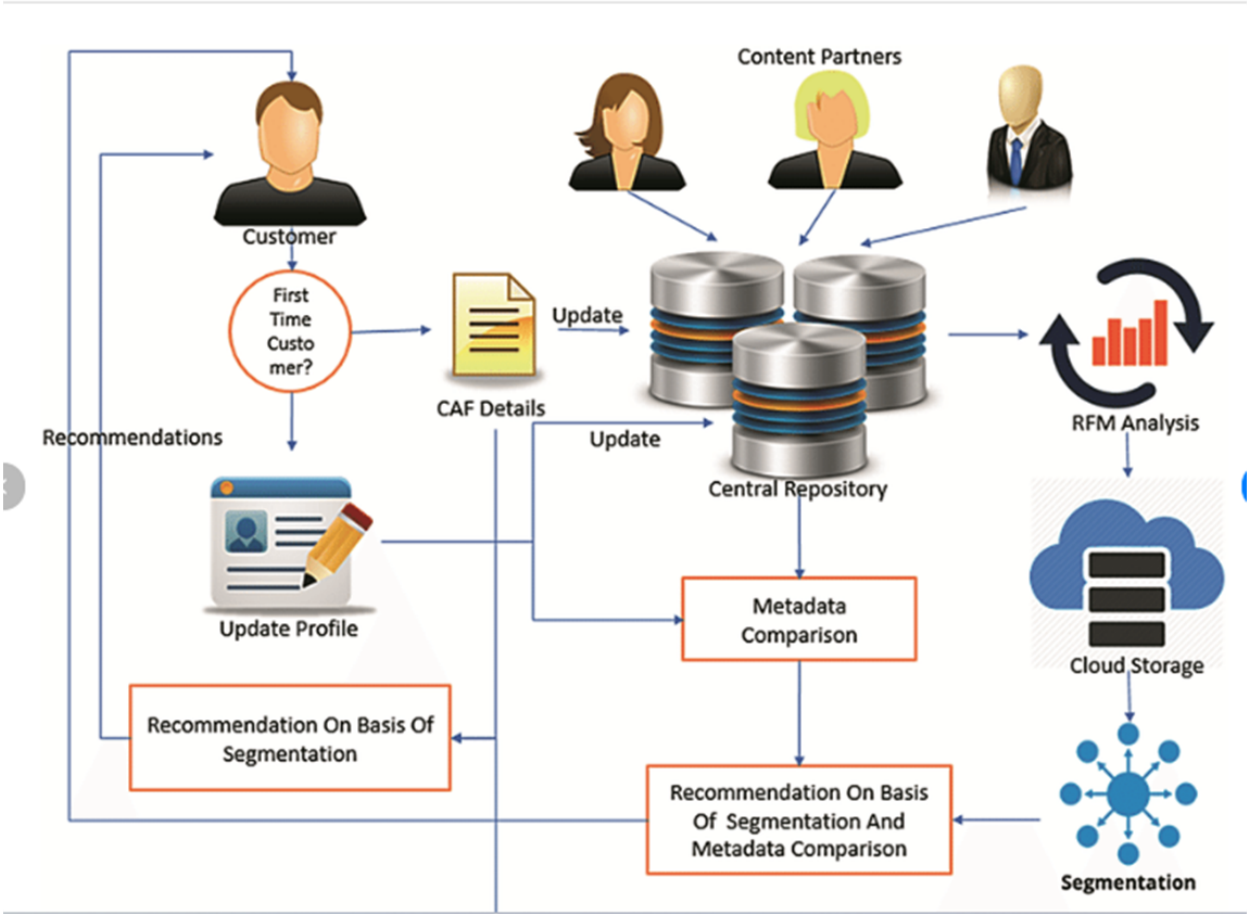

Collect the data
Dataset
Visualize and analyze the data
Pre-process the data
Training and testing set
Build the model
User
Trained model
Input
Build the application
Application
Prediction/Output

| User Type | Functional Requirement (Epic) | User Story Number | User Story/Task | Acceptance Criteria | Priority | Release |
|---|---|---|---|---|---|---|
| Customer (Mobile) | User Registration | USN-1 | Register via mobile app | - User can input email, password, and confirm password. - User data is saved securely. | High | Sprint 1 |
| Customer (Mobile) | User Registration | USN-2 | Register via mobile app using social media | - User can register using Facebook or Google. - User data is linked to their social media accounts. | Medium | Sprint 2 |
| Customer (Mobile) | User Segmentation | USN-3 | View personalized recommendations | - ML model segments the user based on behaviour - User sees product recommendations based on their segment. | High | Sprint 3 |
| Customer (Web) | User Registration | USN-4 | Register via web app | - User can input email, password, and confirm password. - User data is saved securely. | High | Sprint 1 |
| Customer (Web) | User Segmentation | USN-5 | View detailed segment information | - User can see demographic info, preferences, etc., related to their segment. | Medium | Sprint 3 |
| Marketing Team | Segment | USN-6 | Analyse customer | - Marketing team can access segment data. - Data is presented in a user- | High | Sprint 4 |

| User Type | Functional Requirement (Epic) | User Story Number | User Story/Task | Acceptance Criteria | Priority | Release |
|---|---|---|---|---|---|---|
| | Analysis | | segments | friendly format. | | |
| Administrator | User Management | USN-7 | Admin can manage user accounts | - Admin can add, delete, or suspend user accounts. - Changes are reflected in the system. | High | Sprint 5 |
| Administrator | System Maintenance | USN-8 | Perform system maintenance | - Admin can schedule and perform system maintenance tasks. - Maintenance tasks are logged. | Medium | Sprint 5 |

# Project Planning and Scheduling

## Technical Architecture




A technical architecture for market segmentation refers to the design and structure of the technology stack used to implement, manage, and execute market segmentation strategies. The architecture should support data collection, analysis, and the delivery of tailored marketing efforts to different market segments.

## Sprint Planning and Estimation

Sprint 1 - User Registration (mobile and web)
Sprint 2 - User registration through social media links
Sprint 3 - Demographic view and personalized recommendations for users
Sprint 4 - Data access for marketing team administration
Sprint 5 - Admin management and system maintenance

## Sprint Delivery Schedule

Sprint 1 - 7 days
Sprint 2 - 3 days

Sprint 3 - 10 days
Sprint 4 - 7 days
Sprint 5 - 10 days

# Coding and Solutioning

## Data Preprocessing and Visualization

**We first conduct exploratory data analysis on our dataset. The data is checked for null values and a series of charts are made to visualize the data for our convenience. These charts help us identify trends and characteristics of the dataset.**

**After visualization, we convert the dataset features into numerical form so they can be input to the ML models. A function called encoder() is made for this purpose.**

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

df = pd.read_csv('dataset.csv')
df.info()
df.isna().sum()
df.describe().T
df.describe(include=['O']).T

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)

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

```
df['Agebin'] = pd.cut(df['Age'], bins = [17,25, 35, 49, 60, 75], labels = ['17-25','26-35', '36-49', '50-60',
'61-75'])
df['Agebin'].value_counts()/len(df)*100

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

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 = [ax0,ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9,ax10,ax11,ax12,ax13]
for i in range(0,13):
  sns.countplot(data=df,x=category[i],ax=ax[i])
  ax[i].set_title('Distribution of '+category[i])
plt.savefig('count.png')

fig,[ax0,ax1] = plt.subplots(nrows=1,ncols=2,figsize=(10,5))
sns.countplot(x=df['Agebin'],ax=ax0)
sns.histplot(x=df['Age'],ax=ax1)
plt.title('Distribution of Age')
plt.savefig('count1.png')
plt.show()

sns.set_palette('pastel')
fig,([ax0,ax1],[ax2,ax3],[ax4,ax5]) = plt.subplots(nrows=3,ncols=2,figsize=(20,15))
sns.countplot(x=df['yummy'],hue=df['tasty'],ax=ax0)
sns.countplot(x=df['tasty'],hue=df['disgusting'],ax=ax1)
sns.countplot(hue=df['disgusting'],x=df['Like'],ax=ax2)
sns.countplot(x=df['fattening'],hue=df['healthy'],ax=ax3)
sns.countplot(x=df['cheap'],hue=df['expensive'],ax=ax4)
sns.countplot(x=df['spicy'],hue=df['greasy'],ax=ax5)
plt.savefig('count2.png')
plt.show()

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'])
```

```
  plt.show()
plt.savefig('count3.png')

import warnings
warnings.filterwarnings("ignore")

sns.set_palette('husl')
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)
plt.savefig('count4.png')
plt.show()

sns.set_palette('coolwarm')
for i in df.drop(['Agebin','yummy','cheap','healthy','greasy','Age','VisitFrequency'],axis=1):
  grid = sns.FacetGrid(df,height=4,col='Agebin')
  grid = grid.map(sns.histplot,i,bins=30)
plt.savefig('count6.png')
plt.show()

def encoder():
    df['yummy'] = df['yummy'].replace(['Yes','No'],[1,0])
    df['convenient'] = df['convenient'].replace(['Yes','No'],[1,0])
    df['spicy'] = df['spicy'].replace(['Yes','No'],[1,0])
    df['fattening'] = df['fattening'].replace(['Yes','No'],[1,0])
    df['greasy'] = df['greasy'].replace(['Yes','No'],[1,0])
    df['fast'] = df['fast'].replace(['Yes','No'],[1,0])
    df['cheap'] = df['cheap'].replace(['Yes','No'],[1,0])
    df['tasty'] = df['tasty'].replace(['Yes','No'],[1,0])
    df['expensive'] = df['expensive'].replace(['Yes','No'],[1,0])
    df['healthy'] = df['healthy'].replace(['Yes','No'],[1,0])
    df['disgusting'] = df['disgusting'].replace(['Yes','No'],[1,0])
    df['Gender'] = df['Gender'].replace(['Male','Female'],[1,0])
    df['VisitFrequency'] = df['VisitFrequency'].replace(['Never','Once a year','Every three
months','Once a month','Once a week','More than once a week'],[0,1,2,3,4,5])
    df['Like'] = df['Like'].replace(['I hate it!-5','-4','-3','-2','-1','0','1','2','3','4','I love it!+5'],
                            [0,1,2,3,4,5,6,7,8,9,10])
encoder()

plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True)
```

```
plt.savefig('count7.png')
```

## K-Means Clustering

**Principal Component Analysis (PCA) is first applied on the dataset. Then we use a K-Means clustering model to segment the customers into clusters based on their shared features.**

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

plt.figure(figsize=(10,6))
sns.scatterplot(data=pc,x='pc1',y='pc2',color='purple')
plt.title('Data Distribution')
plt.savefig('count8.png')

from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
kmeans = KMeans()
visualizer = KElbowVisualizer(kmeans, k=(1,12)).fit(pc)
visualizer.show()
plt.savefig('count9.png')

kmeans = KMeans(n_clusters=3)
kmeans.fit(pc)

np.random.seed(42)
preds = kmeans.predict(pc)

import pickle
pickle.dump(kmeans,open('clustering.pkl','wb'))

plt.figure(figsize=(10,6))
sns.scatterplot(x=pc['pc1'],y=pc['pc2'],hue=preds)
plt.title('Data Distribution')
plt.savefig('count10.png')
plt.show()

df['cluster'] = preds
```

```
sns.countplot(x = df['cluster'])
plt.savefig('count11.png')
df['cluster'].value_counts()/len(df)*100

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

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

## Logistic Regression Classifier

**We first assign each review to one of five clusters. Then we use  logistic regressor to predict the cluster for each review. We measure the accuracy by comparing the predicted cluster with the assigned cluster.**

```
from sklearn.model_selection import train_test_split
x = df.drop(['cluster','Agebin'],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)
x_train.shape,y_train.shape

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(x_train,y_train)
preds = clf.predict(x_test)

from sklearn.metrics import classification_report,confusion_matrix,ConfusionMatrixDisplay
print(classification_report(y_test,preds))

from sklearn.metrics import accuracy_score
print(accuracy_score(y_test,preds))
```

```
sns.set_style("whitegrid", {'axes.grid' : False})
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(cmap='Greens',colorbar=True,)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.savefig('count12.png')
plt.show()

result = clf.predict(scaler.transform([[1,1,0,0,0,1,1,1,0,1,0,3,30,4,0]]))
print("Cluster = ",result)

import pickle
pickle.dump(clf,open('predictor.pkl','wb'))
pickle.dump(scaler,open('scaler1.pkl','wb'))
```

## HTML Page

**This is the API frontend of our application. The user is asked to enter values in text boxes and click a button at the bottom to predict the cluster. There is a JavaScript frontend to allow interactivity with the application.**

```
<!DOCTYPE html>
<html lang="en">

<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Market Segmentation analysis</title>

 <style type="text/css">
  * {
   text-decoration: none;
  }

  .navbar {
   background: cornflowerblue;
   font-family: calibri;
   padding-right: 15px;
```

```
    padding-left: 15px;
   }

   .navdiv {
    display: flex;
    align-items: center;
    justify-content: space-between;
   }

   .logo a {
    font-size: 35px;
    font-weight: 600;
    color: white;
   }

   li {
    list-style: none;
    display: inline-block;
   }

   li a {
    color: white;
    font-size: 18px;
    font-weight: bold;
    margin-right: 25px;
   }
  </style>
</head>

<body>
  <nav class="navbar">
   <div class="navdiv">
    <div class="logo">
     <a href="#">MARKET SEGMENTATION ANALYSIS USING ML.</a>
    </div>
    <ul>
     <li><a href="#">Home</a></li>
     <li><a href="#">About </a></li>
     <li><a href="#">Contact </a></li>
    </ul>
   </div>
```

```
</nav>

<form action="/predict" method="POST">
  <div style="text-align: center;">

    <label for="yummy">yummy:</label> <br>
    <input type="text" id="yummy" name="yummy"><br><br>
    <label for="convenient">Convenient:</label><br>
    <input type="text" id="convenient" name="convenient"><br><br>

    <label for="spicy">Spicy:</label><br>
    <input type="text" id="spicy" name="spicy"><br><br>

    <label for="fattening">Fattening:</label><br>
    <input type="text" id="fattening" name="fattening"><br><br>

    <label for="greasy">Greasy:</label><br>
    <input type="text" id="greasy" name="greasy"><br><br>

    <label for="fast">Fast:</label><br>
    <input type="text" id="fast" name="fast"><br><br>

    <label for="cheap">Cheap:</label><br>
    <input type="text" id="cheap" name="cheap"><br><br>

    <label for="tasty">Tasty:</label><br>
    <input type="text" id="tasty" name="tasty"><br><br>

    <label for="expensive">Expensive:</label><br>
    <input type="text" id="expensive" name="expensive"><br><br>


    <label for="healthy">healthy:</label><br>
    <input type="text" id="healthy" name="healthy"><br><br>

    <label for="disgusting">disgusting:</label><br>
    <input type="text" id="disgusting" name="disgusting"><br><br>

    <label for="Like">Like:</label><br>
    <input type="text" id="Like" name="Like"><br><br>
```

```
    <label for="Age">Age:</label><br>
    <input type="text" id="Age" name="Age"><br><br>

    <label for="VisitFrequency">VisitFrequency:</label><br>
    <input type="text" id="VisitFrequency" name="VisitFrequency"><br><br>

    <label for="Gender">Gender:</label><br>
    <input type="text" id="Gender" name="Gender"><br><br>

    <button id="predict" class="btn btn-primary btn-block btn-large">Predict!</button><br><br>

   </div>
 </form>
 <br>
 <br>
 {{ prediction_text }}
</body>
<script>
var predictButton = document.getElementById('predict');
   predictButton.addEventListener('click', function() {
      var ym = document.getElementById('yummy').value;
      var cn = document.getElementById('convenient').value
      var sp = document.getElementById('spicy').value;
      var ft = document.getElementById('fattening').value;
      var gr = document.getElementById('greasy').value;
      var fs = document.getElementById('fast').value;
      var ch = document.getElementById('cheap').value;
      var ts = document.getElementById('tasty').value;
      var ex = document.getElementById('expensive').value;
      var he = document.getElementById('healthy').value;
      var ds = document.getElementById('disgusting').value;
      var lk = document.getElementById('Like').value;
      var a = parseInt(document.getElementById('Age').value, 10);
      var vf = document.getElementById('VisitFrequency').value;
      var g = document.getElementById('Gender').value;

      // console.log('hello');

      var xhr = new XMLHttpRequest();
      xhr.open('POST', '/predict_api', true);
      xhr.setRequestHeader('Content-Type', 'application/json');
```

```
        xhr.onreadystatechange = function() {
            if (xhr.readyState === 4 && xhr.status === 200) {
                alert('Data sent to Flask!');
                // You can handle the response from Flask here
            }
        };

        // Send the data as JSON

        xhr.send(JSON.stringify([[ym,cn,sp,ft,gr,fs,ch,ts,ex,he,ds,a,lk,vf,g]]));
    })
</script>
</html>
```

## CSS Style

```
* {
    text-decoration: none;
  }

  .navbar {
    background: cornflowerblue;
    font-family: calibri;
    padding-right: 15px;
    padding-left: 15px;
  }

  .navdiv {
    display: flex;
    align-items: center;
    justify-content: space-between;
  }

  .logo a {
    font-size: 35px;
    font-weight: 600;
    color: white;
  }

  li {
```

```
  list-style: none;
  display: inline-block;
 }

 li a {
  color: white;
  font-size: 18px;
  font-weight: bold;
  margin-right: 25px;
 }
```

## JavaScript Frontend

```
var predictButton = document.getElementById('predict');
  predictButton.addEventListener('click', function() {
     console.log('hello');
  })
```

## Python Request

```
import requests

url = 'http://localhost:5000/predict-api'
r = requests.post(url,json={'yummy':'Yes','convenient':'Yes','spicy':'No'
                    'fattening':'No','greasy':'No','fast':'Yes',
                    'cheap':'Yes','tasty':'Yes','expensive'='No,
                    'healthy':'Yes','disgusting':'No','Age':30,
                    'Like':'3','VisitFrequency':'Once a week',
                    'Gender':'Female'})

print(r.json())
```

## Application

**This is the Flask app which deploys our model as a web application. Our models and scaler are imported here as .pkl files. Every input has an exception which is triggered when an invalid value is entered.**

```
import numpy as np
from flask import Flask,request,jsonify,render_template
import pickle
```

```
app = Flask(__name__)
clust = pickle.load(open('clustering.pkl','rb'))
regr = pickle.load(open('predictor.pkl','rb'))
scaler = pickle.load(open('scaler1.pkl','rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    form_data = request.form

    features = [
        "yummy", "convenient", "spicy", "fattening", "greasy", "fast",
        "cheap", "tasty", "expensive", "healthy", "disgusting",
        "Like", "Age" , "VisitFrequency", "Gender"
    ]

    data = []
    for feature in features:
        value = form_data.get(feature)
        try:

            if value in ['Yes', 'No']:
                value = 1 if value == 'Yes' else 0

            elif feature== 'Age':
                if int(value)>=17:
                    value = int(value)
                else:
                    raise ValueError("You must be at least 17 years old")

            elif feature == 'Like':
                if value == 'I hate it!-5':
                    value = 0
                elif value == '-4':
                    value = 1
                elif value == '-3':
                    value = 2
```

```
            elif value == '-2':
                value = 3
            elif value == '-1':
                value = 4
            elif value == 'I love it!+5':
                value = 10
            elif 0 <= int(value) <= 4:
                value = int(value) + 5
            else:
                raise ValueError("Invalid input for Like")

        elif feature == 'VisitFrequency':
            if value == 'Never':
                value = 0
            elif value == 'Once a year':
                value = 1
            elif value == 'Every three months':
                value = 2
            elif value == 'Once a month':
                value = 3
            elif value == 'Once a week':
                value = 4
            elif value == 'More than once a week':
                value = 5
            else:
                raise ValueError("Invalid input for VisitFrequency")

        elif feature == 'Gender':
            if value == 'Female':
                value = 0
            elif value == 'Male':
                value = 1
            else:
                raise ValueError("Invalid input for Gender. Please enter Male or Female")

        else:
            raise KeyError("Invalid feature encountered")

    except (ValueError, KeyError) as e:
        print(f"Exception occurred: {e}")
```

```
        data.append(value)

    data = np.array(data)
    data = data.reshape(1,-1)

    final = regr.predict(scaler.transform(data))

    output = final[0]

    return render_template('index.html',prediction_text = 'The predicted cluster is
{}'.format(output))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    try:
        data = request.get_json()
        clust_pred = clust.predict(data)
        final = regr.predict(clust_pred)

        output = final[0]
        return jsonify({'prediction':output})
    except Exception as e:
        return jsonify({'error':str(e)})

if __name__=="__main__":
    app.run(debug=True)
```

## Database Schema

```
<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
```

# Performance Testing

```
## performmance of the model
from sklearn.metrics import classification_report,confusion_matrix,ConfusionMatrixDisplay

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

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99        69
           1       0.98      0.98      0.98       124
           2       0.97      0.96      0.96        98

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

---

Observation

- Model is performing great
- Need not to tune parameters

The accuracy score is very high, proving this model's reliability.

# Results

### Importing Libraries

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

### Getting Data

In [2]:
```
df = pd.read_csv('dataset_int.csv')
df.head()
```

Out[2]:

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

```
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   int64 
 12  Age             1453 non-null   int64 
 13  VisitFrequency  1453 non-null   object
 14  Gender          1453 non-null   object
dtypes: int64(2), object(13)
memory usage: 170.4+ KB
```

```
## checking for missing values
df.isna().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
```

Observations

- There are no missing values.

Since there are no missing values, there is no need for imputation.

```
df.describe().T
```

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Like** | 1453.0 | 0.763248 | 3.120244 | -5.0 | -1.0 | 1.0 | 3.0 | 5.0 |
| **Age** | 1453.0 | 44.604955 | 14.221178 | 18.0 | 33.0 | 45.0 | 57.0 | 71.0 |

- Mean of the age of cutomers is 45.
- Min Age is 18 where as maximum is 71.

```
df.describe(include=['O']).T
```

| | count | unique | top | freq |
|---|---|---|---|---|
| **yummy** | 1453 | 2 | Yes | 803 |
| **convenient** | 1453 | 2 | Yes | 1319 |
| **spicy** | 1453 | 2 | No | 1317 |
| **fattening** | 1453 | 2 | Yes | 1260 |
| **greasy** | 1453 | 2 | Yes | 765 |
| **fast** | 1453 | 2 | Yes | 1308 |
| **cheap** | 1453 | 2 | Yes | 870 |
| **tasty** | 1453 | 2 | Yes | 936 |
| **expensive** | 1453 | 2 | No | 933 |
| **healthy** | 1453 | 2 | No | 1164 |
| **disgusting** | 1453 | 2 | No | 1100 |
| **VisitFrequency** | 1453 | 6 | Once a month | 439 |
| **Gender** | 1453 | 2 | Female | 788 |

## Data Processing

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

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

```
Distribution of cheap
Yes    870
No     583
Name: cheap, dtype: int64
```

```
Distribution of spicy
No     1317
Yes     136
Name: spicy, dtype: int64
```

```
Distribution of tasty
Yes    936
No     517
Name: tasty, dtype: int64
```

```
Distribution of fattening
Yes    1260
No      193
Name: fattening, dtype: int64
```

```
Distribution of expensive
No     933
Yes    520
Name: expensive, dtype: int64
```

```
Distribution of greasy
Yes    765
No     688
Name: greasy, dtype: int64
```

```
Distribution of healthy
No     1164
Yes     289
Name: healthy, dtype: int64
```

```
Distribution of fast
Yes    1308
No      145
Name: fast, dtype: int64
```

```
Distribution of disgusting
No     1100
Yes     353
Name: disgusting, dtype: int64
```

```
Distribution of 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: VisitFrequency, dtype: int64
```

```
Distribution of Gender
Female    788
Male      665
Name: Gender, 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

```
In [8]: df['Age'].value_counts().sort_values()
```

```
Out[8]: 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
```

```
## creating bins for the age

df['Agebin'] = pd.cut(df['Age'], bins = [17,25, 35, 49, 60, 75], labels = ['17-25','26-35', '36-49', '50-60', '61-75'])
```

```
df['Agebin'].value_counts()/len(df)*100
```

```
36-49    27.253957
50-60    26.496903
26-35    18.857536
61-75    15.554026
17-25    11.837577
Name: Agebin, dtype: float64
```

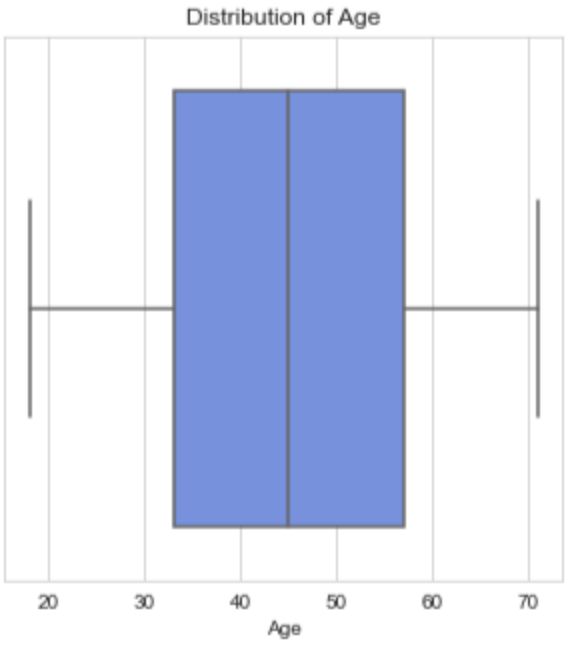
Observations

- More than 50% of the customers belongs to 36-50
- only 11% customers belongs to adult age

We segment our customer base into **five age groups.**

# Data Visualization

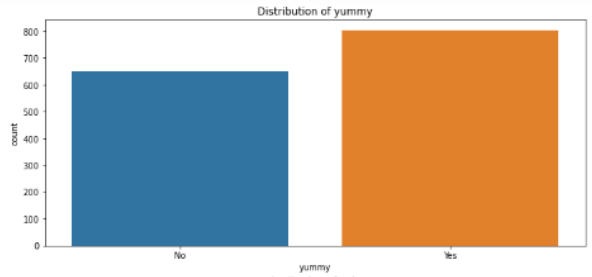
```
sns.set_style('whitegrid')
plt.figure(figsize=(5,5))
sns.set_palette('coolwarm')
sns.boxplot(x=df['Age'])
plt.title('Distribution of Age')
plt.show()
```

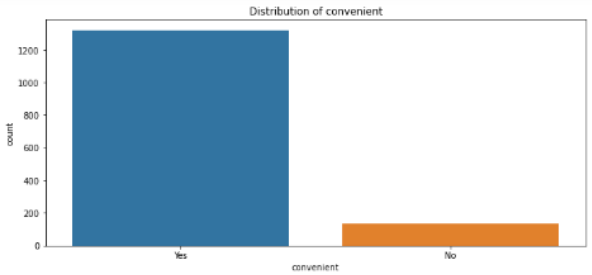


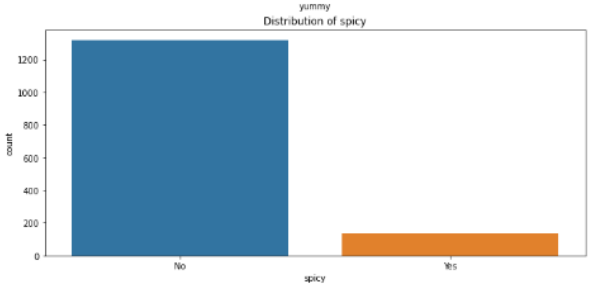

- There are no outliers in the Age

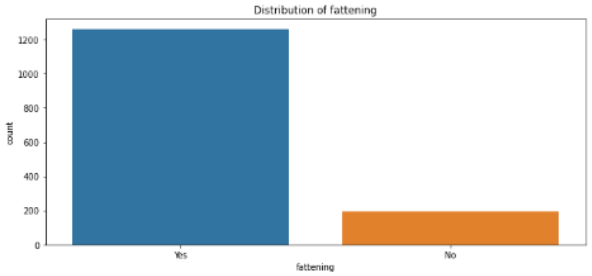
The mean age is 45
The upper quartile is 57 and the lower quartile is 33
The interquartile range is 24

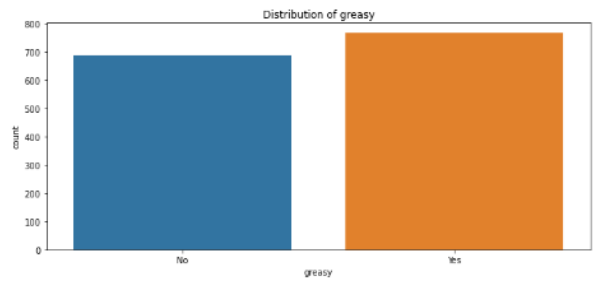
```
fig,([ax0,ax1],[ax2,ax3],[ax4,ax5],[ax6,ax7],[ax8,ax9],[ax10,ax11],[ax12,ax13]) = plt.subplots(ncols=2,nrows=7,figsize=(25,40))

ax = [ax0,ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9,ax10,ax11,ax12,ax13]
for i in range(0,13):
  sns.countplot(data=df,x=category[i],ax=ax[i])
  ax[i].set_title('Distribution of '+category[i])

plt.savefig('count.png')
```

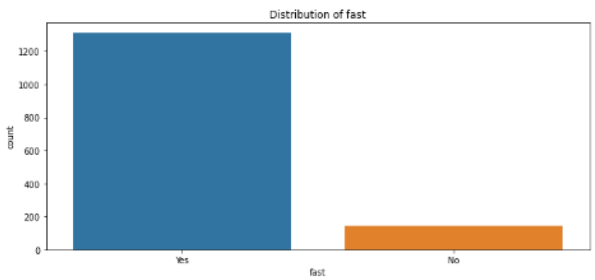

Distribution of yummy
count
yummy
No
Yes

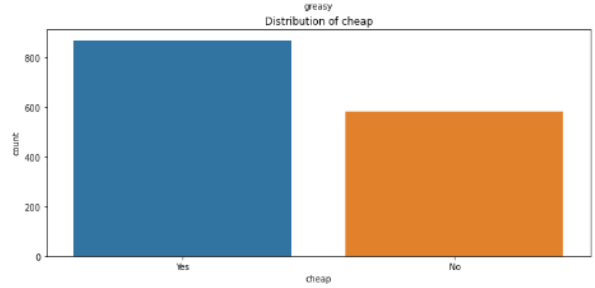

Distribution of convenient
count
convenient
Yes
No

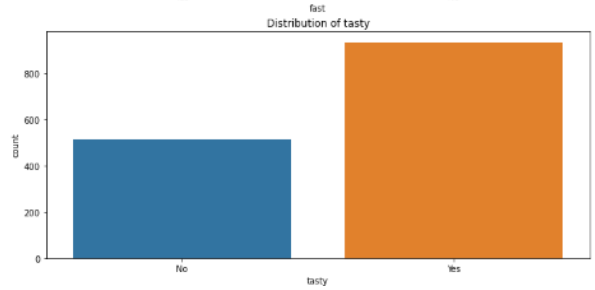

Distribution of spicy
count
spicy
No
Yes

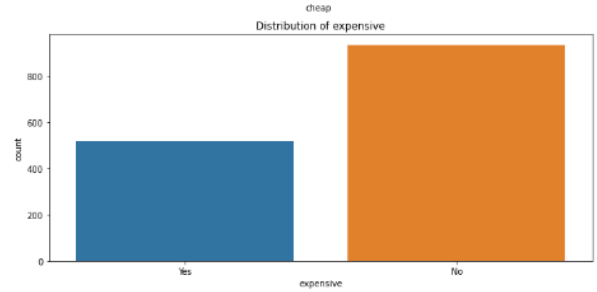

Distribution of fattening
count
fattening
Yes
No

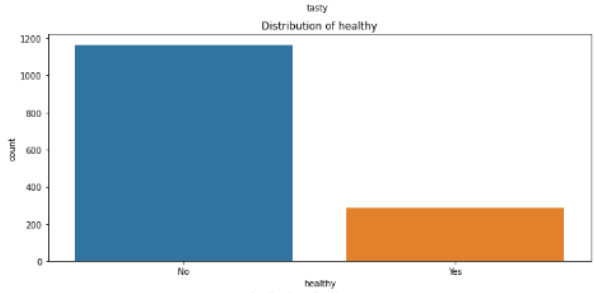

Distribution of greasy
count
greasy
No
Yes

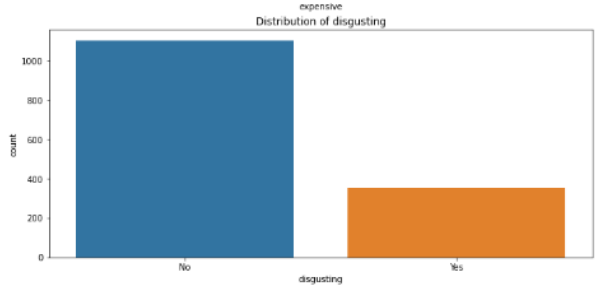

Distribution of fast
count
fast
Yes
No

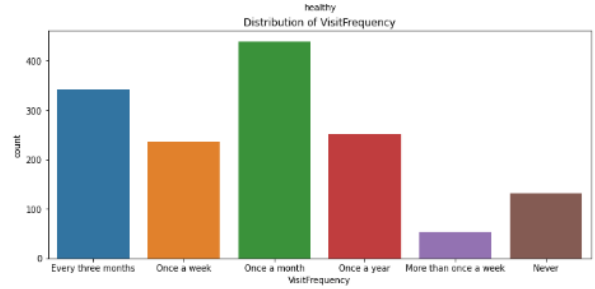

Distribution of cheap
count
cheap
Yes
No

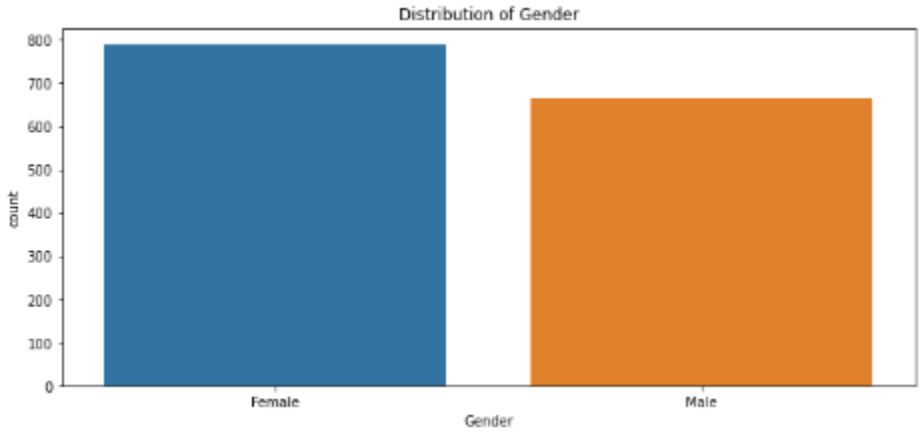

Distribution of tasty
count
tasty
No
Yes


Distribution of expensive
count
expensive
Yes
No


Distribution of healthy
count
healthy
No
Yes


Distribution of disgusting
count
disgusting
No
Yes


Distribution of VisitFrequency
count
VisitFrequency
Every three months
Once a week
Once a month
Once a year
More than once a week
Never



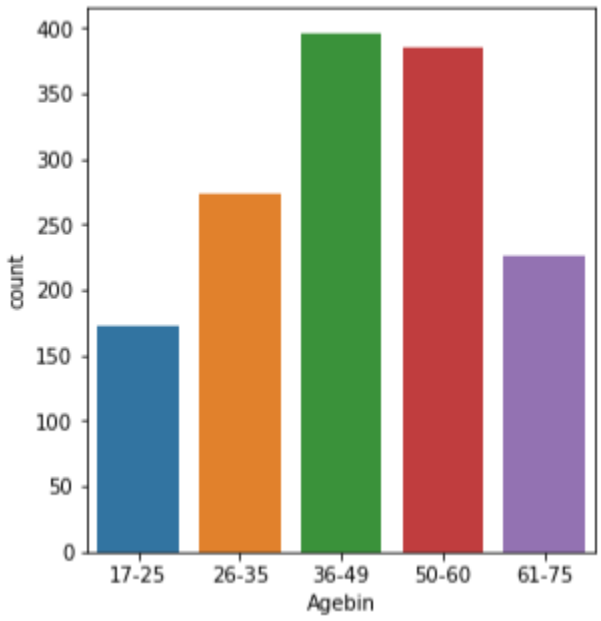

Observations

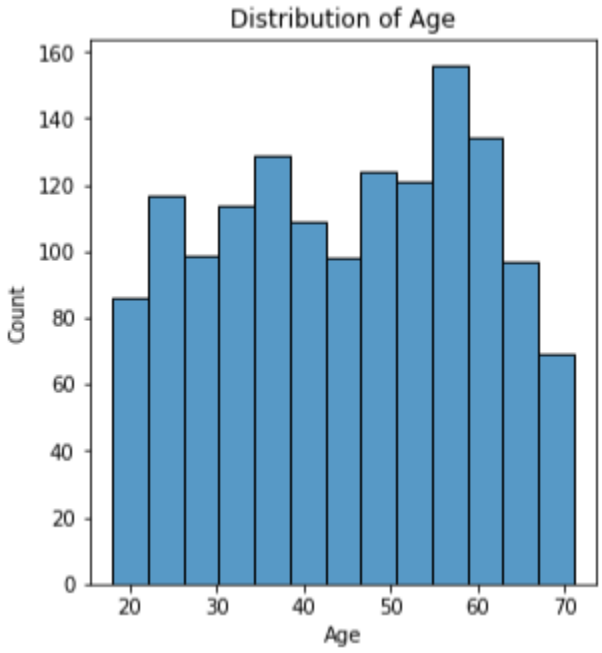
- There are many customers who have never visited once
- Majority of the customers visits once a month
- +3 and +2 is given by approx 30 percent the customers
- 60% customers Found the food yummy
- Approx 90 percent doesn't found convinent and spicy
- Most of the customers found the service fast and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

```
fig,[ax0,ax1] = plt.subplots(nrows=1,ncols=2,figsize=(10,5))
sns.countplot(x=df['Agebin'],ax=ax0)
sns.histplot(x=df['Age'],ax=ax1)
plt.title('Distribution of Age')
plt.savefig('count1.png')
plt.show()
```

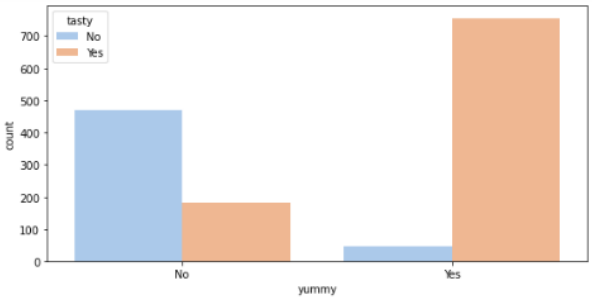
Observations

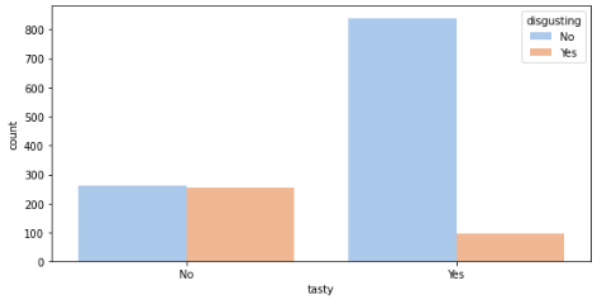
- Majority of the customers aged between 36-49
- Distribution of age is quite a normal
- Atleast 10 percent of the customers belongs to each of the age group

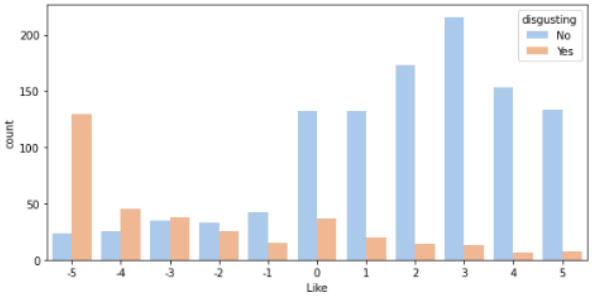


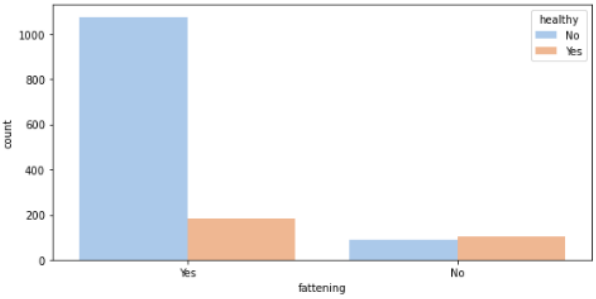



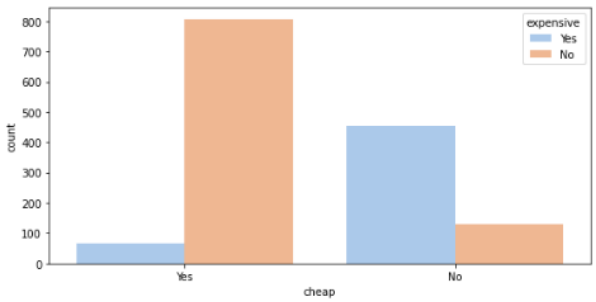

```
sns.set_palette('pastel')
fig,([ax0,ax1],[ax2,ax3],[ax4,ax5]) = plt.subplots(nrows=3,ncols=2,figsize=(20,15))
sns.countplot(x=df['yummy'],hue=df['tasty'],ax=ax0)
sns.countplot(x=df['tasty'],hue=df['disgusting'],ax=ax1)
sns.countplot(hue=df['disgusting'],x=df['Like'],ax=ax2)
sns.countplot(x=df['fattening'],hue=df['healthy'],ax=ax3)
sns.countplot(x=df['cheap'],hue=df['expensive'],ax=ax4)
sns.countplot(x=df['spicy'],hue=df['greasy'],ax=ax5)
plt.savefig('count2.png')
plt.show()
```

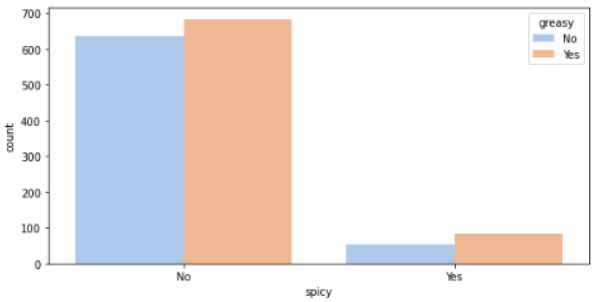


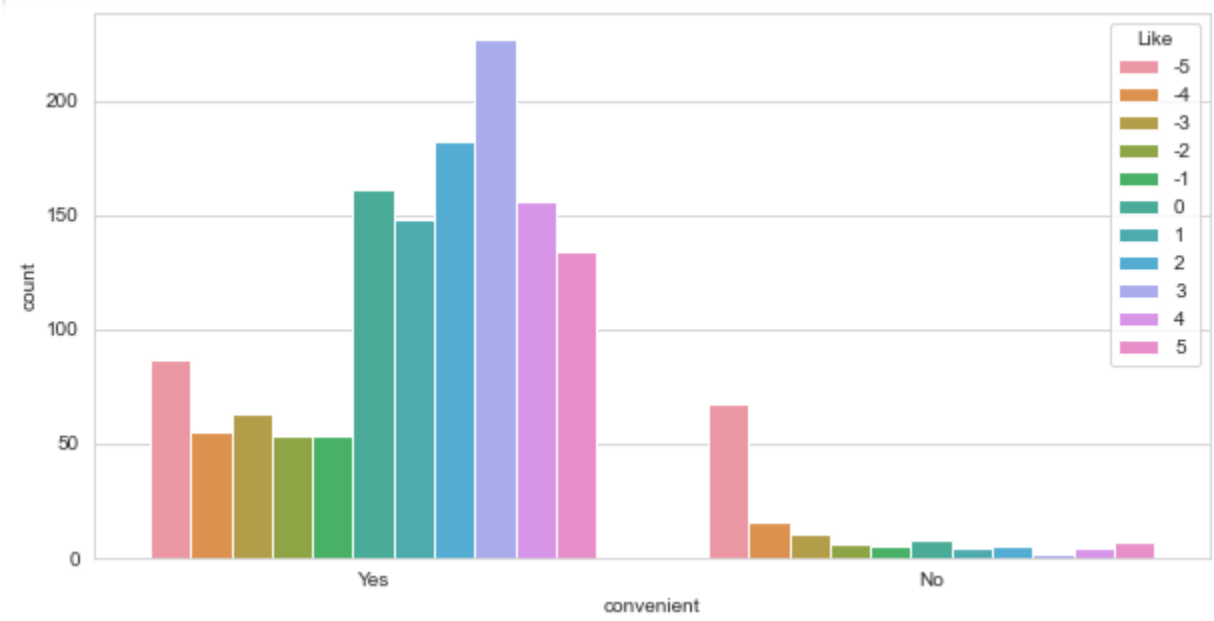


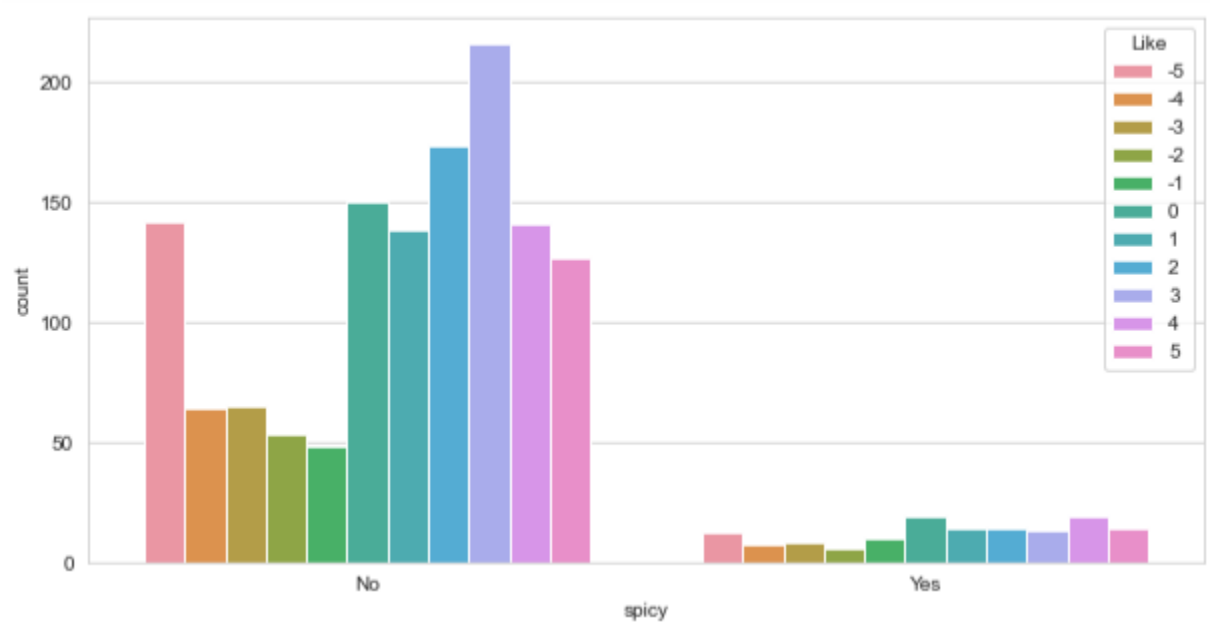



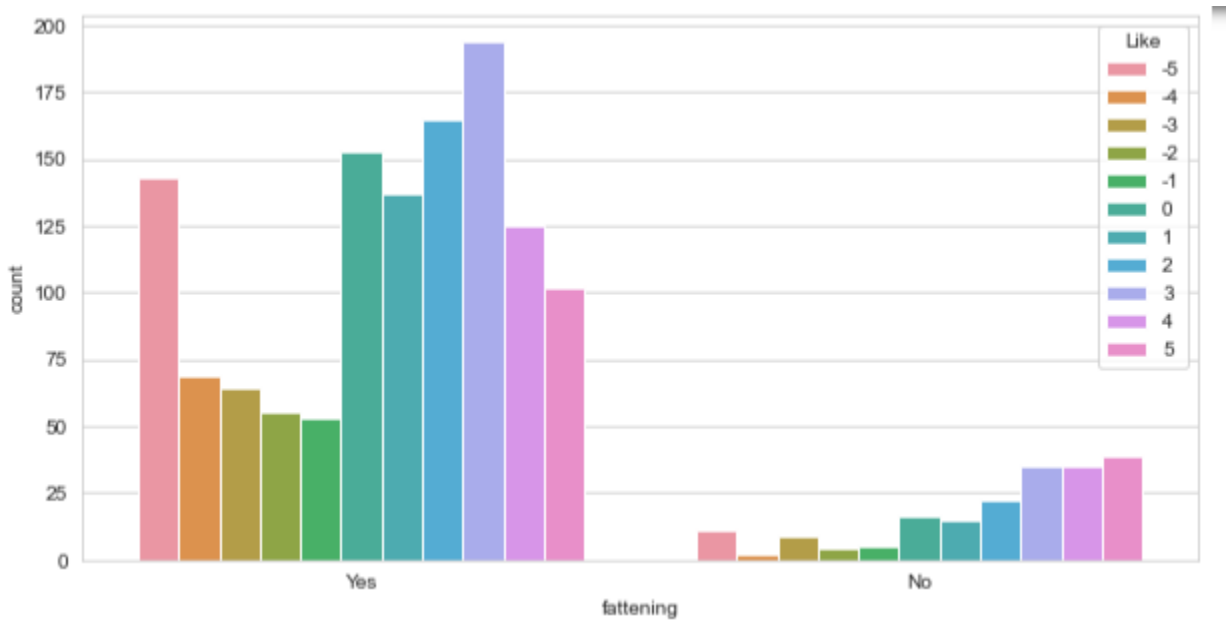

Observations

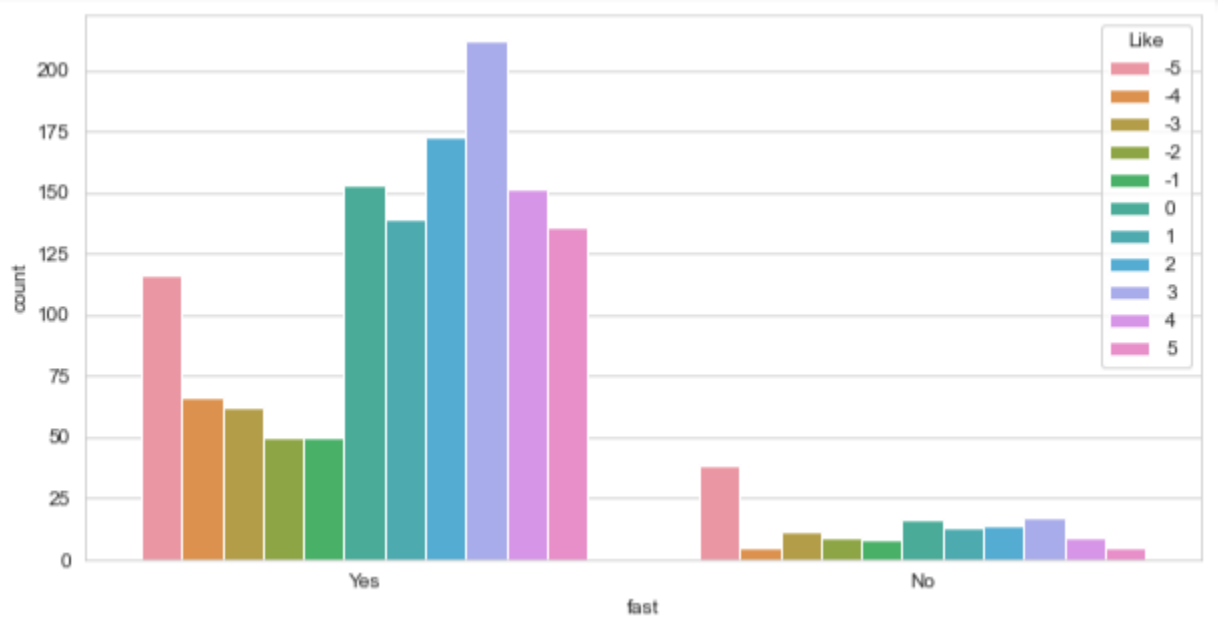
- From the plot it can be seen data have alot of discrepencies
- `yummy` and `tasty` are a kind of same can remove either of one
- Some of the customers rate the food tasty as well as disgusting and vice-versa, needs to check the data
- same error can be seen in `cheap` , `expensive` , `disgusting` , `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

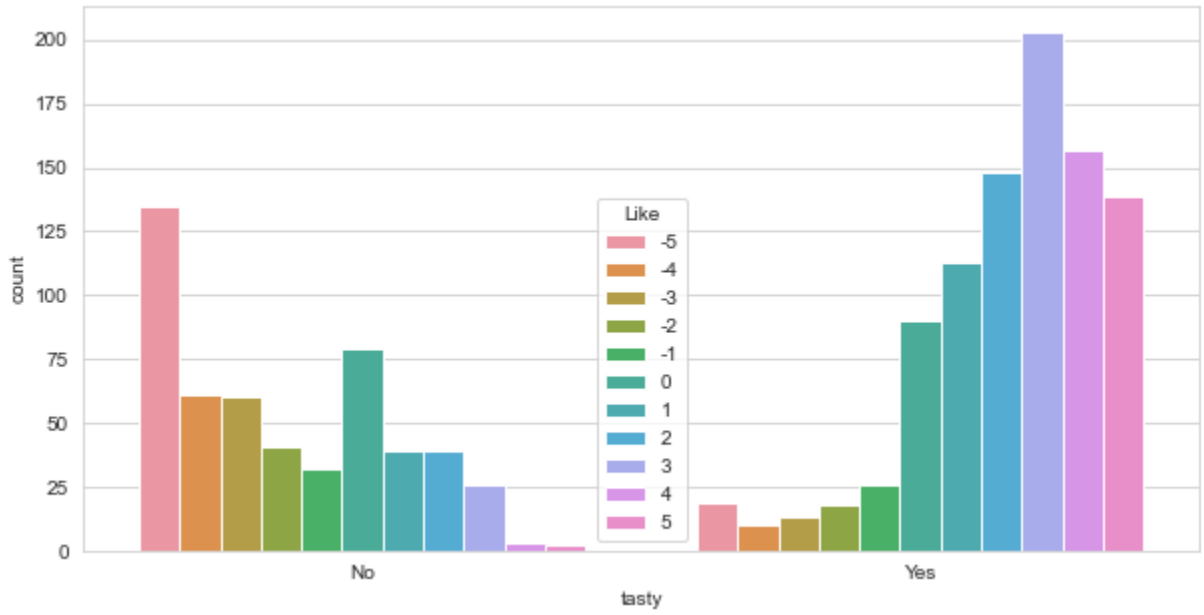
```
sns.set_style('whitegrid')
for i in df.drop(['Like','yummy','cheap','healthy','greasy','Age'],axis=1).columns:
  plt.figure(figsize=(10,5))
  sns.countplot(x=df[i],hue=df['Like'])
  plt.show()

plt.savefig('count3.png')
```

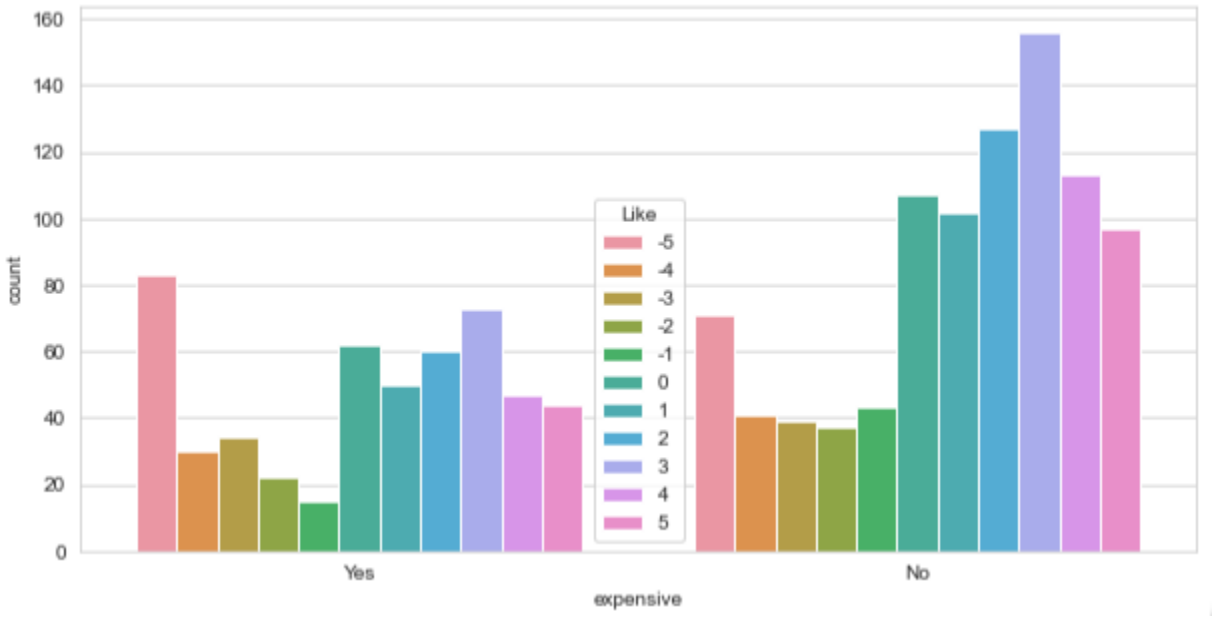


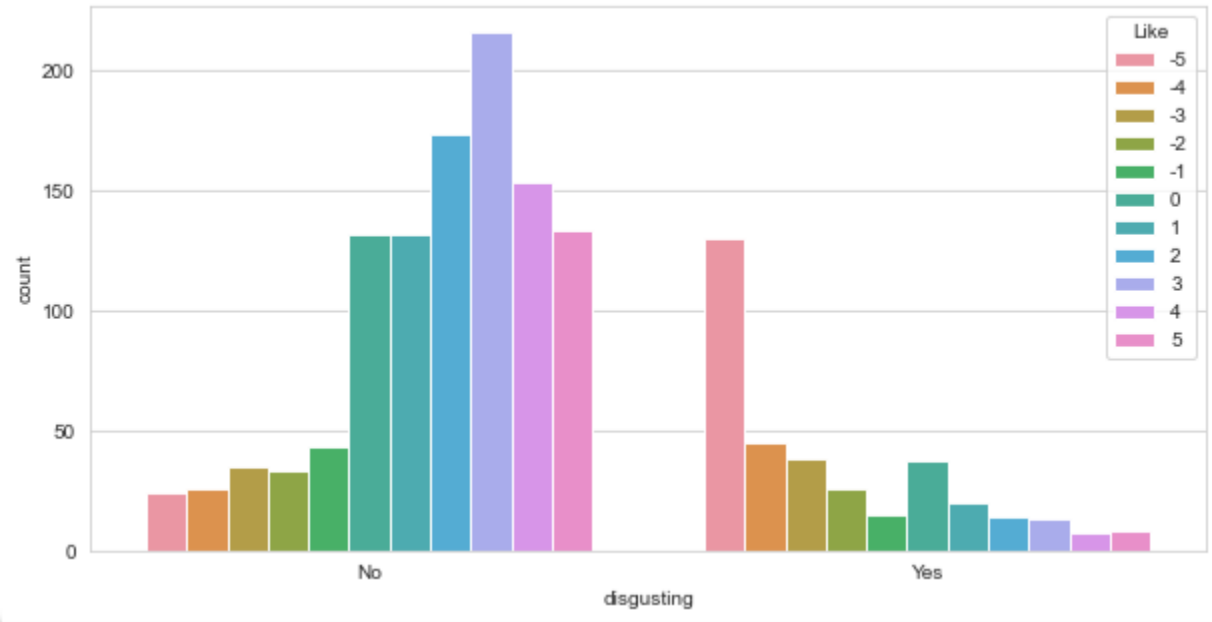

Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
count
200
150
100
50
0
No
Yes
spicy

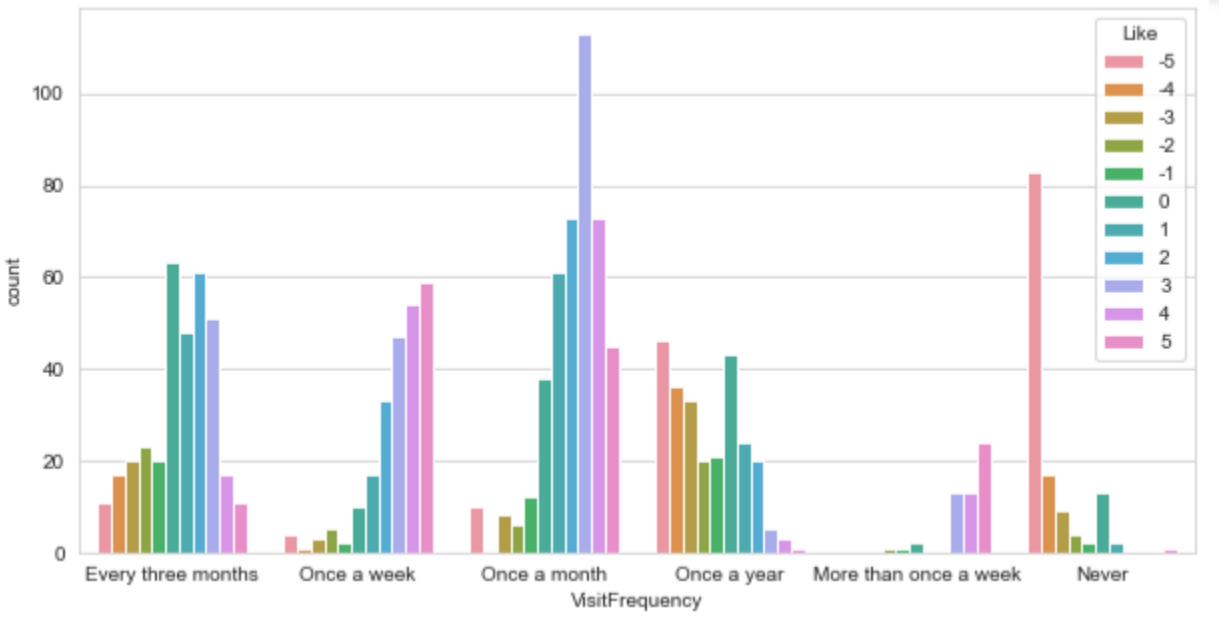

Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
count
200
175
150
125
100
75
50
25
0
Yes
No
fattening

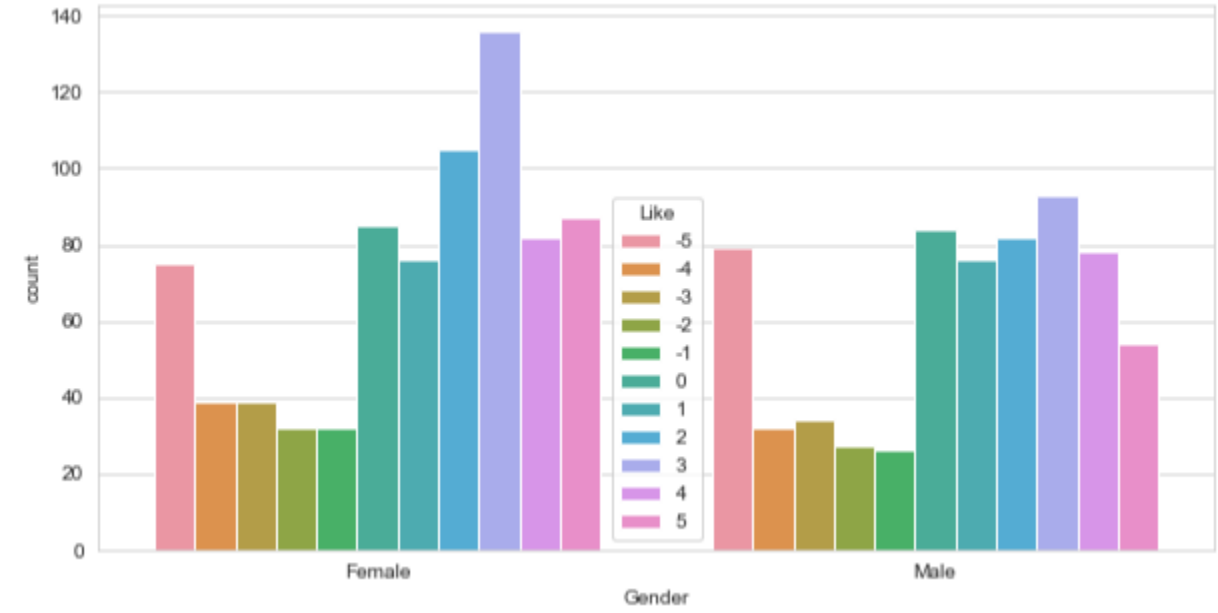

Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
count
200
175
150
125
100
75
50
25
0
Yes
No
fast

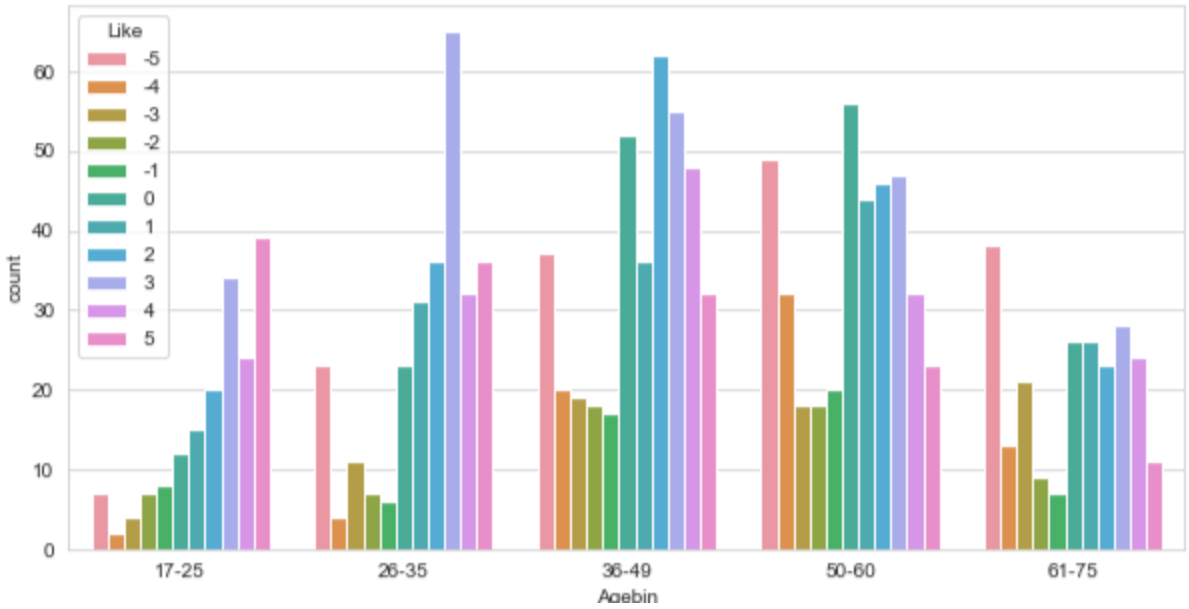

Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
count
200
175
150
125
100
75
50
25
0
No
Yes
tasty


160
140
120
100
80
60
40
20
0
count
Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
Yes
No
expensive


200
150
100
50
0
count
Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
No
Yes
disgusting


Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
count
100
80
60
40
20
0
Every three months
Once a week
Once a month
Once a year
More than once a week
Never
VisitFrequency


Like
-5
-4
-3
-2
-1
0
1
2
3
4
5
count
140
120
100
80
60
40
20
0
Female
Male
Gender

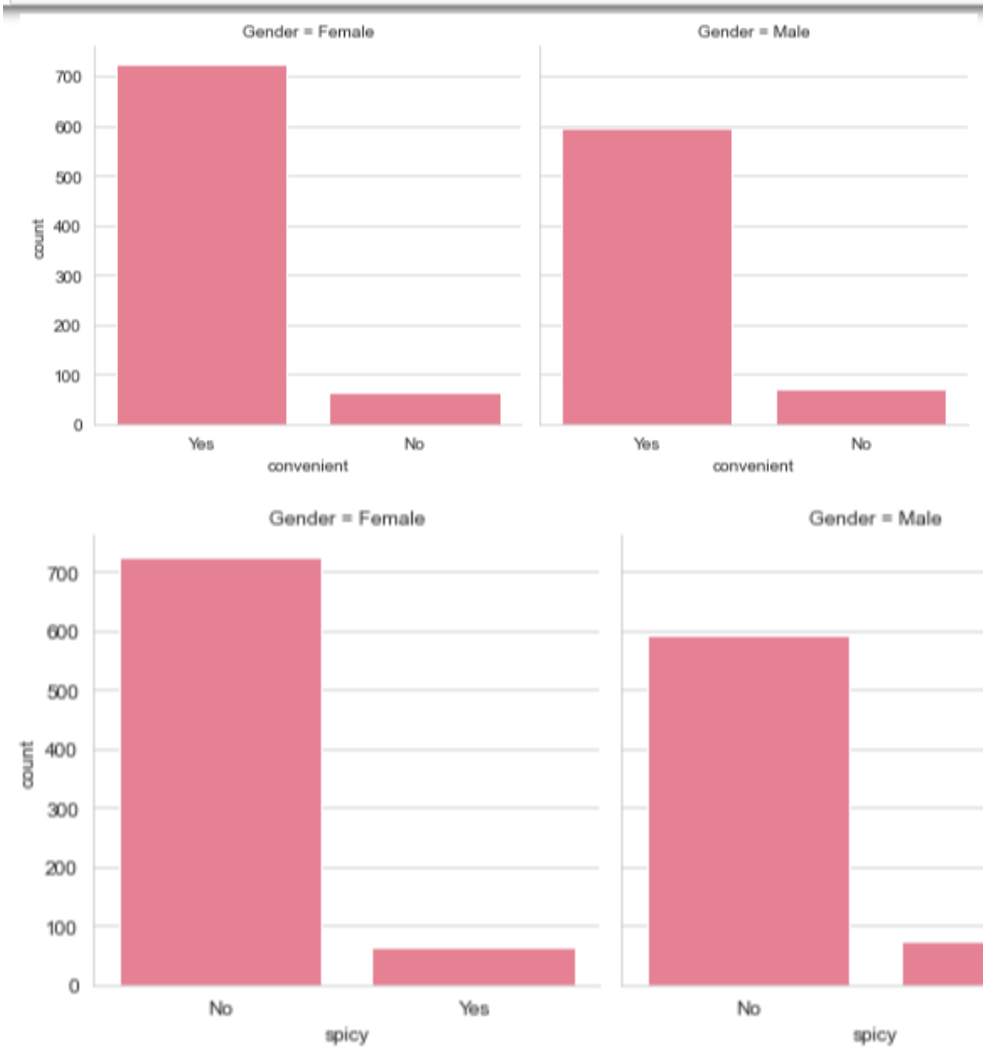


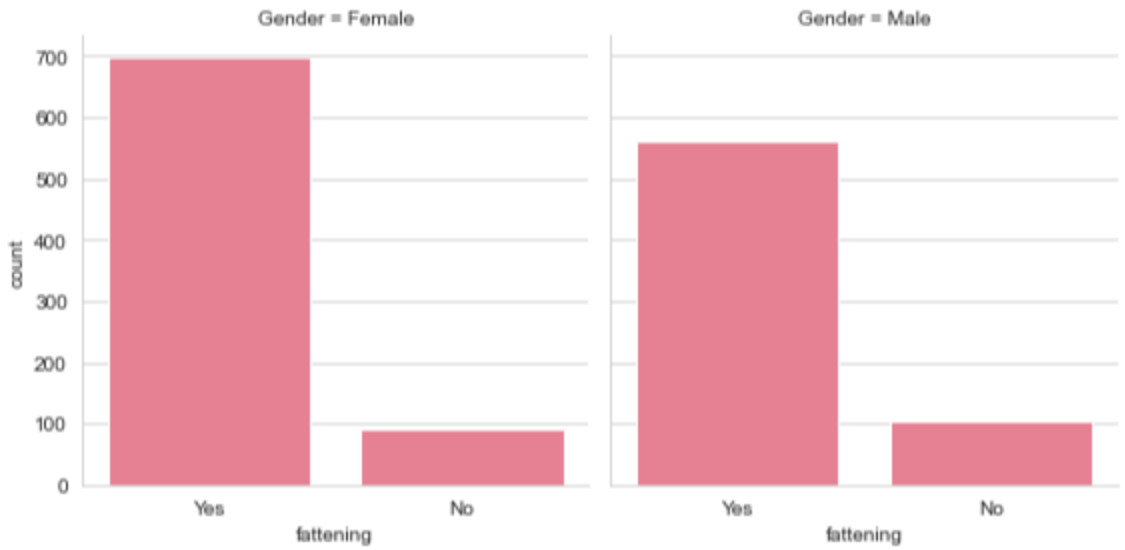

Observations

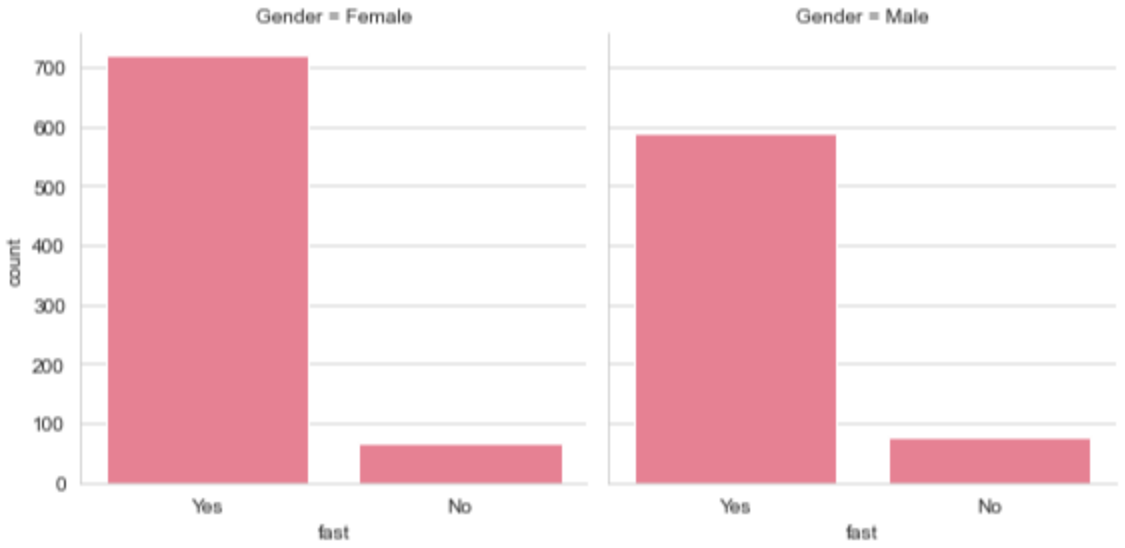
- Customers which found food `inconvenient` have most the time rated `I hate it!-5`
- Most of the customers who doesn't liked the food have given rating of `I hate it!-5`
- If the food is `disgusting` mostly `I hate it!-5` is given by the customers
- Those who `never visited` the store have given worst rating
- Customers who visited `once in a month` majority times rated `+3`
- Customers visiting `more than once a week` more likely to rate `I love it!+5`
- `Female` customers are more likley to rate `+3` where as males ratings are almost equally distributed

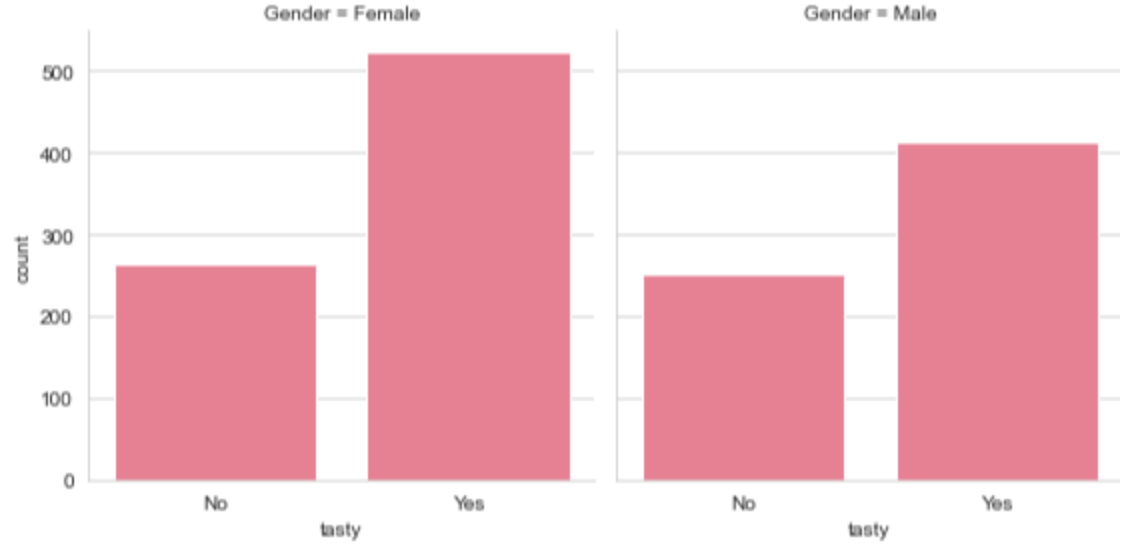
```
import warnings
warnings.filterwarnings("ignore")
```

```
sns.set_palette('husl')
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)
plt.savefig('count4.png')
plt.show()
```

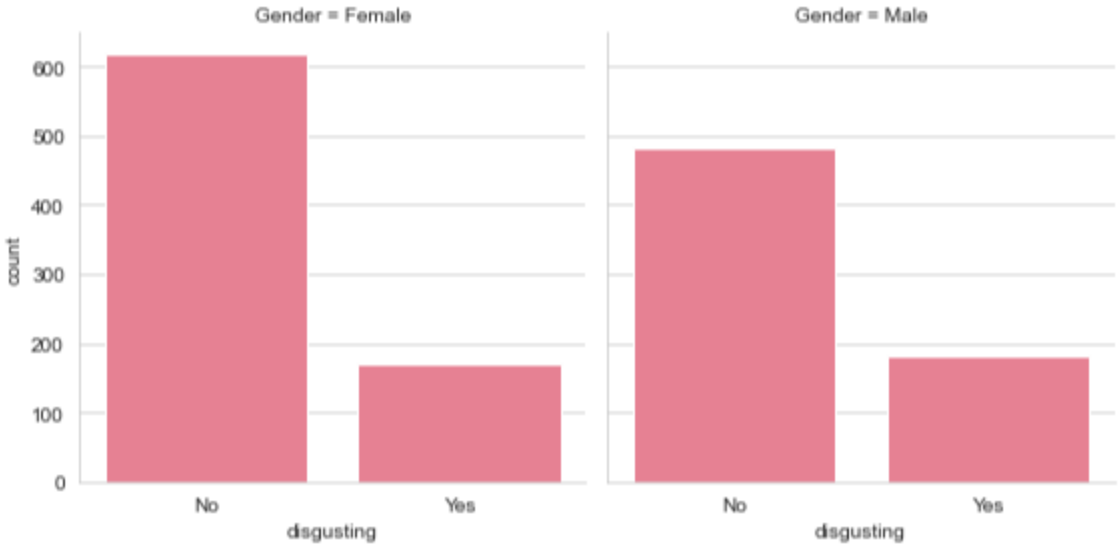


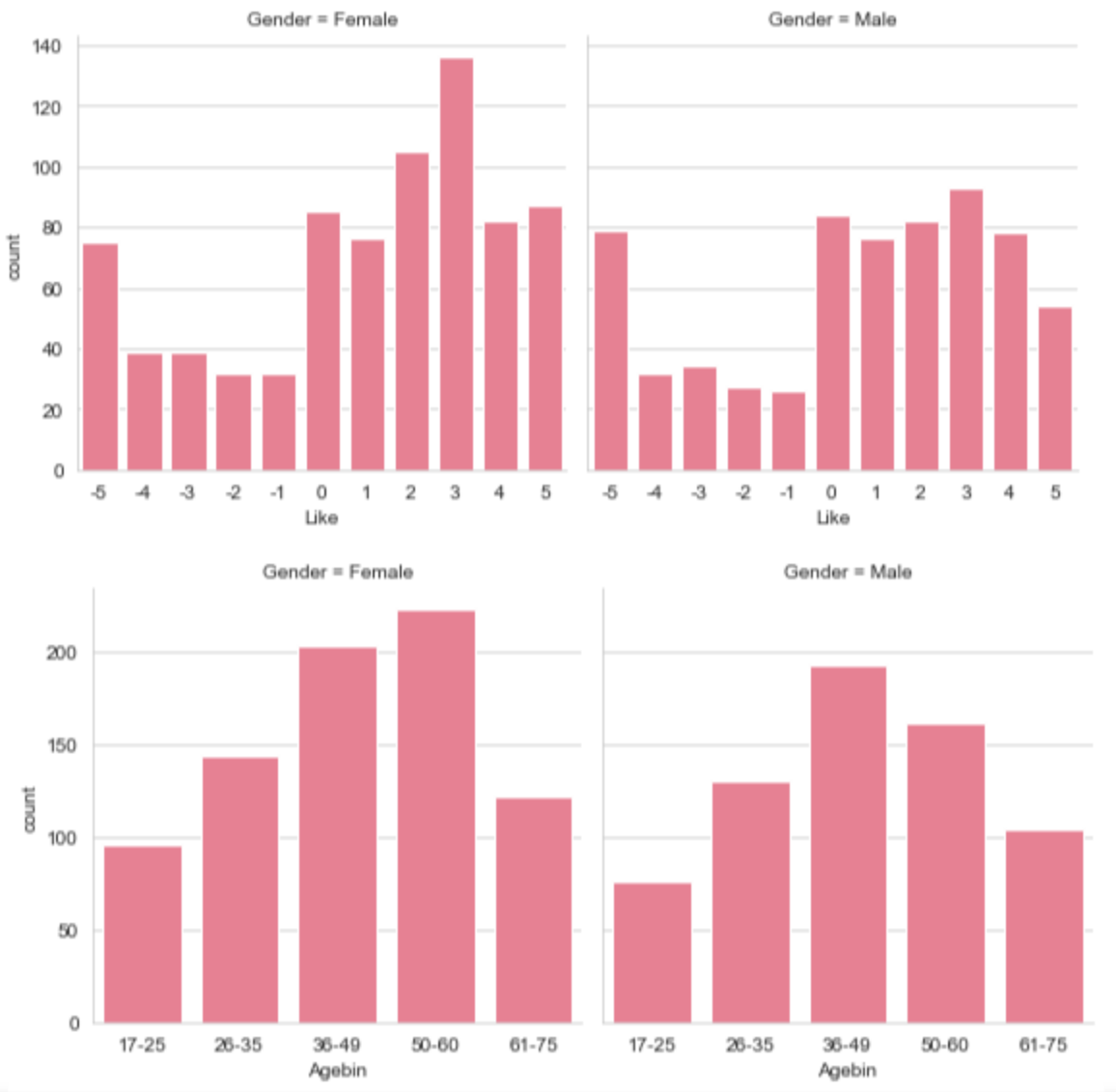

Gender = Female
Gender = Male
count
700
600
500
400
300
200
100
0
Yes
No
fattening


Gender = Female
Gender = Male
count
700
600
500
400
300
200
100
0
Yes
No
fast

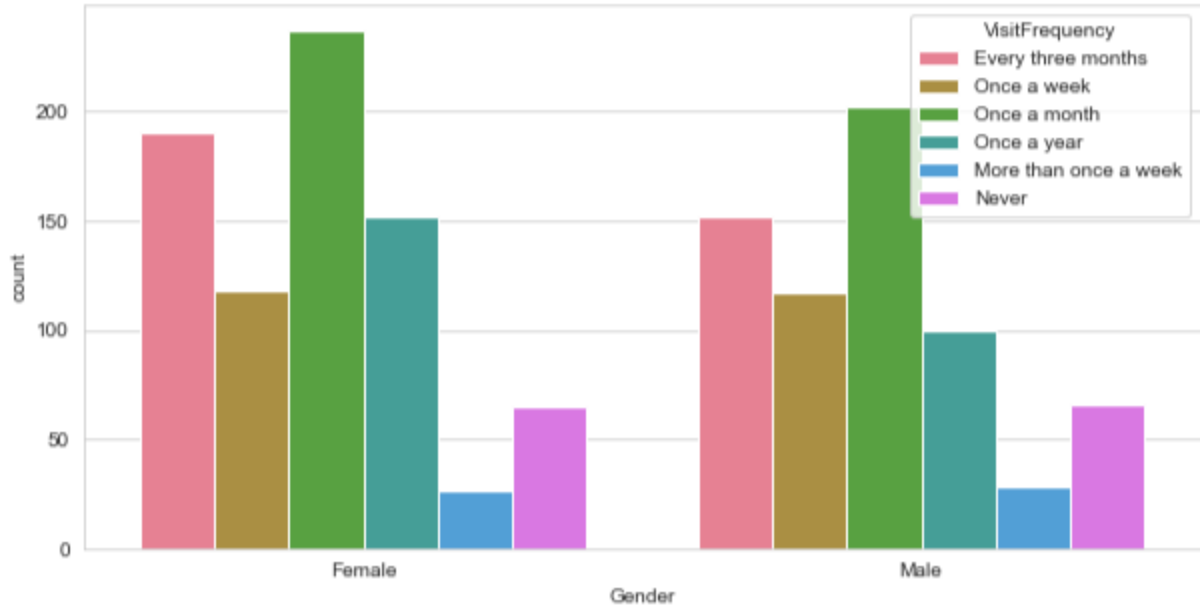

Gender = Female
Gender = Male
count
500
400
300
200
100
0
No
Yes
tasty


Gender = Female
Gender = Male
count
500
400
300
200
100
0
No
Yes
expensive


Gender = Female
Gender = Male
count
600
500
400
300
200
100
0
No
Yes
disgusting


Gender = Female
Gender = Male
count
Like
-5 -4 -3 -2 -1 0 1 2 3 4 5
0 20 40 60 80 100 120 140
Gender = Female
Gender = Male
count
Agebin
17-25 26-35 36-49 50-60 61-75
0 50 100 150 200

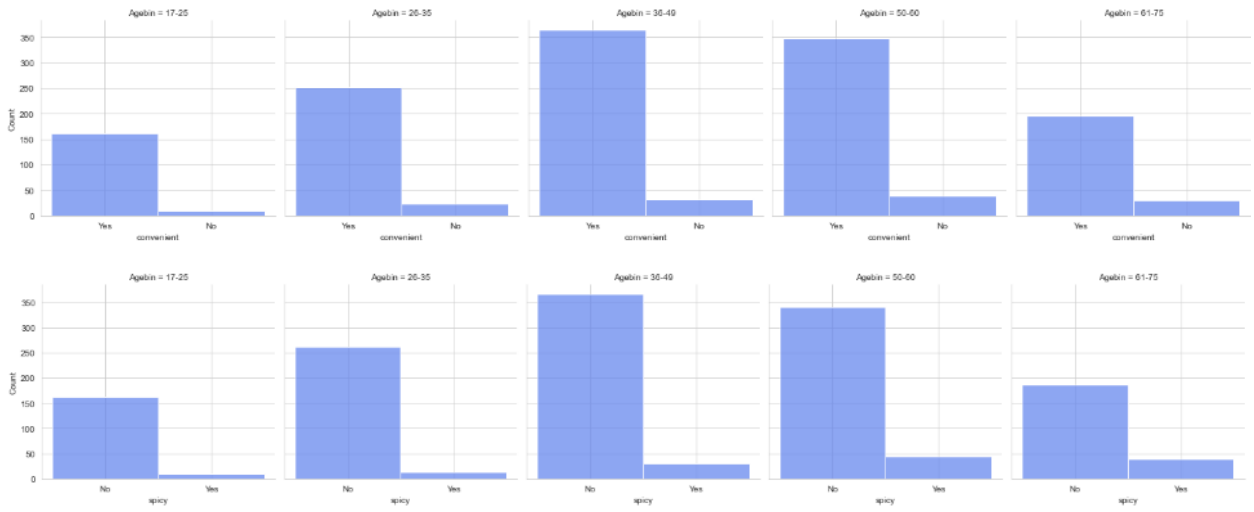
```
plt.figure(figsize=(10,5))
sns.countplot(hue=df['VisitFrequency'],x=df['Gender'])
plt.savefig('count5.png')
```



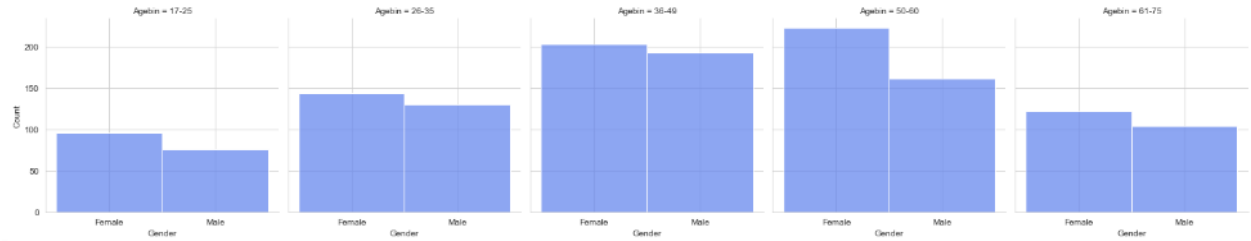

## Observations

- Female customers found it less convenient than male customers
- Majority of the female customers found the food expensive where as males doesn't
- Both the male and the female customers are almost alikly distributed

```
sns.set_palette('coolwarm')
for i in df.drop(['Agebin','yummy','cheap','healthy','greasy','Age','VisitFrequency'],axis=1):
  grid = sns.FacetGrid(df,height=4,col='Agebin')
  grid = grid.map(sns.histplot,i,bins=30)
plt.savefig('count6.png')
plt.show()
```




Agebin = 17-25
Agebin = 26-35
Agebin = 36-49
Agebin = 50-60
Agebin = 61-75
Count
fattening
fast
tasty
expensive
disgusting
Yes
No
Like




# Data Preprocessing

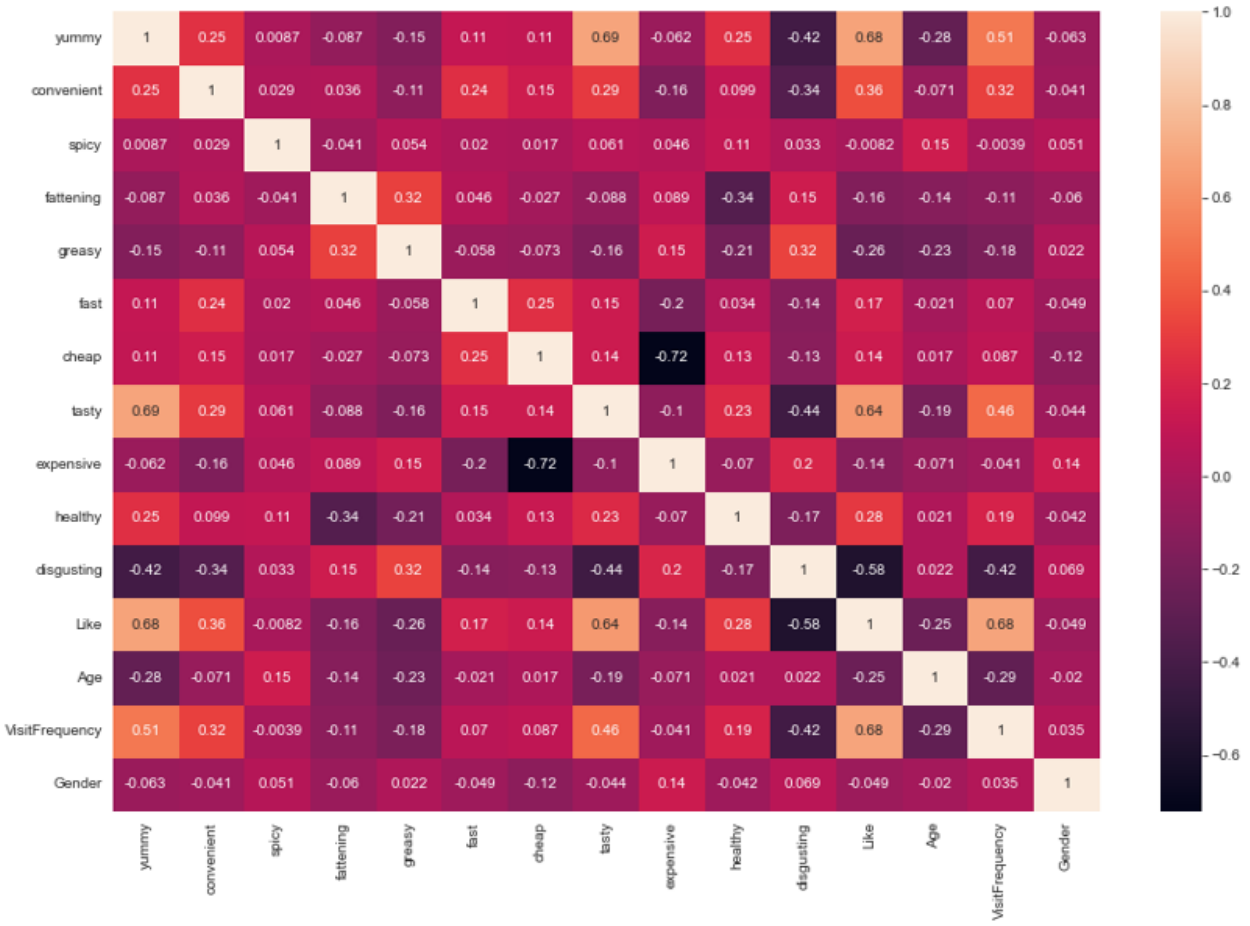
```
df['yummy'] = df['yummy'].replace(['Yes','No'],[1,0])
df['convenient'] = df['convenient'].replace(['Yes','No'],[1,0])
df['spicy'] = df['spicy'].replace(['Yes','No'],[1,0])
df['fattening'] = df['fattening'].replace(['Yes','No'],[1,0])
df['greasy'] = df['greasy'].replace(['Yes','No'],[1,0])
df['fast'] = df['fast'].replace(['Yes','No'],[1,0])
df['cheap'] = df['cheap'].replace(['Yes','No'],[1,0])
df['tasty'] = df['tasty'].replace(['Yes','No'],[1,0])
df['expensive'] = df['expensive'].replace(['Yes','No'],[1,0])
df['healthy'] = df['healthy'].replace(['Yes','No'],[1,0])
df['disgusting'] = df['disgusting'].replace(['Yes','No'],[1,0])
df['Gender'] = df['Gender'].replace(['Male','Female'],[1,0])
df['VisitFrequency'] = df['VisitFrequency'].replace(['Never','Once a year','Every three months','Once a month',
                                                     'Once a week','More than once a week'],[0,1,2,3,4,5])
df['Like'] = df['Like'].replace(['I hate it!-5','-4','-3','-2','-1','0','+1','+2','+3','+4','I love it!+5'],
                                [-5,-4,-3,-2,-1,0,1,2,3,4,5])
```

Convert the data to numeric form so that it can be understood by the ML model

```
df.head()
```

| | yummy | convenient | spicy | fattening | greasy | fast | cheap | tasty | expensive | healthy | disgusting | Like | Age | VisitFrequency | Gender | Agebin |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | -3 | 61 | 2 | 0 | 61-75 |
| **1** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 51 | 2 | 0 | 50-60 |
| **2** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 62 | 2 | 0 | 61-75 |
| **3** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 4 | 69 | 4 | 0 | 61-75 |
| **4** | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 2 | 49 | 3 | 1 | 36-49 |

```
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True)
plt.savefig('count7.png')
```




## Observations

- yummy is correlated with like and tasty
- expensive with cheap
- like is correlated with visitfrequency

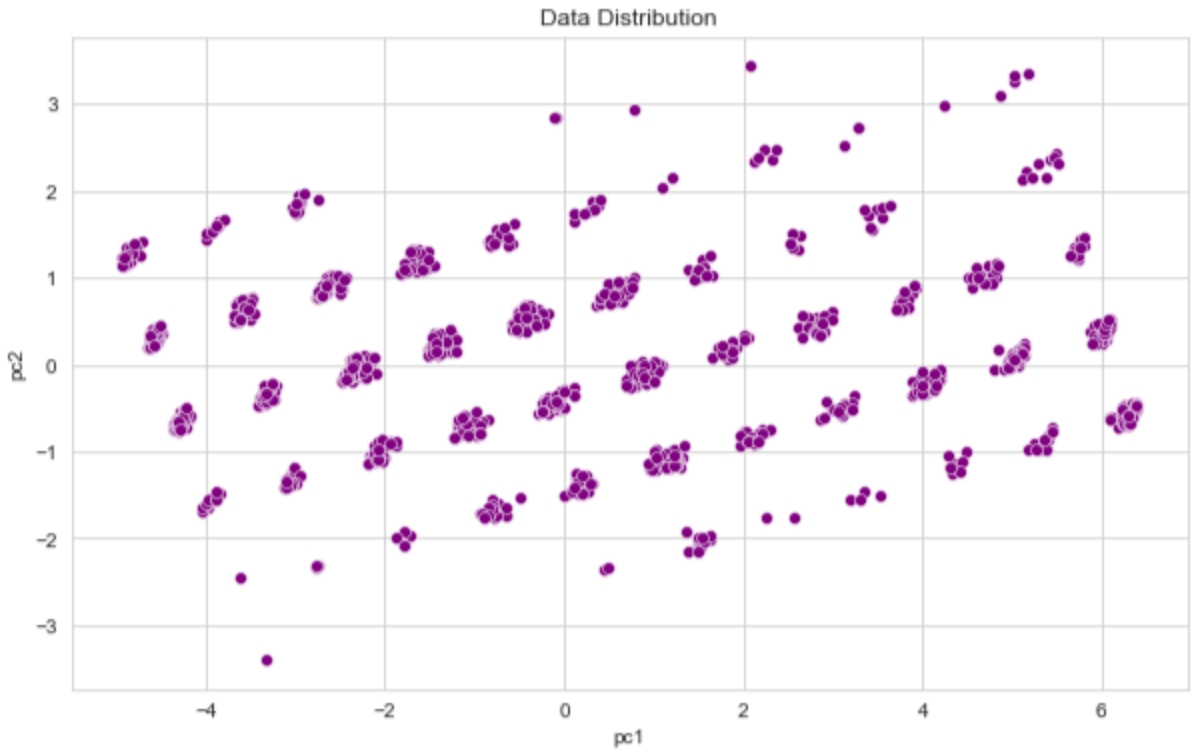
## Extract Segments

```
##Using k-means clustering

from sklearn.decomposition import PCA

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

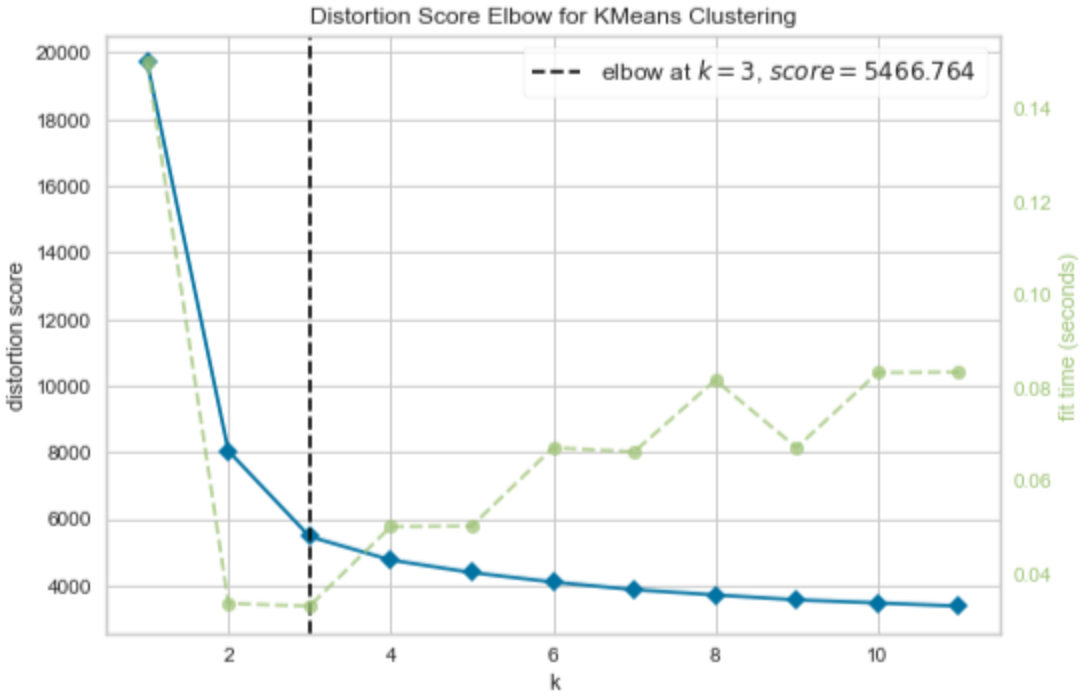
```
pc.head()
```

| | pc1 | pc2 | pc3 | pc4 | pc5 | pc6 | pc7 | pc8 | pc9 | pc10 | pc11 | pc12 | pc13 | pc14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3.734604 | 0.702304 | -0.327975 | 0.440305 | 0.620209 | -0.340562 | -0.322679 | -0.237330 | -0.380350 | -0.187596 | 0.139621 | 0.183905 | 0.538637 | -0.553612 |
| 1 | -1.125805 | -0.718494 | 0.239298 | -0.688357 | 0.332212 | 0.358268 | -0.144966 | -0.083182 | -0.080230 | -0.089313 | -0.036265 | 0.126230 | 0.507700 | -0.531808 |
| 2 | -0.093258 | -0.404862 | 0.744139 | -0.181747 | 0.536263 | 0.190031 | 0.722370 | -0.880779 | -0.632680 | 0.598609 | 0.320475 | -0.318969 | 0.068802 | 0.222549 |
| 3 | -3.531481 | 0.542226 | -0.316520 | -1.033631 | 0.061493 | -0.149492 | 0.278915 | 0.802097 | -0.095106 | 0.062222 | -0.224287 | -0.082179 | -0.214315 | -0.005139 |
| 4 | -1.265562 | 0.229772 | -0.330692 | -0.112704 | -0.800119 | -0.662628 | 0.769920 | -0.654509 | -0.105317 | -0.472214 | 0.209662 | -0.095196 | 0.024268 | 0.136550 |

```
plt.figure(figsize=(10,6))
sns.scatterplot(data=pc,x='pc1',y='pc2',color='purple')
plt.title('Data Distribution')
plt.savefig('count8.png')
```



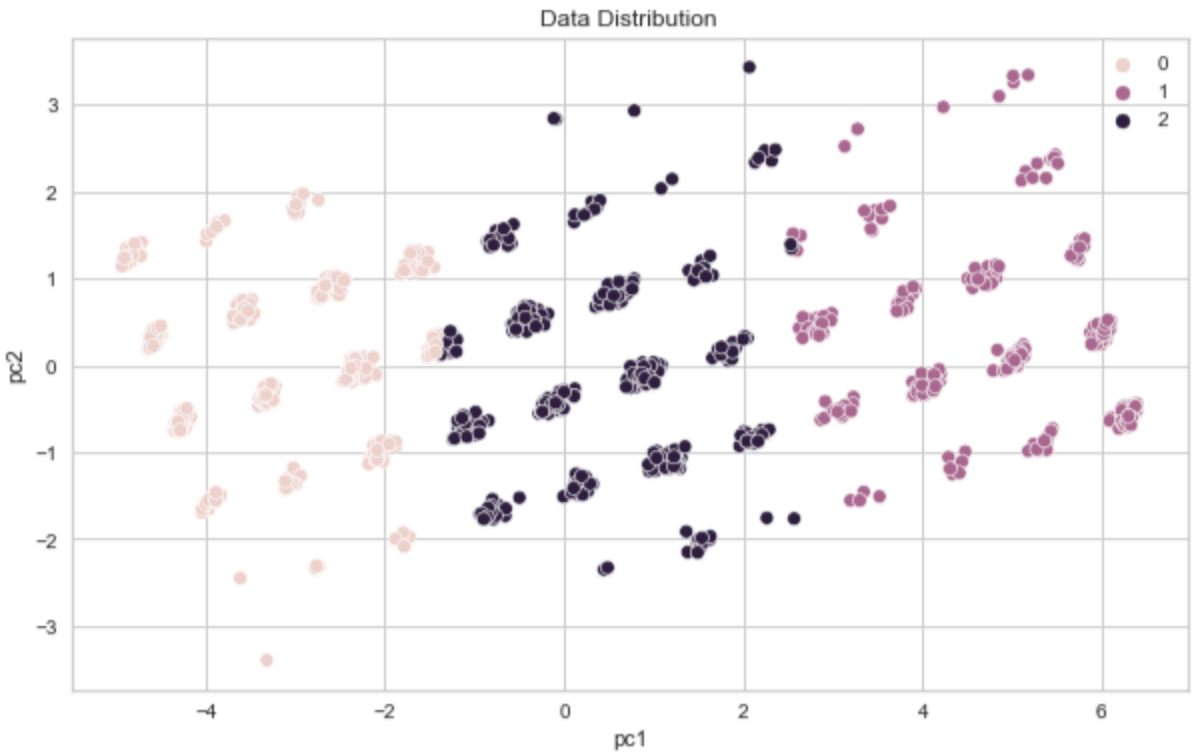
```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
kmeans = KMeans()
visualizer = KElbowVisualizer(kmeans, k=(1,12)).fit(pc)
visualizer.show()
plt.savefig('count9.png')
```




```
<Figure size 576x396 with 0 Axes>
```

Choosing 3 as value of k

```
# plotting the clusters
plt.figure(figsize=(10,6))
sns.scatterplot(x=pc['pc1'],y=pc['pc2'],hue=preds)
plt.title('Data Distribution')
plt.savefig('count10.png')
plt.show()
```




Explained Variance Ratio of PC1 = 0.8056
Explained Variance Ratio of PC2 = PC2 = 0.0603

**PC1 has a higher EV ratio**

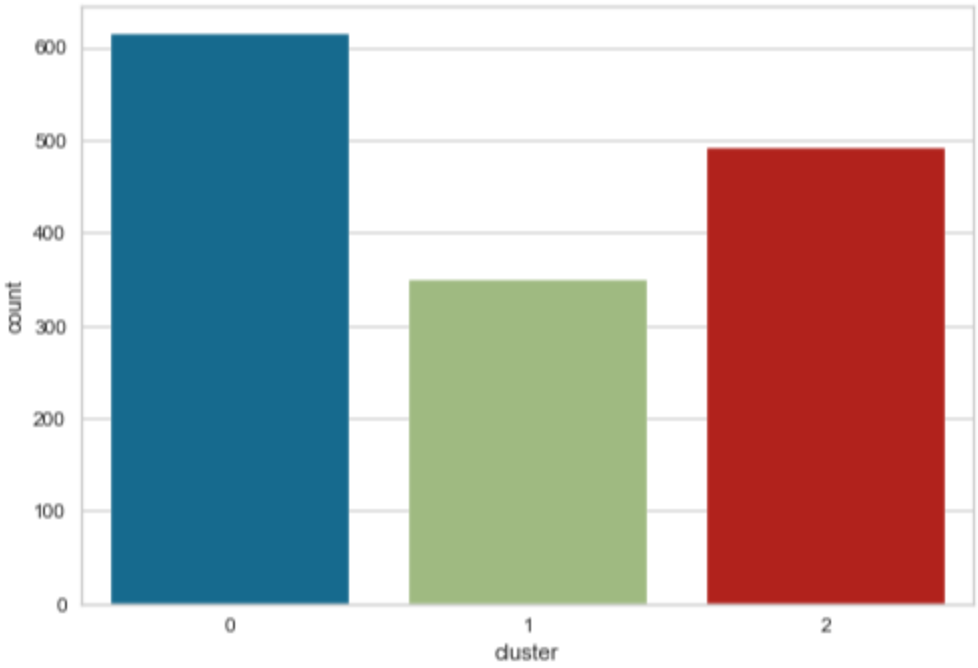
```
# training the model with 3 clusters
kmeans = KMeans(n_clusters=3)
kmeans.fit(pc)
```

```
KMeans
KMeans(n_clusters=3)
```

```
# predicting the clusters
np.random.seed(42)
preds = kmeans.predict(pc)
```

```
df['cluster'] = preds
```

```
sns.countplot(x = df['cluster'])
plt.savefig('count11.png')
```




```
df['cluster'].value_counts()/len(df)*100
```

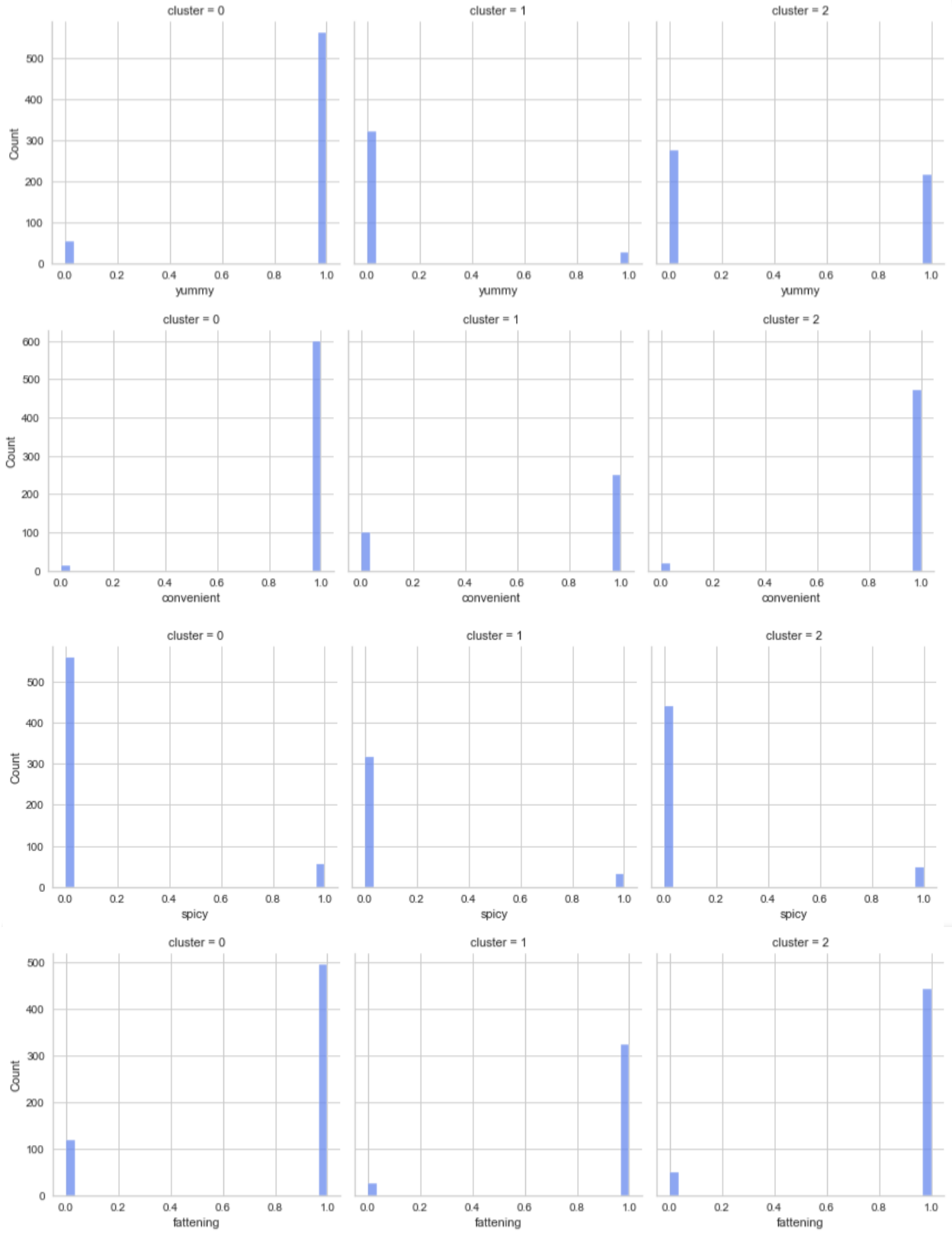
```
0    42.257398
2    33.792154
1    23.950447
Name: cluster, dtype: float64
```

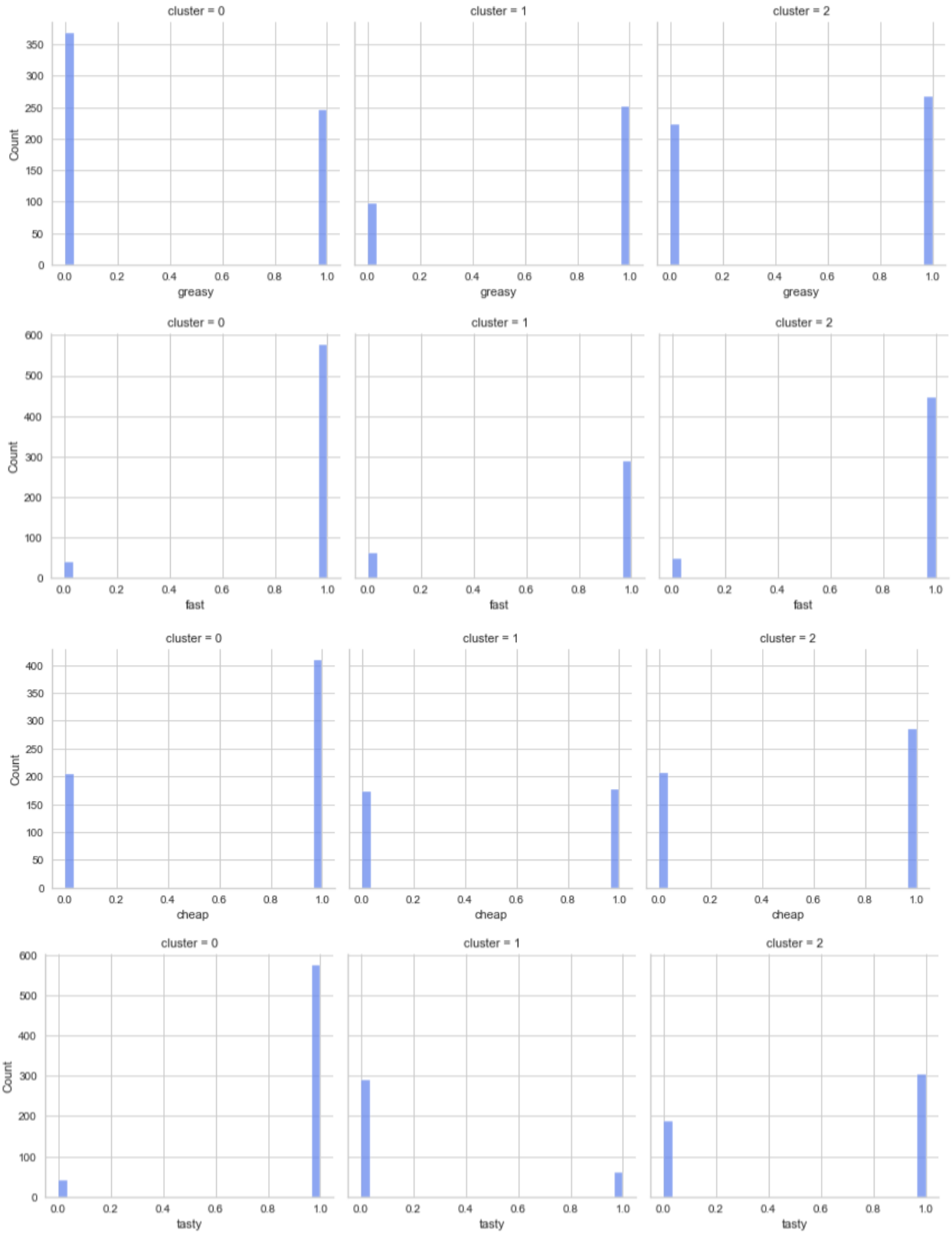
---

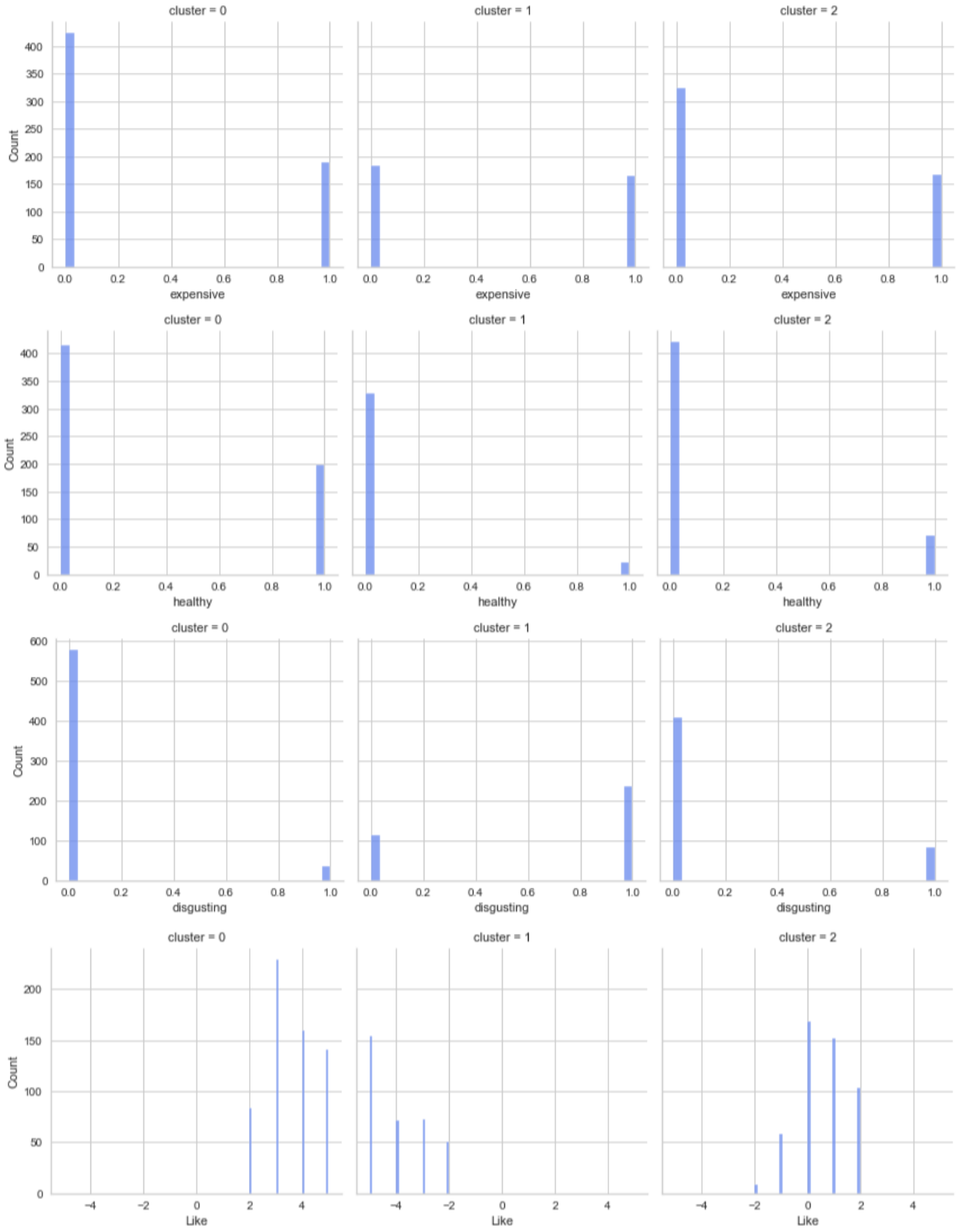
Observations

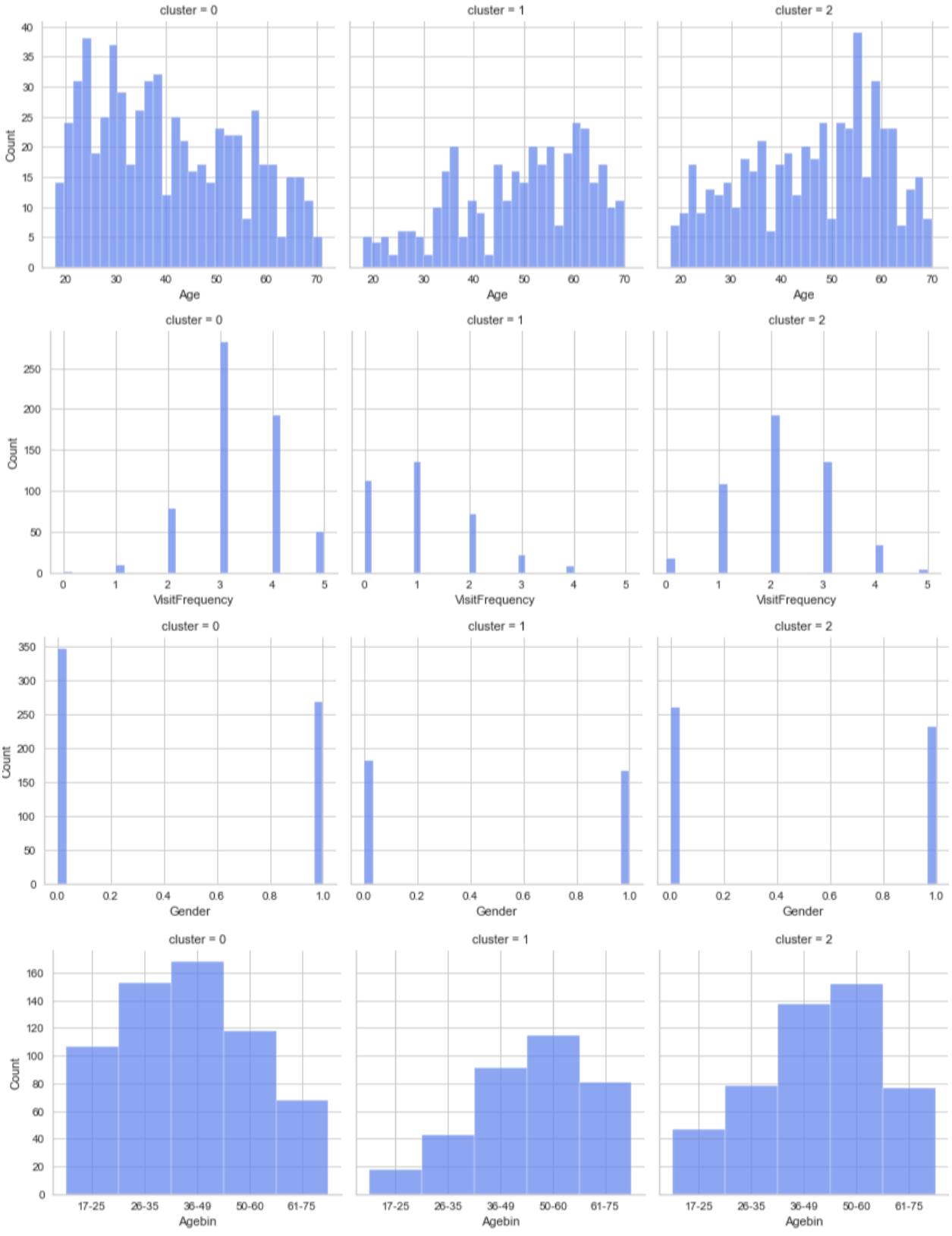
- maximum customers belongs to cluster 0
- approx 25 percent of the customers comes under cluster 0

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

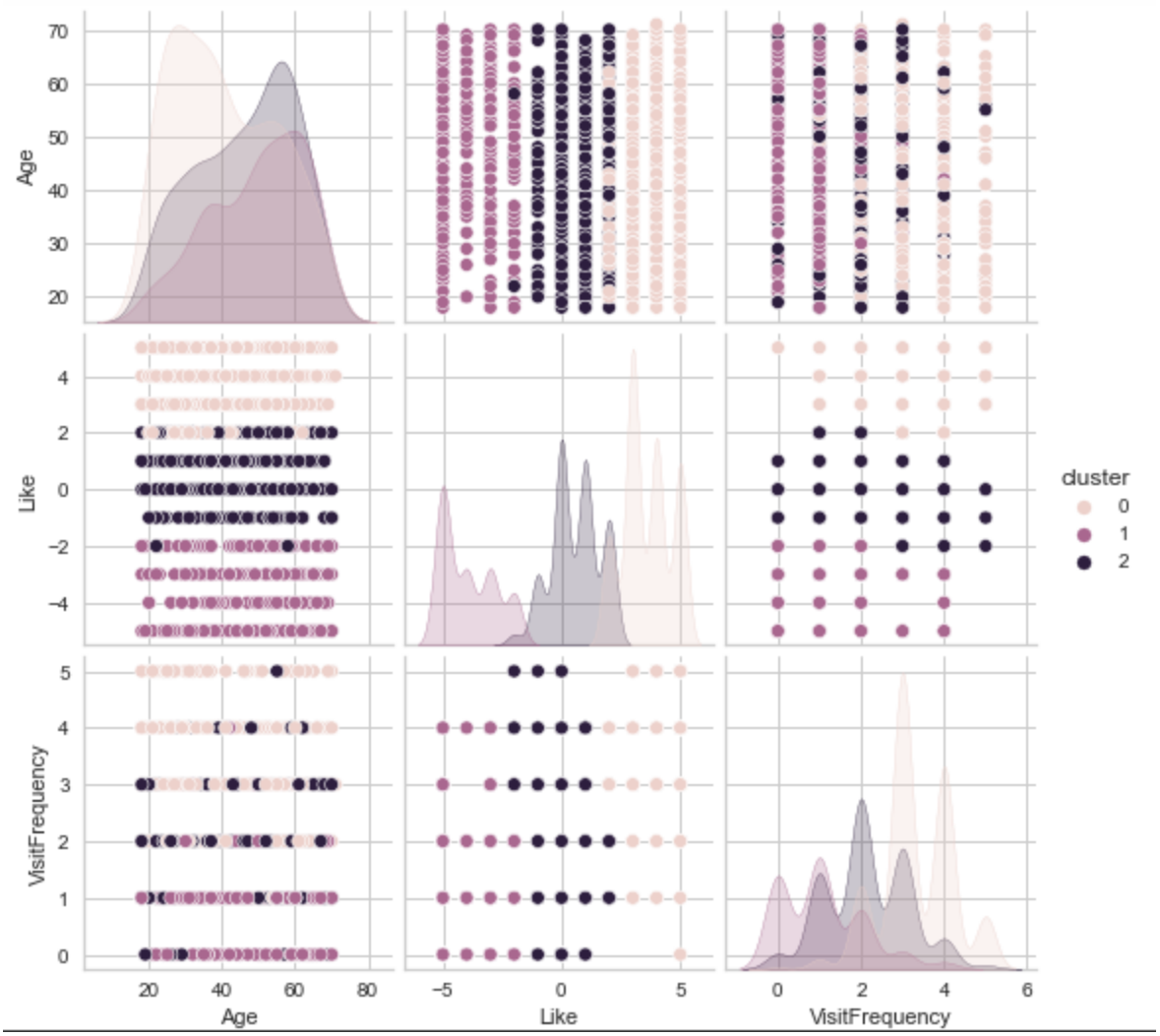

cluster = 0
cluster = 1
cluster = 2
Count
yummy
convenient
spicy
fattening
0.0
0.2
0.4
0.6
0.8
1.0


cluster = 0
cluster = 1
cluster = 2
Count
greasy
fast
cheap
tasty


cluster = 0
cluster = 1
cluster = 2
Count
expensive
healthy
disgusting
Like


cluster = 0
cluster = 1
cluster = 2
Count
Age
VisitFrequency
Gender
Agebin
17-25
26-35
36-49
50-60
61-75

```
# selecting Target variables

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



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

# Classification

```
from sklearn.model_selection import train_test_split
```

```
x = df.drop(['Agebin','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)
x_train.shape,y_train.shape
```

```
((1162, 15), (1162,))
```

```
## scaling the features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

```
#using logisitc regression for classification
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf.fit(x_train,y_train)

## predictions
preds = clf.predict(x_test)
```

```
df
```

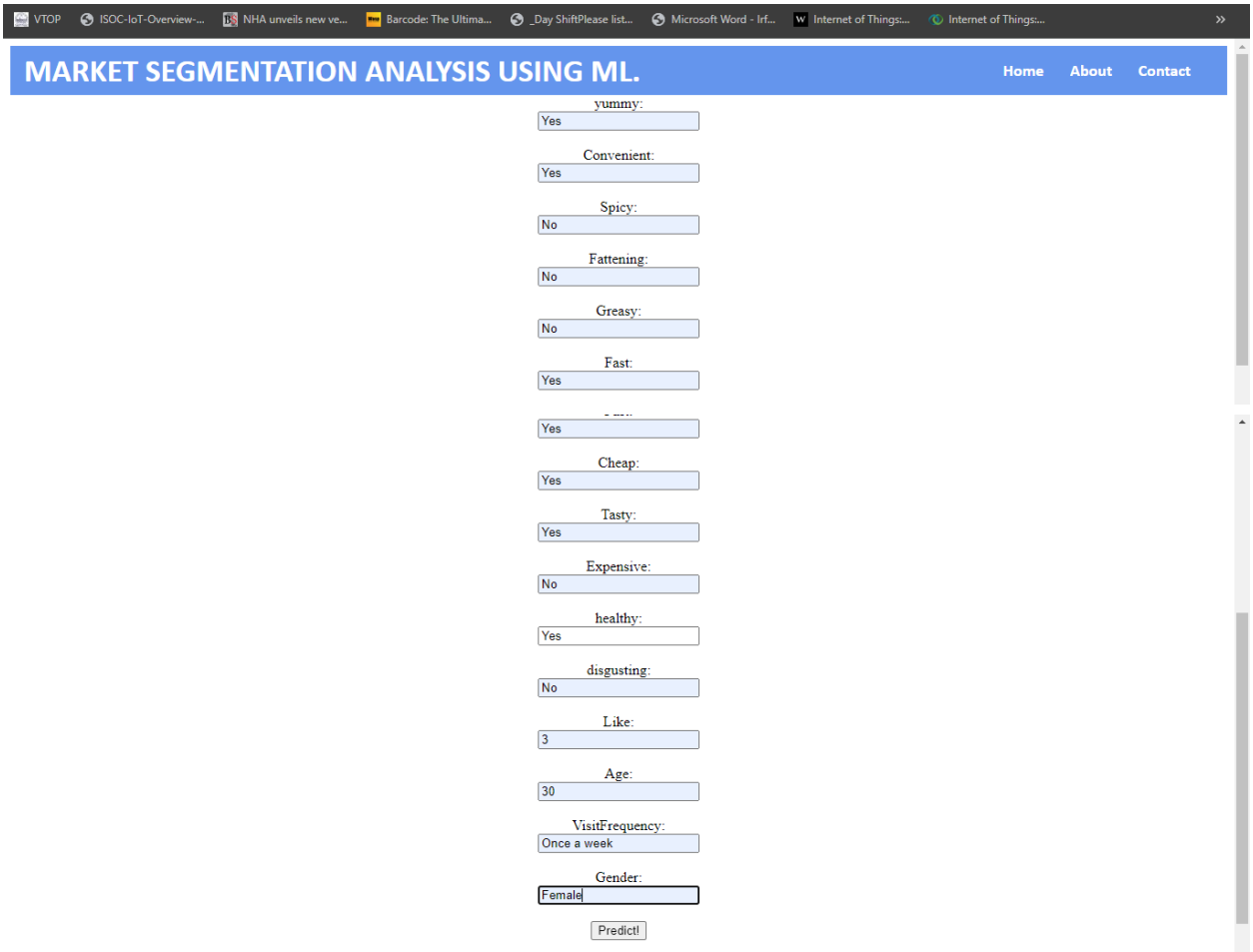
| | yummy | convenient | spicy | fattening | greasy | fast | cheap | tasty | expensive | healthy | disgusting | Like | Age | VisitFrequency | Gender | Agebin | cluster |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | -3 | 61 | 2 | 0 | 61-75 | 1 |
| **1** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 2 | 51 | 2 | 0 | 50-60 | 2 |
| **2** | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 62 | 2 | 0 | 61-75 | 2 |
| **3** | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 4 | 69 | 4 | 0 | 61-75 | 0 |
| **4** | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 2 | 49 | 3 | 1 | 36-49 | 2 |

```
result = clf.predict(scaler.transform([[1,1,0,0,0,1,1,1,0,1,0,3,30,4,0]]))
print("Cluster = ",result)
```

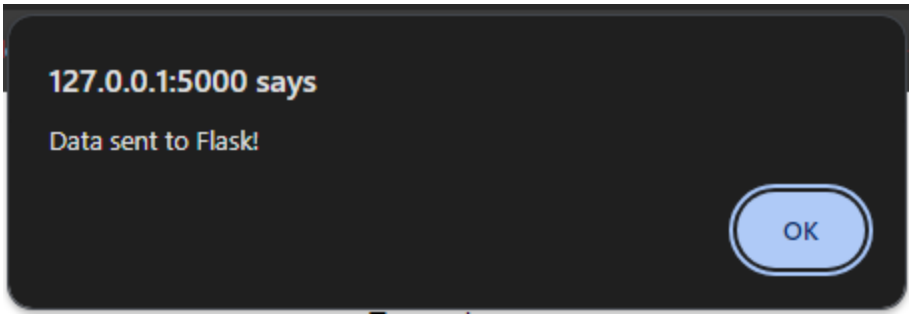
```
Cluster =  [0]
```

This is how we predict the cluster (customer segment)

## Web Application




Output:



Tasty:

Expensive:

healthy:

disgusting:

Like:

Age:

VisitFrequency:

Gender:

Predict!

The predicted cluster is 1

## Exception Handling:

yummy:

Really?

Convenient:

Yes

Spicy:

No

Fattening:

# ValueError

```
ValueError: could not convert string to float: 'Really?'
```

# Advantages and Disadvantages

## Advantages:

- Fast (the prediction and output are instantaneous)
- Very high accuracy of over 97%
- Clean, simple interface which identifies which variables are input

## Disadvantages:

- Not immediately apparent which values count as valid and which are not
- User must look up code to see the info of all three clusters
- Custom exceptions (such as age check) do not work

# Conclusion

This application is excellent for predicting the segment of a particular customer because of its high accuracy and quick execution. But it ignores certain invalid values such as a Like equal to 67 and an age under 17. If proper exceptions were implemented for these values, the code would not work and output would not appear. Hence they are neccessary evils. The user must also be able to read the model code and have knowledge of coding.