informe-prueba-cervecerias-v8

September 29, 2024

1 Informe Notebook Proyecto: Cerveceria casera en el mercado chileno

2 Contexto del caso:

En el rubro de la creación de cervezas cada vez es más conocido el tema de las cervezas casera, estas durante estos últimos años han aumentado en popularidad debido a lo innovador de sus sabores, por eso cerveceras grandes de renombre como Kross quieren tener una parte de ese negocio haciendo ellos sus propios sabores innovadores, para ello la empresa Kross nos pidió a nosotros como grupo DJJ que analicemos un csv con muestras de cervezas caseras estadounidenses para que veamos cuales fueron las mejores valoradas para integrarlas en el mercado chileno.

2.1 Fase 1: Comprender Negocio

Nuestro objetivo como grupo es analizar los datos de las cervezas estadounidenses para poder ver cuáles fueron las mejores calificadas, para esto ocuparemos diferentes métodos y gráficos para sacar en claro todo los datos necesarios.

además de responder las siguientes preguntas

¿Cuáles son los tipos de cerveza mejor evaluados?

¿Cuáles son las cervezas con mejor sabor valorado?

¿Cuál de las cervezas tiene más reviews dentro de los datos?

3 Importar librerias y el CSV

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
```

[2]:
```
file_path = 'C:/Users/massr/Desktop/reviewsCerveceriaKedro/cerveceriar/data/
↪01_raw/beer_reviews.csv'
df = pd.read_csv(file_path, sep=",")
```

4 Fase 2: Comprender los datos del CSV

Ahora que tenemos el csv cargado veremos la cantidad de datos y columnas además de analizar los datos que más se nos hagan útiles para el proyecto

[3]:
```
df
```

[3]:

	brewery_id	brewery_name	review_time	review_overall	review_aroma	review_appearance	review_profilename	beer_style	review_palate	review_taste
0	10325	Vecchio Birraio	1234817823	1.5	2.0	2.5	stcules	Hefeweizen	1.5	1.5
1	10325	Vecchio Birraio	1235915097	3.0	2.5	3.0	stcules	English Strong Ale	3.0	3.0
2	10325	Vecchio Birraio	1235916604	3.0	2.5	3.0	stcules	Foreign / Export Stout	3.0	3.0
3	10325	Vecchio Birraio	1234725145	3.0	3.0	3.5	stcules	German Pilsener	2.5	3.0
4	1075	Caldera Brewing Company	1293735206	4.0	4.5	4.0	johnmichaelsen	American Double / Imperial IPA	4.0	4.5
…	…	…	…	…	…	…	…	…	…	…
1586609	14359	The Defiant Brewing Company	1162684892	5.0	4.0	3.5	maddogruss	Pumpkin Ale	4.0	4.0
1586610	14359	The Defiant Brewing Company	1161048566	4.0	5.0	2.5	yelterdow	Pumpkin Ale	2.0	4.0
1586611	14359	The Defiant Brewing Company	1160702513	4.5	3.5	3.0	TongoRad			
1586612	14359	The Defiant Brewing Company	1160023044	4.0	4.5	4.5	dherling			
1586613	14359	The Defiant Brewing Company	1160005319	5.0	4.5	4.5	cbl2			

```
1586611                    Pumpkin Ale            3.5           4.0
1586612                    Pumpkin Ale            4.5           4.5
1586613                    Pumpkin Ale            4.5           4.5

                      beer_name  beer_abv  beer_beerid
0                   Sausa Weizen       5.0        47986
1                       Red Moon       6.2        48213
2        Black Horse Black Beer       6.5        48215
3                     Sausa Pils       5.0        47969
4                  Cauldron DIPA       7.7        64883
…                            …       …           …
1586609      The Horseman's Ale       5.2        33061
1586610      The Horseman's Ale       5.2        33061
1586611      The Horseman's Ale       5.2        33061
1586612      The Horseman's Ale       5.2        33061
1586613      The Horseman's Ale       5.2        33061

[1586614 rows x 13 columns]
```

ver número de filas y columnas del csv

[4]:
```
df.shape
```

[4]:
```
(1586614, 13)
```

ver columnas del csv

[5]:
```
df.columns
```

[5]:
```
Index(['brewery_id', 'brewery_name', 'review_time', 'review_overall',
       'review_aroma', 'review_appearance', 'review_profilename', 'beer_style',
       'review_palate', 'review_taste', 'beer_name', 'beer_abv',
       'beer_beerid'],
      dtype='object')
```

Resumen de los datos del csv

[6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1586614 entries, 0 to 1586613
Data columns (total 13 columns):
 #   Column              Non-Null Count    Dtype
---  ------              --------------    -----
 0   brewery_id          1586614 non-null  int64
 1   brewery_name        1586599 non-null  object
 2   review_time         1586614 non-null  int64
 3   review_overall      1586614 non-null  float64
 4   review_aroma        1586614 non-null  float64
```

```
 5   review_appearance   1586614 non-null  float64
 6   review_profilename  1586266 non-null  object
 7   beer_style          1586614 non-null  object
 8   review_palate       1586614 non-null  float64
 9   review_taste        1586614 non-null  float64
 10  beer_name           1586614 non-null  object
 11  beer_abv            1518829 non-null  float64
 12  beer_beerid         1586614 non-null  int64
dtypes: float64(6), int64(3), object(4)
memory usage: 157.4+ MB
```

Resumen estadistico de las columnas

[7]:
```
df.describe()
```

[7]:
```
         brewery_id   review_time  review_overall  review_aroma  \
count  1.586614e+06  1.586614e+06    1.586614e+06  1.586614e+06
mean   3.130099e+03  1.224089e+09    3.815581e+00  3.735636e+00
std    5.578104e+03  7.654427e+07    7.206219e-01  6.976167e-01
min    1.000000e+00  8.406720e+08    0.000000e+00  1.000000e+00
25%    1.430000e+02  1.173224e+09    3.500000e+00  3.500000e+00
50%    4.290000e+02  1.239203e+09    4.000000e+00  4.000000e+00
75%    2.372000e+03  1.288568e+09    4.500000e+00  4.000000e+00
max    2.800300e+04  1.326285e+09    5.000000e+00  5.000000e+00

       review_appearance  review_palate  review_taste      beer_abv  \
count       1.586614e+06   1.586614e+06  1.586614e+06  1.518829e+06
mean        3.841642e+00   3.743701e+00  3.792860e+00  7.042387e+00
std         6.160928e-01   6.822184e-01  7.319696e-01  2.322526e+00
min         0.000000e+00   1.000000e+00  1.000000e+00  1.000000e-02
25%         3.500000e+00   3.500000e+00  3.500000e+00  5.200000e+00
50%         4.000000e+00   4.000000e+00  4.000000e+00  6.500000e+00
75%         4.000000e+00   4.000000e+00  4.500000e+00  8.500000e+00
max         5.000000e+00   5.000000e+00  5.000000e+00  5.770000e+01

        beer_beerid
count  1.586614e+06
mean   2.171279e+04
std    2.181834e+04
min    3.000000e+00
25%    1.717000e+03
50%    1.390600e+04
75%    3.944100e+04
max    7.731700e+04
```

#Ver los datos que nos resulten más interesantes

Nombre de la cerveceria

[8]:
```
df.brewery_name
```

[8]:
```
0                      Vecchio Birraio
1                      Vecchio Birraio
2                      Vecchio Birraio
3                      Vecchio Birraio
4              Caldera Brewing Company
                       …
1586609    The Defiant Brewing Company
1586610    The Defiant Brewing Company
1586611    The Defiant Brewing Company
1586612    The Defiant Brewing Company
1586613    The Defiant Brewing Company
Name: brewery_name, Length: 1586614, dtype: object
```

nombre de las cervezas

[9]:
```
df.beer_name
```

[9]:
```
0                    Sausa Weizen
1                        Red Moon
2          Black Horse Black Beer
3                      Sausa Pils
4                  Cauldron DIPA
                   …
1586609        The Horseman's Ale
1586610        The Horseman's Ale
1586611        The Horseman's Ale
1586612        The Horseman's Ale
1586613        The Horseman's Ale
Name: beer_name, Length: 1586614, dtype: object
```

regusto en el paladar

[10]:
```
df.review_palate
```

[10]:
```
0          1.5
1          3.0
2          3.0
3          2.5
4          4.0
          …
1586609    4.0
1586610    2.0
1586611    3.5
1586612    4.5
1586613    4.5
Name: review_palate, Length: 1586614, dtype: float64
```

reseña general de la cerveza

[11]:
```
df.review_overall
```

[11]:
```
0          1.5
1          3.0
2          3.0
3          3.0
4          4.0
          …
1586609    5.0
1586610    4.0
1586611    4.5
1586612    4.0
1586613    5.0
Name: review_overall, Length: 1586614, dtype: float64
```

gusto de la cerveza

[12]:
```
df.review_taste
```

[12]:
```
0          1.5
1          3.0
2          3.0
3          3.0
4          4.5
          …
1586609    4.0
1586610    4.0
1586611    4.0
1586612    4.5
1586613    4.5
Name: review_taste, Length: 1586614, dtype: float64
```

apariencia de la cerveza

[13]:
```
df.review_appearance
```

[13]:
```
0          2.5
1          3.0
2          3.0
3          3.5
4          4.0
          …
1586609    3.5
1586610    2.5
1586611    3.0
1586612    4.5
```

```
1586613    4.5
Name: review_appearance, Length: 1586614, dtype: float64
```

grados de alcohol en la cerveza

[14]:
```
df.beer_abv
```

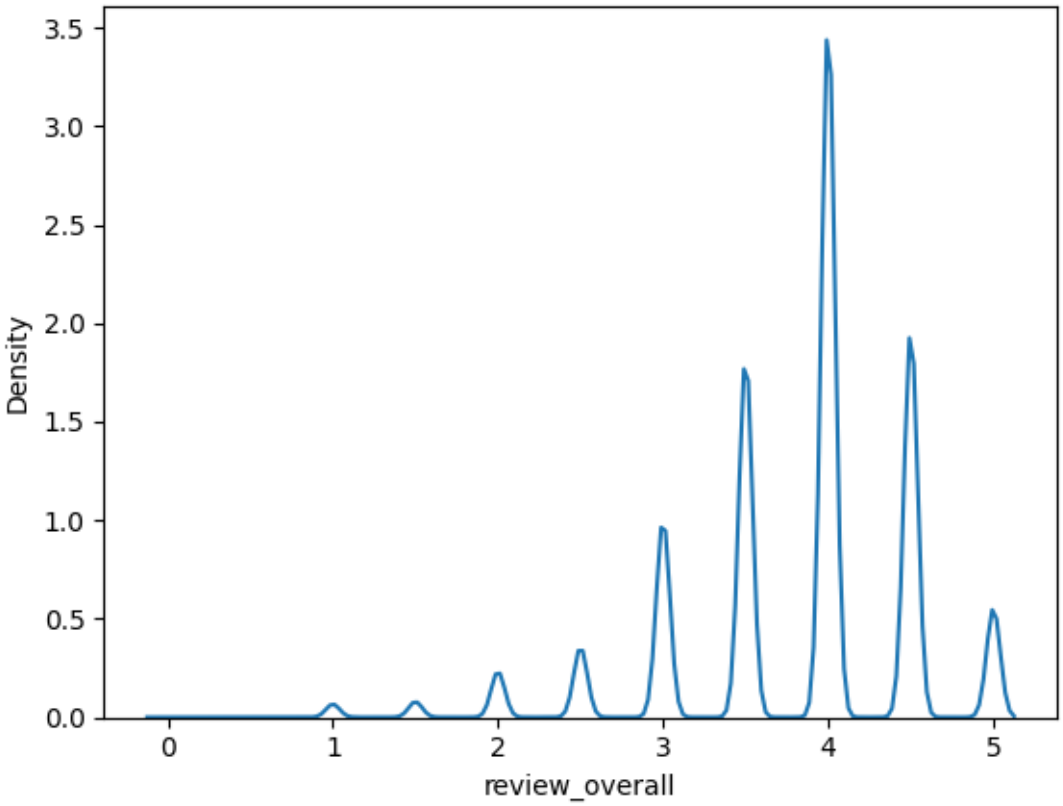
[14]:
```
0          5.0
1          6.2
2          6.5
3          5.0
4          7.7
          …
1586609    5.2
1586610    5.2
1586611    5.2
1586612    5.2
1586613    5.2
Name: beer_abv, Length: 1586614, dtype: float64
```

#Empezaremos con la graficación y análisis de los datos

En esta parte del proyecto nos encargamos de ver bien los datos que consideramos más factibles a tomar en este proyecto, en relación a las preguntas que planteamos en la parte 1.

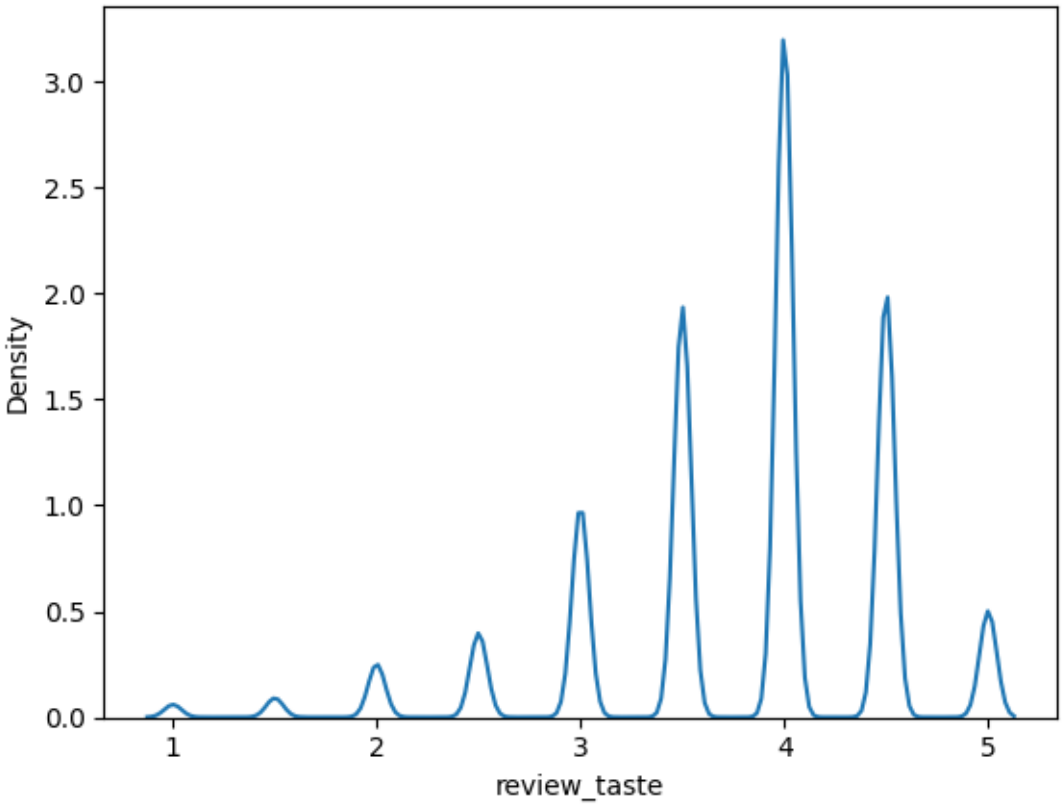
[15]:
```
from seaborn import kdeplot
kdeplot(df.review_overall)
```

[15]:
```
<Axes: xlabel='review_overall', ylabel='Density'>
```



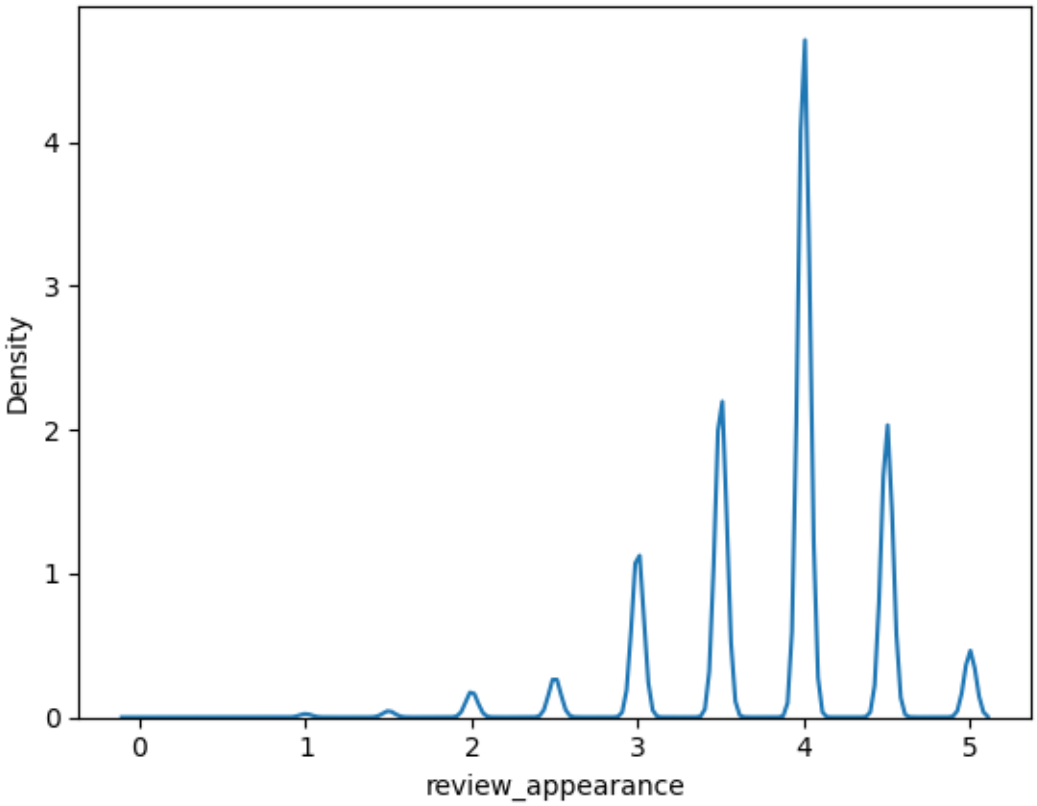
[16]:
```
from seaborn import kdeplot
kdeplot(df.review_taste)
```

[16]: <Axes: xlabel='review_taste', ylabel='Density'>



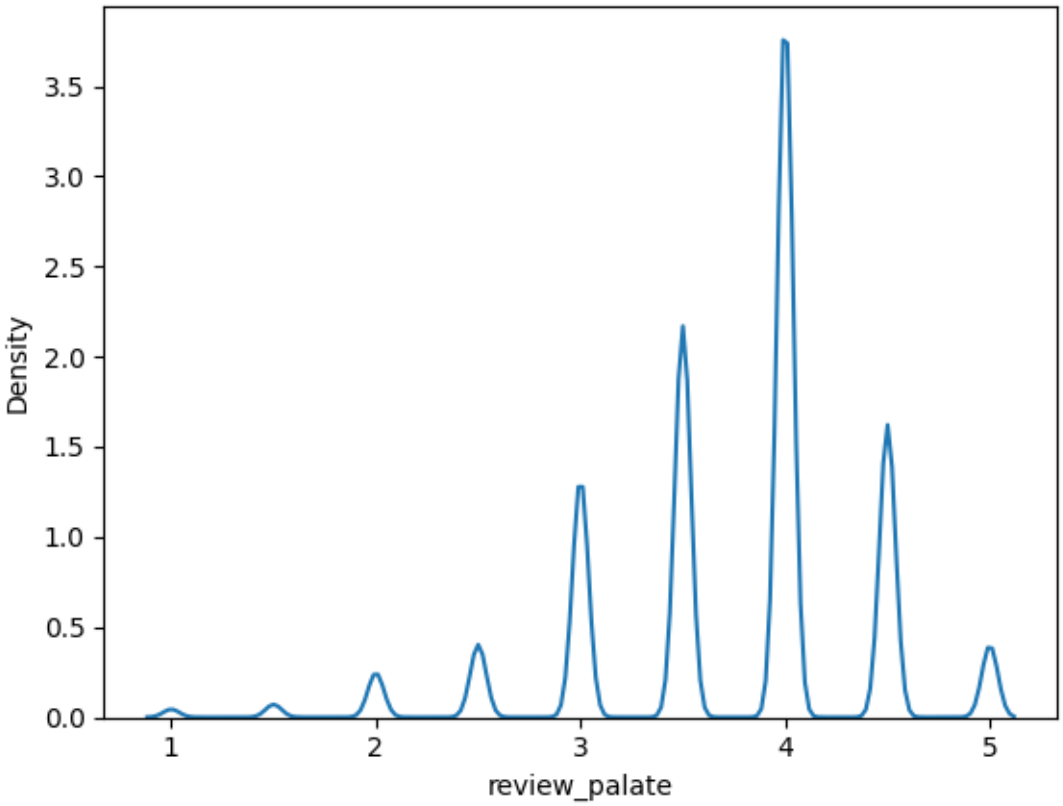
[17]:
```
from seaborn import kdeplot
kdeplot(df.review_appearance)
```

[17]: <Axes: xlabel='review_appearance', ylabel='Density'>



[18]:
```
from seaborn import kdeplot
kdeplot(df.review_palate)
```

[18]: <Axes: xlabel='review_palate', ylabel='Density'>



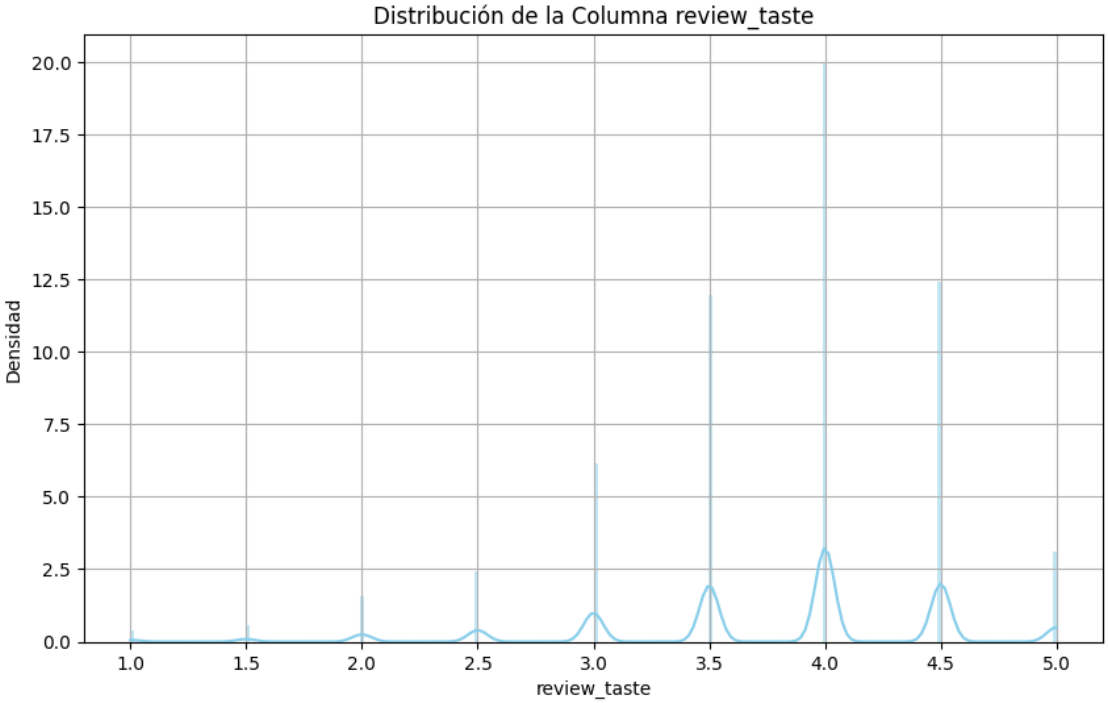
5 Identificando datos

Después de cargar el CSV decidimos que los datos que vamos a tomar para analizar seran review_overall que no ayudará a identificar a las cervezas que tienen una mejor reseña entre los catadores. El review_taste sera utilizado para identificar cuál de las cervezas tuvieron una mejor valoración en cuanto al sabor, el review_appearance y review_palate serán utilizado para ver cuales de las cervezas tiene una mejor apariencia y sabor en el paladar ya que como grupo que este es un factor también a tomar en cuenta. Además de todo esto se tendrá en cuenta a los datos de beer_abv para clasificar a las cervezas.

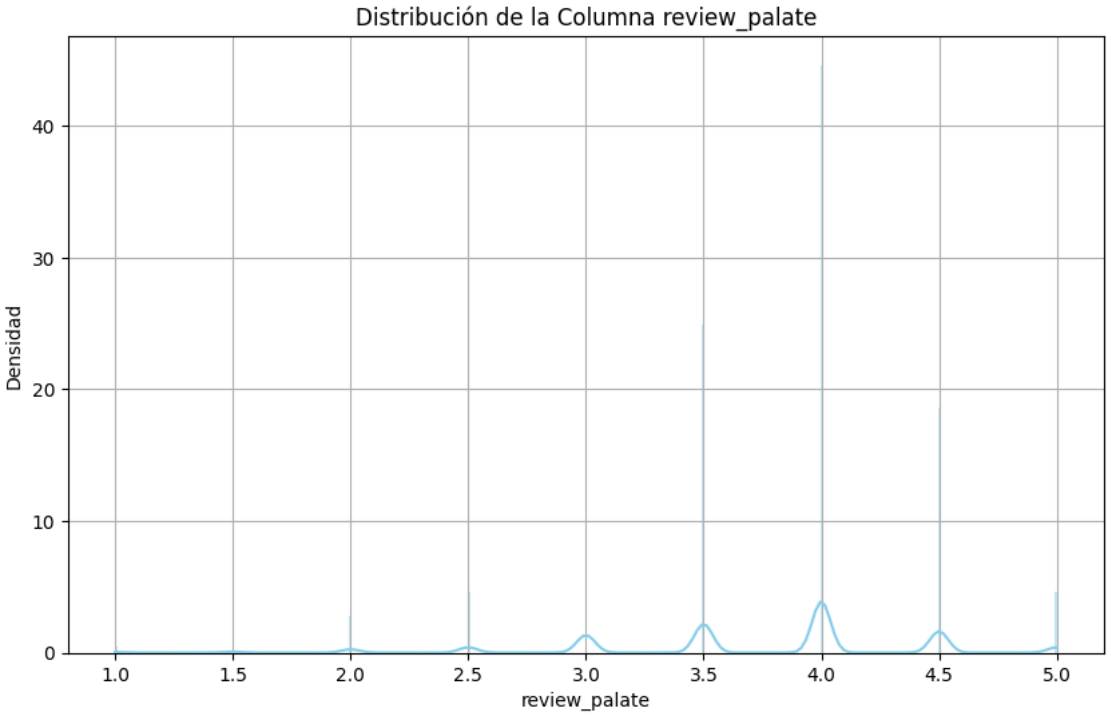
En los siguientes gráficos intentamos ver a más detalle la distribución de los anteriores datos que graficamos.

[19]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Extraer la columna de interés
data = df['review_taste']

# Crear el histograma con una distribución normal superpuesta
```

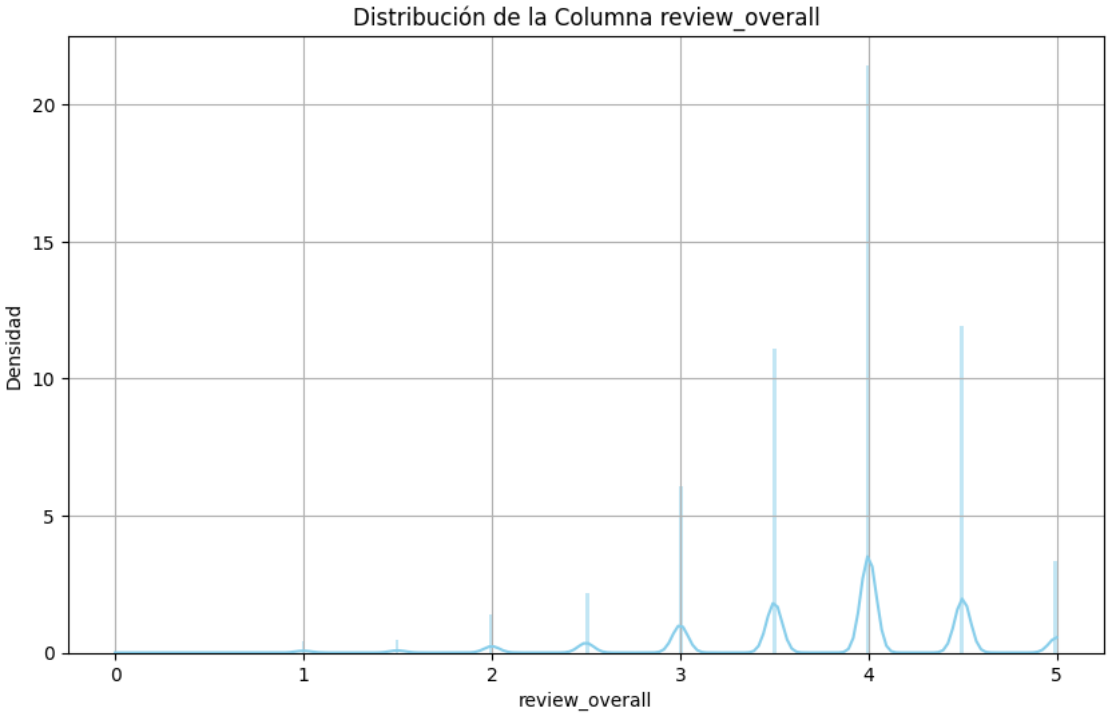
```
plt.figure(figsize=(10, 6))
sns.histplot(data, kde=True, stat="density", linewidth=0, color='skyblue')
plt.title('Distribución de la Columna review_taste')
plt.xlabel('review_taste')
plt.ylabel('Densidad')
plt.grid(True)
plt.show()
```



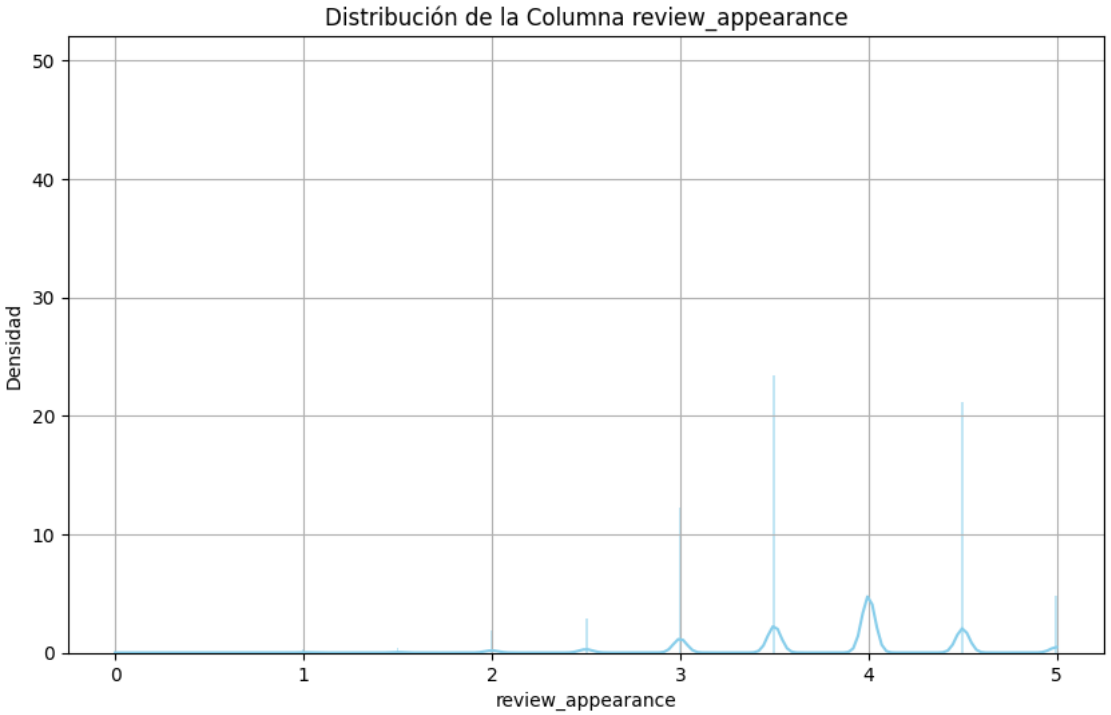
[20]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Extraer la columna de interés
data = df['review_palate']

# Crear el histograma con una distribución normal superpuesta
plt.figure(figsize=(10, 6))
sns.histplot(data, kde=True, stat="density", linewidth=0, color='skyblue')
plt.title('Distribución de la Columna review_palate')
plt.xlabel('review_palate')
plt.ylabel('Densidad')
plt.grid(True)
plt.show()
```



[21]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Extraer la columna de interés
data = df['review_overall']

# Crear el histograma con una distribución normal superpuesta
plt.figure(figsize=(10, 6))
sns.histplot(data, kde=True, stat="density", linewidth=0, color='skyblue')
plt.title('Distribución de la Columna review_overall')
plt.xlabel('review_overall')
plt.ylabel('Densidad')
plt.grid(True)
plt.show()
```



[22]:
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Extraer la columna de interés
data = df['review_appearance']

# Crear el histograma con una distribución normal superpuesta
plt.figure(figsize=(10, 6))
sns.histplot(data, kde=True, stat="density", linewidth=0, color='skyblue')
plt.title('Distribución de la Columna review_appearance')
plt.xlabel('review_appearance')
plt.ylabel('Densidad')
plt.grid(True)
plt.show()
```



6 Datos de tendencia central

en esta parte con la información que graficamos anteriormente no era suficiente así que queríamos tener a detalle de la media, moda y medina de los datos ya analizados

[24]:
```
#Codigo que analiza la media, mediana y moda de los datos
import pandas as pd
from scipy import stats


# codigo que ayuda a calcular la media de las rewies
media = df['review_overall'].mean()
print(f'Media: {media}')

# Codigo que ayuda a calcular la mediana de las rewies
mediana = df['review_overall'].median()
print(f'Mediana: {mediana}')

# Codigo que se usa para medir la moda de las rewies
moda = df['review_overall'].mode()
print(f'Moda: {moda.values}')

# Estadísticas descriptivas adicionales
```

```
estadisticas = df['review_overall'].describe()
print('\nEstadísticas descriptivas:')
print(estadisticas)
```

```
Media: 3.8155808533140387
Mediana: 4.0
Moda: [4.]

Estadísticas descriptivas:
count    1.586614e+06
mean     3.815581e+00
std      7.206219e-01
min      0.000000e+00
25%      3.500000e+00
50%      4.000000e+00
75%      4.500000e+00
max      5.000000e+00
Name: review_overall, dtype: float64
```

[25]:
```
#Codigo que analiza la media, mediana y moda de los datos
import pandas as pd
from scipy import stats


# codigo que ayuda a calcular la media del sabor
media = df['review_taste'].mean()
print(f'Media: {media}')

# Codigo que ayuda a calcular la mediana del sabor
mediana = df['review_taste'].median()
print(f'Mediana: {mediana}')

# Codigo que se usa para medir la moda del sabor
moda = df['review_taste'].mode()
print(f'Moda: {moda.values}')

# Estadísticas descriptivas adicionales
estadisticas = df['review_taste'].describe()
print('\nEstadísticas descriptivas:')
print(estadisticas)
```

```
Media: 3.792860456292457
Mediana: 4.0
Moda: [4.]

Estadísticas descriptivas:
count    1.586614e+06
mean     3.792860e+00
```

```
std      7.319696e-01
min      1.000000e+00
25%      3.500000e+00
50%      4.000000e+00
75%      4.500000e+00
max      5.000000e+00
Name: review_taste, dtype: float64
```

[26]:
```python
#Codigo que analiza la media, mediana y moda de los datos
import pandas as pd
from scipy import stats


# codigo que ayuda a calcular la media de la apariencia
media = df['review_appearance'].mean()
print(f'Media: {media}')

# Codigo que ayuda a calcular la mediana de la apariencia
mediana = df['review_appearance'].median()
print(f'Mediana: {mediana}')

# Codigo que se usa para medir la moda de la apariencia
moda = df['review_appearance'].mode()
print(f'Moda: {moda.values}')

# Estadísticas descriptivas adicionales
estadisticas = df['review_appearance'].describe()
print('\nEstadísticas descriptivas:')
print(estadisticas)
```

```
Media: 3.8416416973504584
Mediana: 4.0
Moda: [4.]

Estadísticas descriptivas:
count    1.586614e+06
mean     3.841642e+00
std      6.160928e-01
min      0.000000e+00
25%      3.500000e+00
50%      4.000000e+00
75%      4.000000e+00
max      5.000000e+00
Name: review_appearance, dtype: float64
```

[27]:
```python
#Codigo que analiza la media, mediana y moda de los datos
import pandas as pd
```

```
from scipy import stats


# codigo que ayuda a calcular la media de la apariencia
media = df['review_palate'].mean()
print(f'Media: {media}')

# Codigo que ayuda a calcular la mediana de la apariencia
mediana = df['review_palate'].median()
print(f'Mediana: {mediana}')

# Codigo que se usa para medir la moda de la apariencia
moda = df['review_palate'].mode()
print(f'Moda: {moda.values}')

# Estadísticas descriptivas adicionales
estadisticas = df['review_palate'].describe()
print('\nEstadísticas descriptivas:')
print(estadisticas)
```

```
Media: 3.7437013665579655
Mediana: 4.0
Moda: [4.]

Estadísticas descriptivas:
count    1.586614e+06
mean     3.743701e+00
std      6.822184e-01
min      1.000000e+00
25%      3.500000e+00
50%      4.000000e+00
75%      4.000000e+00
max      5.000000e+00
Name: review_palate, dtype: float64
```

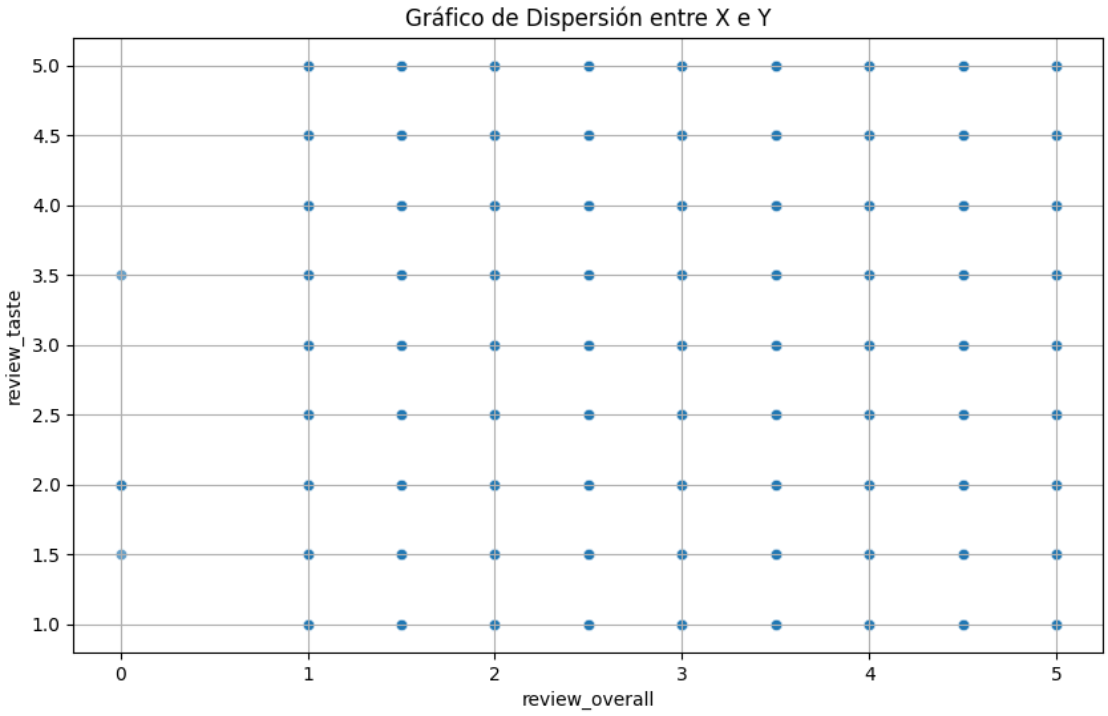
7 Gráfico de dispersión

En esta parte del proyecto tomamos los datos que analizamos anteriormente y los graficamos de tal forma que podamos ver cuales son los datos que destacan del resto.

[28]:
```
# Importar las bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

varianza_x = df['review_overall'].var()
```

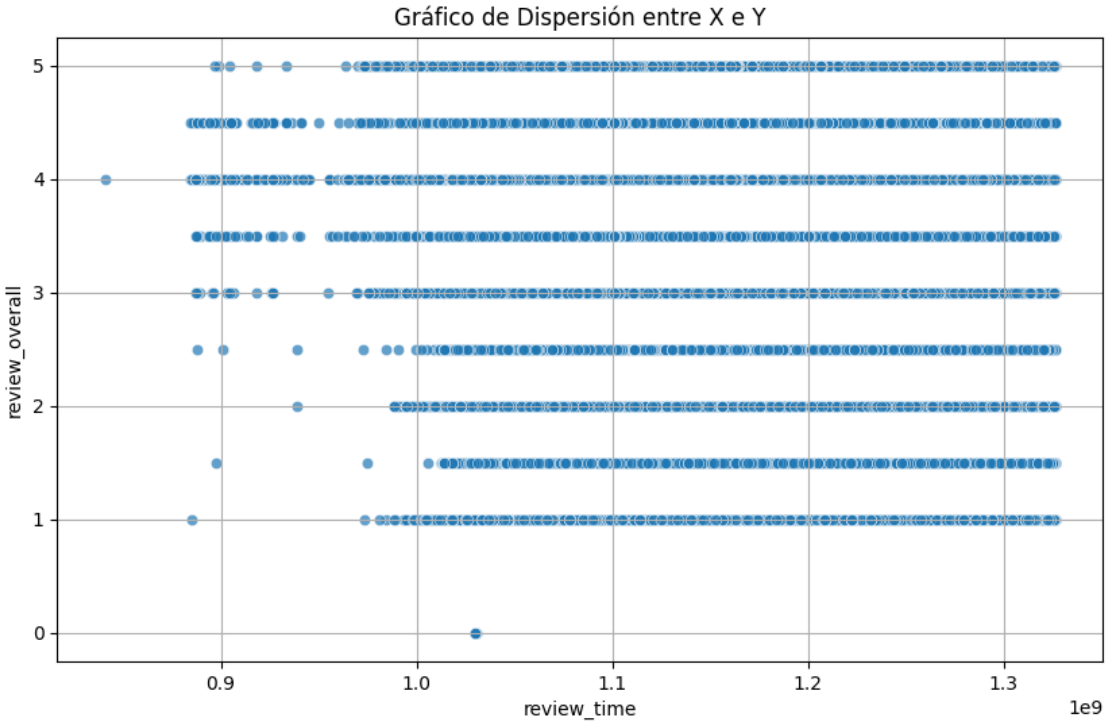
```python
desviacion_x = df['review_overall'].std()

varianza_y = df['review_taste'].var()
desviacion_y = df['review_taste'].std()

print(f'Varianza de X: {varianza_x}')
print(f'Desviación estándar de X: {desviacion_x}')
print(f'Varianza de Y: {varianza_y}')
print(f'Desviación estándar de Y: {desviacion_y}')

# Crear un gráfico de dispersión
plt.figure(figsize=(10, 6))
sns.scatterplot(x='review_overall', y='review_taste', data=df, alpha=0.7)
plt.title('Gráfico de Dispersión entre X e Y')
plt.xlabel('review_overall')
plt.ylabel('review_taste')
plt.grid(True)
plt.show()
```

```
Varianza de X: 0.519295876714356
Desviación estándar de X: 0.7206218680517238
Varianza de Y: 0.5357795098053175
Desviación estándar de Y: 0.7319696098919117
```



[29]:
```python
# Importar las bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

varianza_x = df['review_time'].var()
desviacion_x = df['review_time'].std()

varianza_y = df['review_overall'].var()
desviacion_y = df['review_overall'].std()

print(f'Varianza de X: {varianza_x}')
print(f'Desviación estándar de X: {desviacion_x}')
print(f'Varianza de Y: {varianza_y}')
print(f'Desviación estándar de Y: {desviacion_y}')

# Crear un gráfico de dispersión
plt.figure(figsize=(10, 6))
sns.scatterplot(x='review_time', y='review_overall', data=df, alpha=0.7)
plt.title('Gráfico de Dispersión entre X e Y')
plt.xlabel('review_time')
plt.ylabel('review_overall')
plt.grid(True)
plt.show()
```

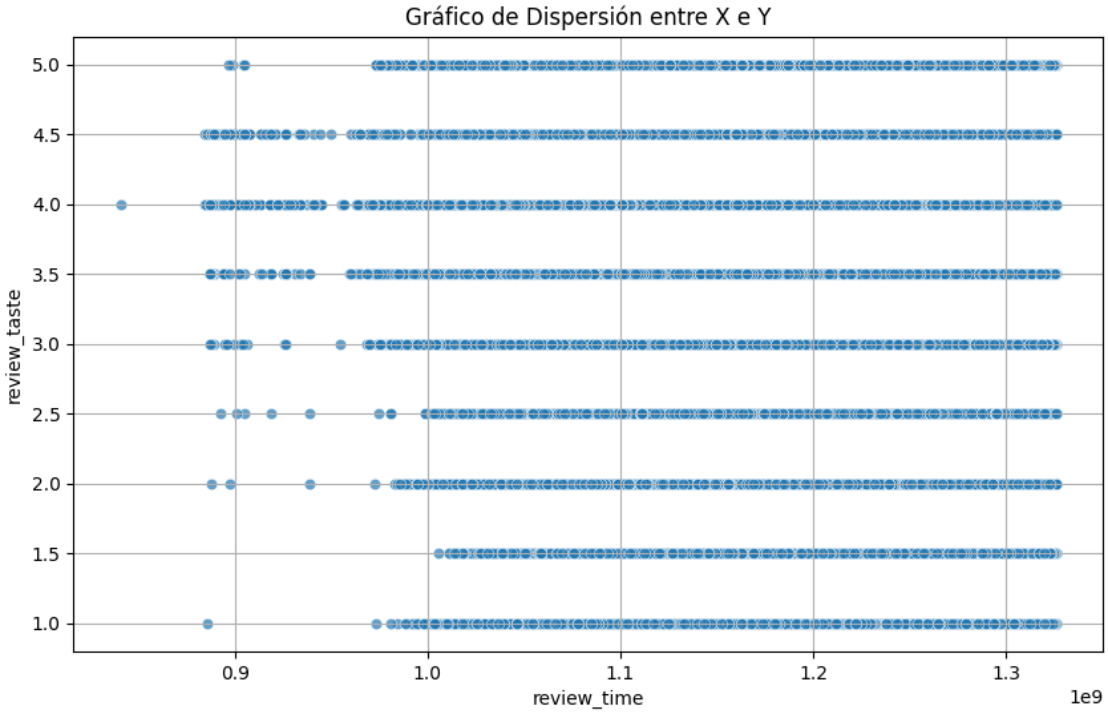
```
Varianza de X: 5859025964738176.0
Desviación estándar de X: 76544274.53923759
Varianza de Y: 0.519295876714356
Desviación estándar de Y: 0.7206218680517238
```



[30]:
```python
# Importar las bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

varianza_x = df['review_time'].var()
desviacion_x = df['review_time'].std()

varianza_y = df['review_taste'].var()
desviacion_y = df['review_taste'].std()

print(f'Varianza de X: {varianza_x}')
print(f'Desviación estándar de X: {desviacion_x}')
print(f'Varianza de Y: {varianza_y}')
print(f'Desviación estándar de Y: {desviacion_y}')

# Crear un gráfico de dispersión
plt.figure(figsize=(10, 6))
sns.scatterplot(x='review_time', y='review_taste', data=df, alpha=0.7)
plt.title('Gráfico de Dispersión entre X e Y')
plt.xlabel('review_time')
plt.ylabel('review_taste')
```

```
plt.grid(True)
plt.show()
```

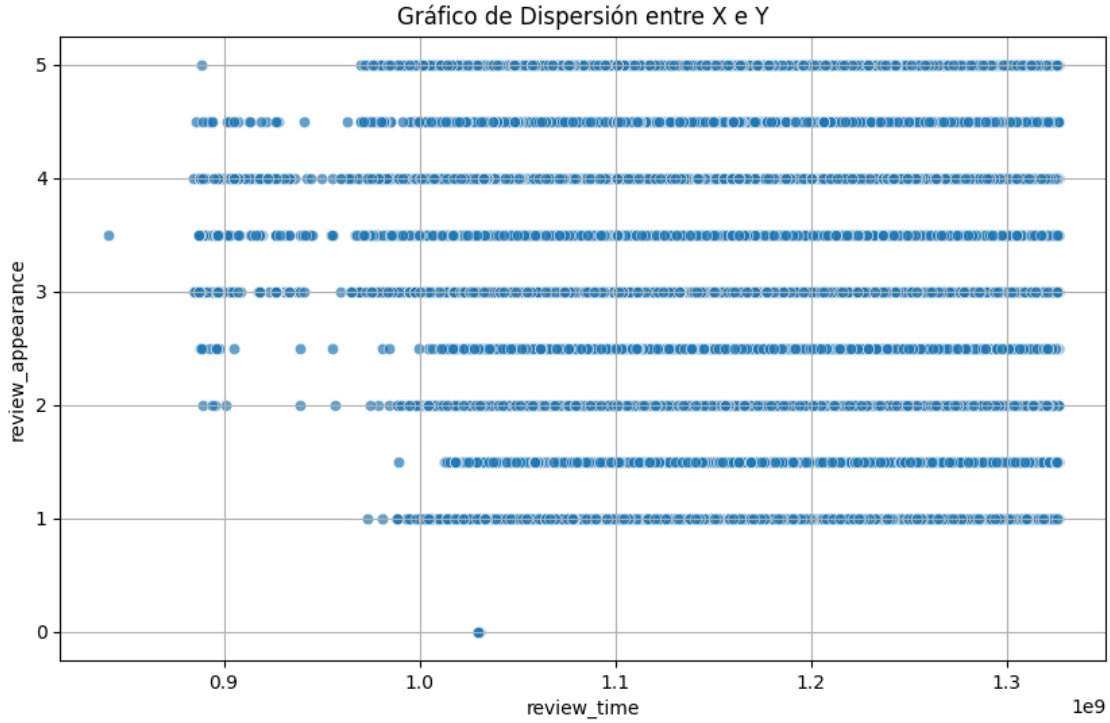
```
Varianza de X: 5859025964738176.0
Desviación estándar de X: 76544274.53923759
Varianza de Y: 0.5357795098053175
Desviación estándar de Y: 0.7319696098919117
```



[31]:
```
# Importar las bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

varianza_x = df['review_time'].var()
desviacion_x = df['review_time'].std()

varianza_y = df['review_appearance'].var()
desviacion_y = df['review_appearance'].std()

print(f'Varianza de X: {varianza_x}')
print(f'Desviación estándar de X: {desviacion_x}')
print(f'Varianza de Y: {varianza_y}')
print(f'Desviación estándar de Y: {desviacion_y}')
```

```
# Crear un gráfico de dispersión
plt.figure(figsize=(10, 6))
sns.scatterplot(x='review_time', y='review_appearance', data=df, alpha=0.7)
plt.title('Gráfico de Dispersión entre X e Y')
plt.xlabel('review_time')
plt.ylabel('review_appearance')
plt.grid(True)
plt.show()
```

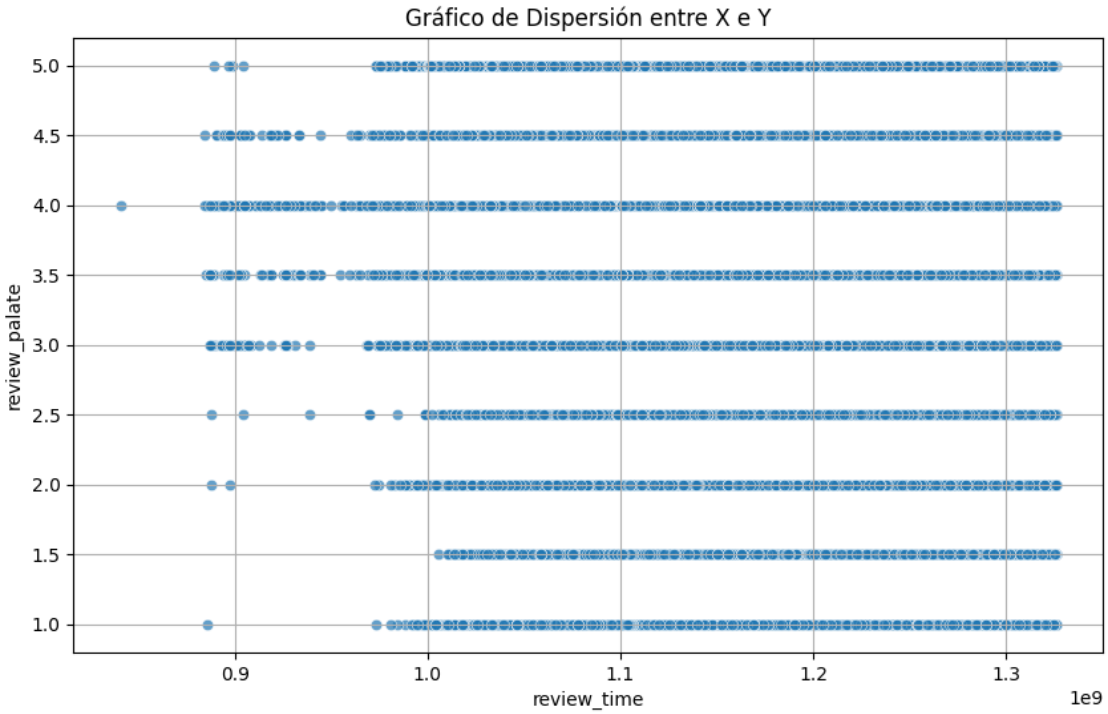
```
Varianza de X: 5859025964738176.0
Desviación estándar de X: 76544274.53923759
Varianza de Y: 0.3795702998810948
Desviación estándar de Y: 0.6160927688920678
```



[32]:
```
# Importar las bibliotecas necesarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

varianza_x = df['review_time'].var()
desviacion_x = df['review_time'].std()
```

```
varianza_y = df['review_palate'].var()
desviacion_y = df['review_palate'].std()

print(f'Varianza de X: {varianza_x}')
print(f'Desviación estándar de X: {desviacion_x}')
print(f'Varianza de Y: {varianza_y}')
print(f'Desviación estándar de Y: {desviacion_y}')

# Crear un gráfico de dispersión
plt.figure(figsize=(10, 6))
sns.scatterplot(x='review_time', y='review_palate', data=df, alpha=0.7)
plt.title('Gráfico de Dispersión entre X e Y')
plt.xlabel('review_time')
plt.ylabel('review_palate')
plt.grid(True)
plt.show()
```

```
Varianza de X: 5859025964738176.0
Desviación estándar de X: 76544274.53923759
Varianza de Y: 0.4654218953246297
Desviación estándar de Y: 0.6822183633739491
```



8 Fase 3: Preparar los datos

9 Eliminacion de datos nulos

En esta etapa del proyecto nos dedicaremos como grupo a preparar los datos de mejor forma para poder tener solo los mejores.

Eliminamos los dato que sean nulos del CSV

La eliminación de los datos nulos nos ayudarán a identificar de mejor manera la información que necesitamos del csv, este se hace por si acaso llegara a haber un dato nulo o faltante que afecte a nuestro analisis

[35]:
```
df.shape
```

[35]:
```
(1518478, 13)
```

[ ]:
```

```

[34]:
```
df = df.dropna()
```

[36]:
```
df.shape
```

[36]:
```
(1518478, 13)
```

[ ]:
```

```

Después de esta eliminación de datos, podemos darnos cuenta de que al menos 68,136 de la tabla eran nulos. Esto nos ayudará a no tener fallas en nuestro análisis detallado de las cervezas.

10 normalización y estandarización de los datos

La estandarización y normalización son necesarios para el preprocesamiento de los datos de las cervezas, este proceso puede mejorar significativamente el rendimiento y la estabilidad de nuestros futuros modelos. Ademas, los datos que pasemos por estos procesos seran tratados de manera justa y adecuada, facilitando un análisis más efectivo y resultados más precisos.

('review_overall','review_aroma', 'review_appearance', 'review_palate', 'review_taste', 'beer_abv')

[37]:
```
#Normalizacion de los datos cerveciles
from sklearn.preprocessing import MinMaxScaler

# Crear el escalador
scaler = MinMaxScaler()

# Normalizar solo las columnas 'A' y 'B'
```

```
df[['review_overall', 'beer_abv', 'review_aroma', 'review_appearance' ,␣
 ↪'review_palate' , 'review_taste' ]] = scaler.
 ↪fit_transform(df[['review_overall', 'beer_abv', 'review_aroma',␣
 ↪'review_appearance' , 'review_palate' , 'review_taste' ]])

# Mostrar el DataFrame con columnas normalizadas
print(df)
```

```
         brewery_id                brewery_name  review_time  review_overall  \
0             10325             Vecchio Birraio   1234817823             0.3
1             10325             Vecchio Birraio   1235915097             0.6
2             10325             Vecchio Birraio   1235916604             0.6
3             10325             Vecchio Birraio   1234725145             0.6
4              1075     Caldera Brewing Company   1293735206             0.8
…               …                           …            …               …
1586609       14359  The Defiant Brewing Company   1162684892             1.0
1586610       14359  The Defiant Brewing Company   1161048566             0.8
1586611       14359  The Defiant Brewing Company   1160702513             0.9
1586612       14359  The Defiant Brewing Company   1160023044             0.8
1586613       14359  The Defiant Brewing Company   1160005319             1.0

         review_aroma  review_appearance review_profilename  \
0               0.250                0.5            stcules
1               0.375                0.6            stcules
2               0.375                0.6            stcules
3               0.500                0.7            stcules
4               0.875                0.8     johnmichaelsen
…                 …                  …                …
1586609         0.750                0.7         maddogruss
1586610         1.000                0.5          yelterdow
1586611         0.625                0.6           TongoRad
1586612         0.875                0.9           dherling
1586613         0.875                0.9               cbl2

                           beer_style  review_palate  review_taste  \
0                           Hefeweizen          0.125         0.125
1                  English Strong Ale          0.500         0.500
2               Foreign / Export Stout          0.500         0.500
3                     German Pilsener          0.375         0.500
4        American Double / Imperial IPA          0.750         0.875
…                                  …              …             …
1586609                   Pumpkin Ale          0.750         0.750
1586610                   Pumpkin Ale          0.250         0.750
1586611                   Pumpkin Ale          0.625         0.750
1586612                   Pumpkin Ale          0.875         0.875
1586613                   Pumpkin Ale          0.875         0.875
```

```
                      beer_name  beer_abv  beer_beerid
0                   Sausa Weizen  0.086497        47986
1                       Red Moon  0.107298        48213
2          Black Horse Black Beer  0.112498        48215
3                     Sausa Pils  0.086497        47969
4                  Cauldron DIPA  0.133299        64883
…                            …       …           …
1586609      The Horseman's Ale  0.089964        33061
1586610      The Horseman's Ale  0.089964        33061
1586611      The Horseman's Ale  0.089964        33061
1586612      The Horseman's Ale  0.089964        33061
1586613      The Horseman's Ale  0.089964        33061

[1518478 rows x 13 columns]
```

```
C:\Users\massr\AppData\Local\Temp\ipykernel_24740\3120235629.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[['review_overall', 'beer_abv', 'review_aroma', 'review_appearance' ,
'review_palate' , 'review_taste' ]] = scaler.fit_transform(df[['review_overall',
'beer_abv', 'review_aroma', 'review_appearance' , 'review_palate' ,
'review_taste' ]])
```

[38]:
```
#Estandarizacion de los datos cerveciles
from sklearn.preprocessing import StandardScaler

# Crear el escalador
scaler = StandardScaler()

# Estandarizar solo las columnas 'A' y 'B'
df[['review_overall', 'beer_abv', 'review_aroma', 'review_appearance' ,␣
 ↪'review_palate' , 'review_taste' ]] = scaler.
 ↪fit_transform(df[['review_overall', 'beer_abv', 'review_aroma',␣
 ↪'review_appearance' , 'review_palate' , 'review_taste' ]])

# Mostrar el DataFrame con columnas estandarizadas
print(df)
```

```
         brewery_id                brewery_name  review_time  review_overall  \
0             10325              Vecchio Birraio   1234817823       -3.239994
1             10325              Vecchio Birraio   1235915097       -1.148720
2             10325              Vecchio Birraio   1235916604       -1.148720
3             10325              Vecchio Birraio   1234725145       -1.148720
4              1075      Caldera Brewing Company   1293735206        0.245463
```

```
…              …                              …           …              …
1586609       14359  The Defiant Brewing Company   1162684892        1.639646
1586610       14359  The Defiant Brewing Company   1161048566        0.245463
1586611       14359  The Defiant Brewing Company   1160702513        0.942555
1586612       14359  The Defiant Brewing Company   1160023044        0.245463
1586613       14359  The Defiant Brewing Company   1160005319        1.639646

         review_aroma  review_appearance review_profilename  \
0           -2.511302          -2.198210            stcules
1           -1.792233          -1.384289            stcules
2           -1.792233          -1.384289            stcules
3           -1.073164          -0.570368            stcules
4            1.084042           0.243553     johnmichaelsen
…               …                 …                …
1586609      0.364974          -0.570368         maddogruss
1586610      1.803111          -2.198210          yelterdow
1586611     -0.354095          -1.384289           TongoRad
1586612      1.084042           1.057473           dherling
1586613      1.084042           1.057473               cbl2

                            beer_style  review_palate  review_taste  \
0                            Hefeweizen      -3.317561     -3.162309
1                    English Strong Ale      -1.109519     -1.103587
2                Foreign / Export Stout      -1.109519     -1.103587
3                     German Pilsener        -1.845533     -1.103587
4        American Double / Imperial IPA       0.362510      0.955134
…                                …             …            …
1586609                     Pumpkin Ale       0.362510      0.268894
1586610                     Pumpkin Ale      -2.581547      0.268894
1586611                     Pumpkin Ale      -0.373505      0.268894
1586612                     Pumpkin Ale       1.098524      0.955134
1586613                     Pumpkin Ale       1.098524      0.955134

                      beer_name  beer_abv  beer_beerid
0                  Sausa Weizen -0.879410        47986
1                      Red Moon -0.362740        48213
2        Black Horse Black Beer -0.233573        48215
3                   Sausa Pils -0.879410        47969
4                 Cauldron DIPA  0.283097        64883
…                           …       …          …
1586609      The Horseman's Ale -0.793298        33061
1586610      The Horseman's Ale -0.793298        33061
1586611      The Horseman's Ale -0.793298        33061
1586612      The Horseman's Ale -0.793298        33061
1586613      The Horseman's Ale -0.793298        33061

[1518478 rows x 13 columns]
```

```
C:\Users\massr\AppData\Local\Temp\ipykernel_24740\3683372954.py:8:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[['review_overall', 'beer_abv', 'review_aroma', 'review_appearance' ,
'review_palate' , 'review_taste' ]] = scaler.fit_transform(df[['review_overall',
'beer_abv', 'review_aroma', 'review_appearance' , 'review_palate' ,
'review_taste' ]])
```

Despues de optimisar de mejor manera los datos que como grupo concideramos principales, comenzaremas a graficar y dimensionar la informacion extraida de estos mismo,

11 Tratamiento de datos Outliers

Para esta sección de la preparación del dataset tomamos los datos para eliminar los que sean Outliers. Utilizamos un tratamiento basado en el método IQR para los valores atípicos.

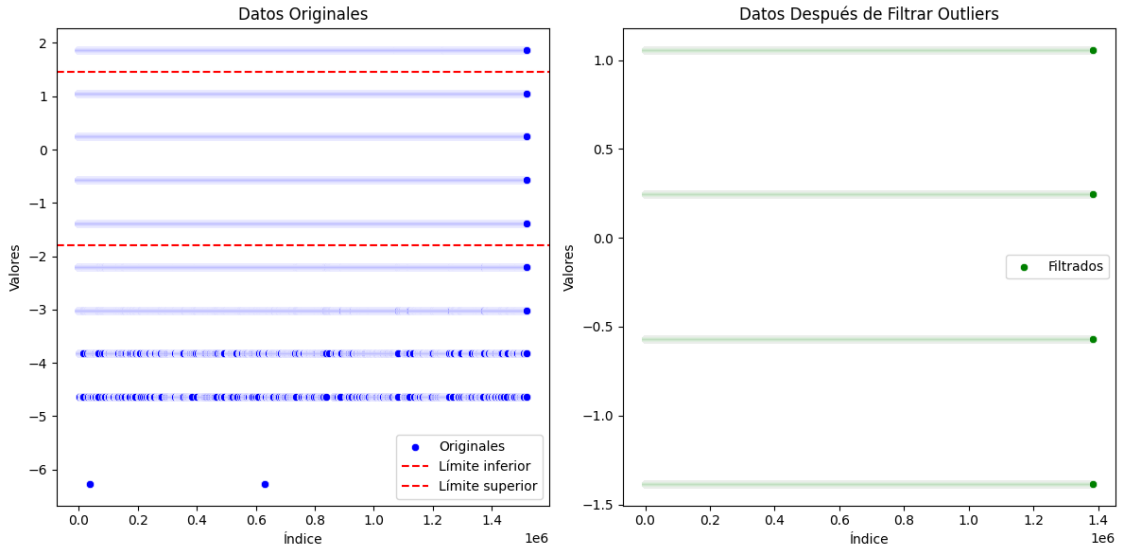
[39]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Muestra las primeras filas del dataframe para verificar su contenido
print("Datos originales:")
print(df.head())


data = df['review_appearance']

# Calcular los cuartiles
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)

# Calcular el rango intercuartílico (IQR)
IQR = Q3 - Q1

# Definir los límites para los valores válidos
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtrar los datos para eliminar outliers
filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

# Crear un nuevo DataFrame con los datos sin los outliers
```

```
filtered_df = df[(df['review_appearance'] >= lower_bound) &␣
 ↪(df['review_appearance'] <= upper_bound)]

# Crear gráficos de dispersión con los datos
plt.figure(figsize=(12, 6))

# Gráfico de dispersión de datos originales
plt.subplot(1, 2, 1)
sns.scatterplot(x=np.arange(len(data)), y=data, color='blue',␣
 ↪label='Originales')
plt.axhline(y=lower_bound, color='r', linestyle='--', label='Límite inferior')
plt.axhline(y=upper_bound, color='r', linestyle='--', label='Límite superior')
plt.title('Datos Originales')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

# Gráfico de dispersión de datos filtrados
plt.subplot(1, 2, 2)
sns.scatterplot(x=np.arange(len(filtered_data)), y=filtered_data,␣
 ↪color='green', label='Filtrados')
plt.title('Datos Después de Filtrar Outliers')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Datos originales:
   brewery_id             brewery_name  review_time  review_overall  \
0       10325          Vecchio Birraio   1234817823       -3.239994
1       10325          Vecchio Birraio   1235915097       -1.148720
2       10325          Vecchio Birraio   1235916604       -1.148720
3       10325          Vecchio Birraio   1234725145       -1.148720
4        1075  Caldera Brewing Company   1293735206        0.245463

   review_aroma  review_appearance review_profilename  \
0     -2.511302          -2.198210            stcules
1     -1.792233          -1.384289            stcules
2     -1.792233          -1.384289            stcules
3     -1.073164          -0.570368            stcules
4      1.084042           0.243553     johnmichaelsen

                      beer_style  review_palate  review_taste  \
0                      Hefeweizen      -3.317561     -3.162309
1             English Strong Ale      -1.109519     -1.103587
```

```
2          Foreign / Export Stout      -1.109519     -1.103587
3                  German Pilsener      -1.845533     -1.103587
4  American Double / Imperial IPA       0.362510      0.955134

                beer_name  beer_abv  beer_beerid
0            Sausa Weizen -0.879410        47986
1                Red Moon -0.362740        48213
2  Black Horse Black Beer -0.233573        48215
3              Sausa Pils -0.879410        47969
4           Cauldron DIPA  0.283097        64883
```

```
C:\Users\massr\AppData\Local\Temp\ipykernel_24740\2624559006.py:51: UserWarning:
Creating legend with loc="best" can be slow with large amounts of data.
  plt.tight_layout()
c:\Users\massr\Desktop\reviewsCerveceriaKedro\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with
loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



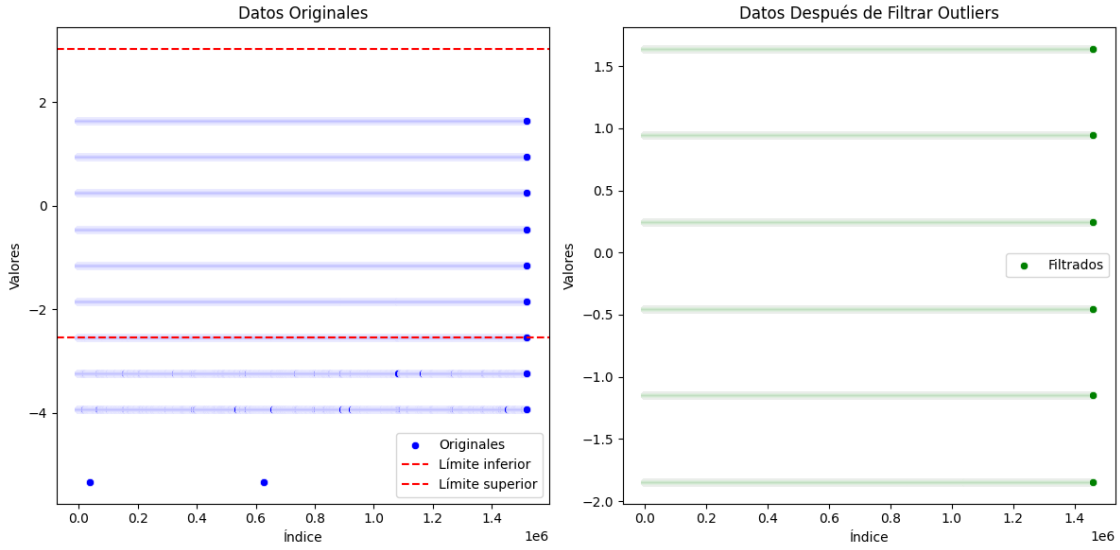
[40]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Muestra las primeras filas del dataframe para verificar su contenido
print("Datos originales:")
print(df.head())
```

```
data = df['review_overall']

# Calcular los cuartiles
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)

# Calcular el rango intercuartílico (IQR)
IQR = Q3 - Q1

# Definir los límites para los valores válidos
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtrar los datos para eliminar outliers
filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

# Crear un nuevo DataFrame con los datos sin outliers
filtered_df = df[(df['review_overall'] >= lower_bound) & (df['review_overall']␣
 ↪<= upper_bound)]

# Crear gráficos de dispersión
plt.figure(figsize=(12, 6))

# Gráfico de dispersión de datos originales
plt.subplot(1, 2, 1)
sns.scatterplot(x=np.arange(len(data)), y=data, color='blue',␣
 ↪label='Originales')
plt.axhline(y=lower_bound, color='r', linestyle='--', label='Límite inferior')
plt.axhline(y=upper_bound, color='r', linestyle='--', label='Límite superior')
plt.title('Datos Originales')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

# Gráfico de dispersión de datos filtrados
plt.subplot(1, 2, 2)
sns.scatterplot(x=np.arange(len(filtered_data)), y=filtered_data,␣
 ↪color='green', label='Filtrados')
plt.title('Datos Después de Filtrar Outliers')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

plt.tight_layout()
plt.show()
```

Datos originales:

```
   brewery_id             brewery_name  review_time  review_overall  \
0       10325          Vecchio Birraio   1234817823       -3.239994
1       10325          Vecchio Birraio   1235915097       -1.148720
2       10325          Vecchio Birraio   1235916604       -1.148720
3       10325          Vecchio Birraio   1234725145       -1.148720
4        1075  Caldera Brewing Company   1293735206        0.245463

   review_aroma  review_appearance review_profilename  \
0     -2.511302          -2.198210            stcules
1     -1.792233          -1.384289            stcules
2     -1.792233          -1.384289            stcules
3     -1.073164          -0.570368            stcules
4      1.084042           0.243553     johnmichaelsen

                      beer_style  review_palate  review_taste  \
0                      Hefeweizen      -3.317561     -3.162309
1             English Strong Ale      -1.109519     -1.103587
2         Foreign / Export Stout      -1.109519     -1.103587
3                German Pilsener      -1.845533     -1.103587
4  American Double / Imperial IPA       0.362510      0.955134

               beer_name  beer_abv  beer_beerid
0           Sausa Weizen -0.879410        47986
1               Red Moon -0.362740        48213
2  Black Horse Black Beer -0.233573        48215
3             Sausa Pils -0.879410        47969
4          Cauldron DIPA  0.283097        64883
```

```
C:\Users\massr\AppData\Local\Temp\ipykernel_24740\1443692622.py:51: UserWarning:
Creating legend with loc="best" can be slow with large amounts of data.
  plt.tight_layout()
c:\Users\massr\Desktop\reviewsCerveceriaKedro\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with
loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



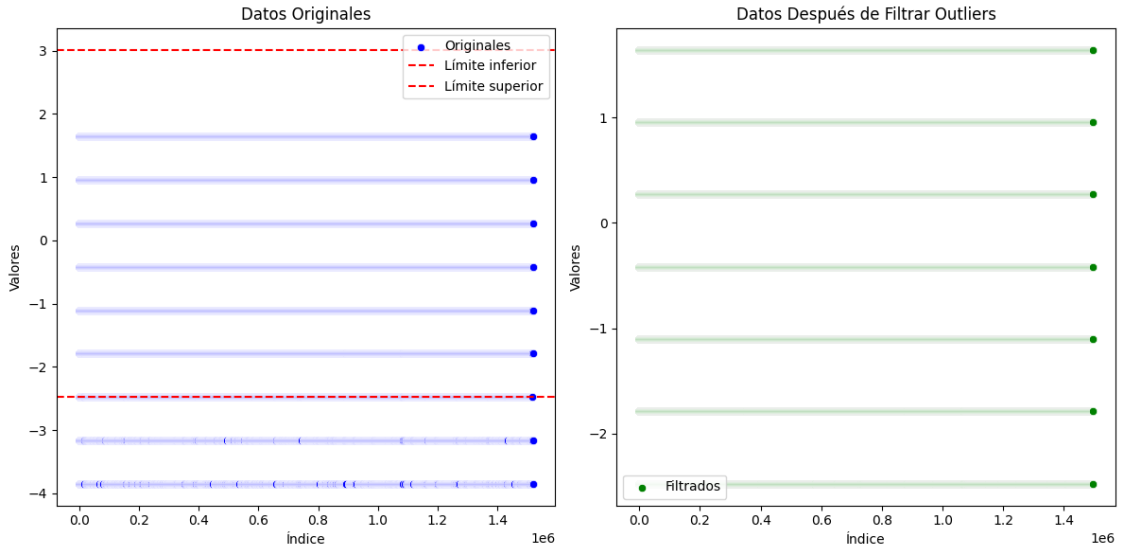
[41]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Muestra las primeras filas del dataframe para verificar su contenido
print("Datos originales:")
print(df.head())


data = df['review_taste']

# Calcular los cuartiles
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)

# Calcular el rango intercuartílico (IQR)
IQR = Q3 - Q1

# Definir los límites para los valores válidos
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtrar los datos para eliminar outliers
filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

# Crear un nuevo DataFrame con los datos sin outliers
```

```
filtered_df = df[(df['review_taste'] >= lower_bound) & (df['review_taste'] <=␣
 ↪upper_bound)]

# Crear gráficos de dispersión
plt.figure(figsize=(12, 6))

# Gráfico de dispersión de datos originales
plt.subplot(1, 2, 1)
sns.scatterplot(x=np.arange(len(data)), y=data, color='blue',␣
 ↪label='Originales')
plt.axhline(y=lower_bound, color='r', linestyle='--', label='Límite inferior')
plt.axhline(y=upper_bound, color='r', linestyle='--', label='Límite superior')
plt.title('Datos Originales')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

# Gráfico de dispersión de datos filtrados
plt.subplot(1, 2, 2)
sns.scatterplot(x=np.arange(len(filtered_data)), y=filtered_data,␣
 ↪color='green', label='Filtrados')
plt.title('Datos Después de Filtrar Outliers')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

plt.tight_layout()
plt.show()
```

```
Datos originales:
   brewery_id             brewery_name  review_time  review_overall  \
0       10325          Vecchio Birraio   1234817823       -3.239994
1       10325          Vecchio Birraio   1235915097       -1.148720
2       10325          Vecchio Birraio   1235916604       -1.148720
3       10325          Vecchio Birraio   1234725145       -1.148720
4        1075  Caldera Brewing Company   1293735206        0.245463

   review_aroma  review_appearance review_profilename  \
0     -2.511302          -2.198210            stcules
1     -1.792233          -1.384289            stcules
2     -1.792233          -1.384289            stcules
3     -1.073164          -0.570368            stcules
4      1.084042           0.243553     johnmichaelsen

                      beer_style  review_palate  review_taste  \
0                      Hefeweizen      -3.317561     -3.162309
1             English Strong Ale      -1.109519     -1.103587
```

```
2          Foreign / Export Stout      -1.109519     -1.103587
3                  German Pilsener      -1.845533     -1.103587
4  American Double / Imperial IPA       0.362510      0.955134

                beer_name  beer_abv  beer_beerid
0            Sausa Weizen -0.879410        47986
1                Red Moon -0.362740        48213
2  Black Horse Black Beer -0.233573        48215
3              Sausa Pils -0.879410        47969
4           Cauldron DIPA  0.283097        64883
```

```
C:\Users\massr\AppData\Local\Temp\ipykernel_24740\3500587280.py:51: UserWarning:
Creating legend with loc="best" can be slow with large amounts of data.
  plt.tight_layout()
c:\Users\massr\Desktop\reviewsCerveceriaKedro\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with
loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



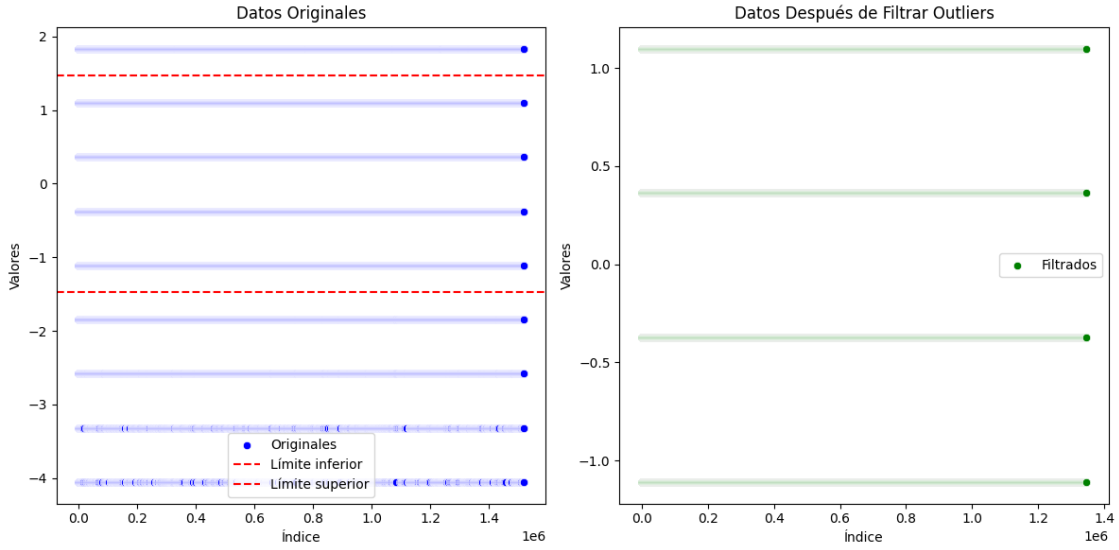
[42]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Muestra las primeras filas del dataframe para verificar su contenido
print("Datos originales:")
print(df.head())
```

```
data = df['review_palate']

# Calcular los cuartiles
Q1 = data.quantile(0.25)
Q3 = data.quantile(0.75)

# Calcular el rango intercuartílico (IQR)
IQR = Q3 - Q1

# Definir los límites para los valores válidos
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtrar los datos para eliminar outliers
filtered_data = data[(data >= lower_bound) & (data <= upper_bound)]

# Crear un nuevo DataFrame con los datos sin outliers
filtered_df = df[(df['review_palate'] >= lower_bound) & (df['review_palate'] <=␣
 ↪upper_bound)]

# Crear gráficos de dispersión
plt.figure(figsize=(12, 6))

# Gráfico de dispersión de datos originales
plt.subplot(1, 2, 1)
sns.scatterplot(x=np.arange(len(data)), y=data, color='blue',␣
 ↪label='Originales')
plt.axhline(y=lower_bound, color='r', linestyle='--', label='Límite inferior')
plt.axhline(y=upper_bound, color='r', linestyle='--', label='Límite superior')
plt.title('Datos Originales')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

# Gráfico de dispersión de datos filtrados
plt.subplot(1, 2, 2)
sns.scatterplot(x=np.arange(len(filtered_data)), y=filtered_data,␣
 ↪color='green', label='Filtrados')
plt.title('Datos Después de Filtrar Outliers')
plt.xlabel('Índice')
plt.ylabel('Valores')
plt.legend()

plt.tight_layout()
plt.show()
```

Datos originales:

```
   brewery_id             brewery_name  review_time  review_overall  \
0       10325          Vecchio Birraio   1234817823       -3.239994
1       10325          Vecchio Birraio   1235915097       -1.148720
2       10325          Vecchio Birraio   1235916604       -1.148720
3       10325          Vecchio Birraio   1234725145       -1.148720
4        1075  Caldera Brewing Company   1293735206        0.245463

   review_aroma  review_appearance review_profilename  \
0     -2.511302          -2.198210            stcules
1     -1.792233          -1.384289            stcules
2     -1.792233          -1.384289            stcules
3     -1.073164          -0.570368            stcules
4      1.084042           0.243553     johnmichaelsen

                     beer_style  review_palate  review_taste  \
0                     Hefeweizen      -3.317561     -3.162309
1             English Strong Ale      -1.109519     -1.103587
2         Foreign / Export Stout      -1.109519     -1.103587
3                German Pilsener      -1.845533     -1.103587
4  American Double / Imperial IPA       0.362510      0.955134

               beer_name  beer_abv  beer_beerid
0           Sausa Weizen -0.879410        47986
1               Red Moon -0.362740        48213
2  Black Horse Black Beer -0.233573        48215
3             Sausa Pils -0.879410        47969
4          Cauldron DIPA  0.283097        64883
```

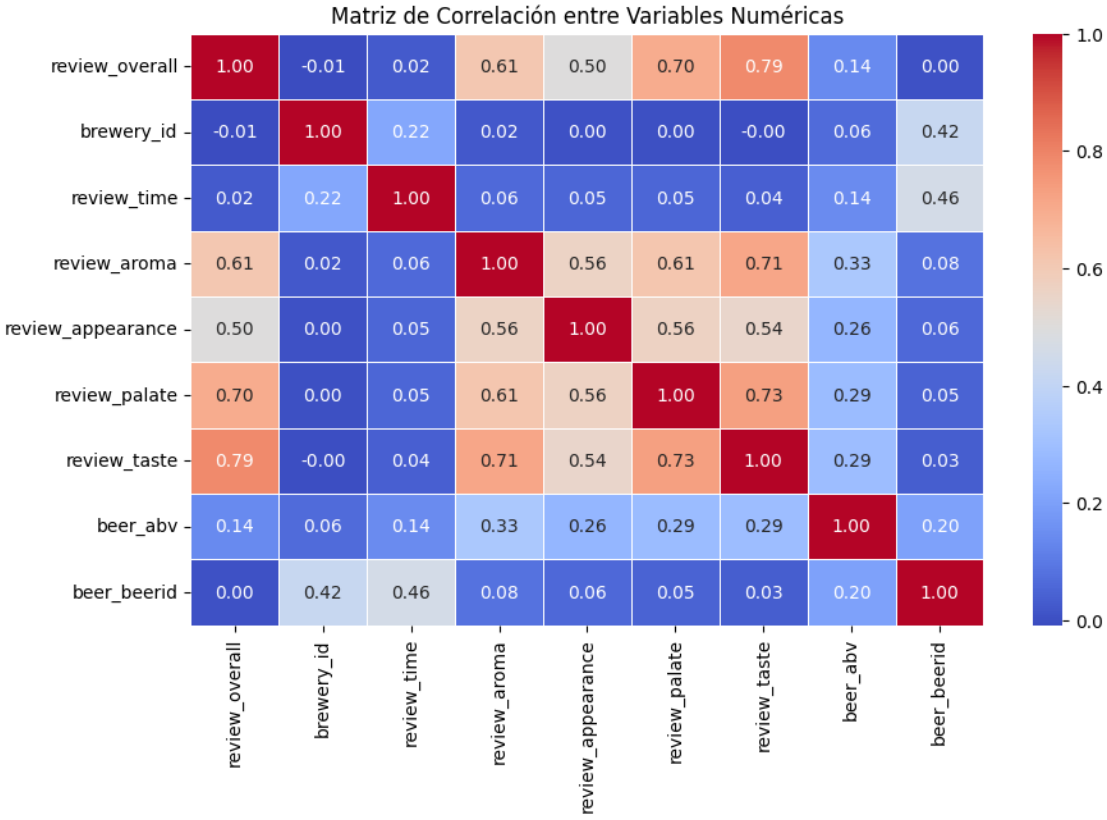
```
C:\Users\massr\AppData\Local\Temp\ipykernel_24740\2056312737.py:51: UserWarning:
Creating legend with loc="best" can be slow with large amounts of data.
  plt.tight_layout()
c:\Users\massr\Desktop\reviewsCerveceriaKedro\venv\Lib\site-
packages\IPython\core\pylabtools.py:170: UserWarning: Creating legend with
loc="best" can be slow with large amounts of data.
  fig.canvas.print_figure(bytes_io, **kw)
```



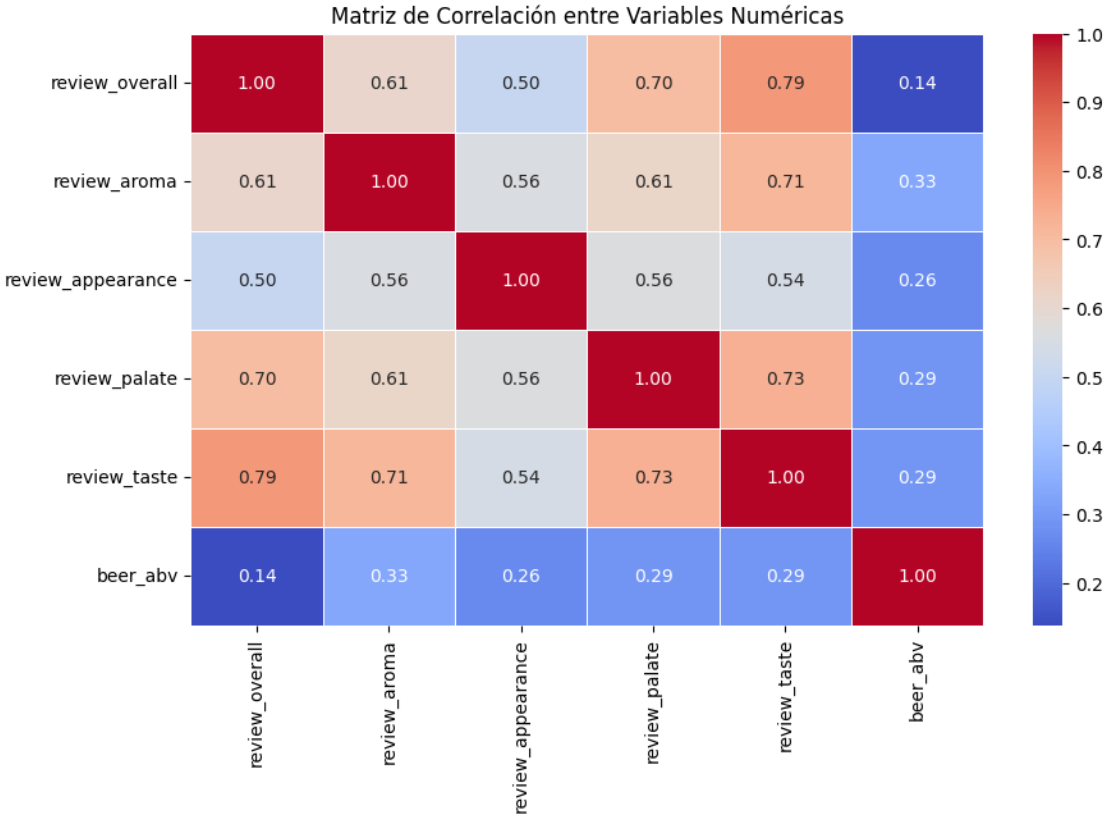
12 Mapas de calor

Graficamos mediante mapas de calor para ver que datos es tan correlacionados y cuales no

[43]:
```
# Importar las bibliotecas necesarias
import seaborn as sns
import matplotlib.pyplot as plt

# Seleccionar las columnas numéricas para la correlación
numeric_columns = ['review_overall','brewery_id', 'review_time','review_aroma',␣
 ↪'review_appearance','review_palate', 'review_taste',␣
 ↪'beer_abv','beer_beerid']

# Calcular la matriz de correlación
correlation_matrix = df[numeric_columns].corr()

# Crear la matriz de calor
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5,␣
 ↪fmt=".2f")
plt.title('Matriz de Correlación entre Variables Numéricas')
plt.show()
```



[44]:
```
# Importar las bibliotecas necesarias
import seaborn as sns
import matplotlib.pyplot as plt

# Seleccionar las columnas numéricas para la correlación
numeric_columns = [ 'review_overall','review_aroma', 'review_appearance',
                    'review_palate', 'review_taste', 'beer_abv']

# Calcular la matriz de correlación
correlation_matrix = df[numeric_columns].corr()

# Crear la matriz de calor
plt.figure(figsize=(10, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5,␣
 ↪fmt=".2f")
plt.title('Matriz de Correlación entre Variables Numéricas')
plt.show()
```



Eliminamos review_profilename ya que para nosotros como equipo no es necesario para el futuro del proyecto.

[45]:
```
df = df.dropna()
df.columns
df2=df.drop(["review_profilename"], axis=1, inplace=True)
```

[46]:
```
df.columns
```

[46]:
```
Index(['brewery_id', 'brewery_name', 'review_time', 'review_overall',
       'review_aroma', 'review_appearance', 'beer_style', 'review_palate',
       'review_taste', 'beer_name', 'beer_abv', 'beer_beerid'],
      dtype='object')
```

#Filtración de datos En esta parte tomamos como dato principal el de review_overall para filtrar los datos de las cerveza y así responder a la pregunta de cuál de las cervezas es la mejor.

En este filtrado de datos solo tomamos en cuenta los datos que sean mayor o igual a 4.5 dejándonos con un total de 403.000 datos lo cual es bastante.

13 Filtracion de cervezas por graduacion alcoholica

[47]:
```
#Filtrando datos con el beer_abv para cervezas suaves
Cervezas_Suave = df[df['beer_abv'] >=3.5]
```

[48]:
```
Cervezas_Suave
```

[48]:

```
         brewery_id                                brewery_name  review_time  \
9634            395  Bluegrass Brewing Co. - East St. Matthew's   1227631991
9635            395  Bluegrass Brewing Co. - East St. Matthew's   1226032900
12911          6513                                Schorschbräu   1283880857
12912          6513                                Schorschbräu   1262984846
12913          6513                                Schorschbräu   1256279422
…               …                                         …            …
1453199       14060                              Shoes Brewery   1159327560
1483002       12015                       Arctic Craft Brewery   1201895641
1498776        4950               Devil's Canyon Brewing Company   1276400730
1498777        4950               Devil's Canyon Brewing Company   1266386891
1498778        4950               Devil's Canyon Brewing Company   1266256647

         review_overall  review_aroma  review_appearance  \
9634           0.245463      1.084042           0.243553
9635          -0.451628     -0.354095          -0.570368
12911          0.245463     -0.354095          -1.384289
12912         -0.451628     -1.073164          -1.384289
12913          0.245463     -0.354095          -1.384289
…                 …             …                  …
1453199       -2.542902     -1.073164          -0.570368
1483002        0.245463      1.084042           0.243553
1498776        0.245463     -0.354095           0.243553
1498777       -2.542902     -1.073164          -1.384289
1498778       -1.148720     -0.354095          -1.384289

                             beer_style  review_palate  review_taste  \
9634     American Double / Imperial Stout       1.098524      0.268894
9635     American Double / Imperial Stout       0.362510     -0.417347
12911                        Doppelbock      -0.373505      0.268894
12912                        Doppelbock      -1.109519     -0.417347
12913                        Doppelbock      -1.109519      0.268894
…                                   …              …             …
1453199              English Barleywine      -1.845533     -1.789828
1483002                         Eisbock       0.362510      0.955134
1498776             American Barleywine      -0.373505     -0.417347
1498777             American Barleywine      -1.845533     -1.103587
1498778             American Barleywine      -1.109519     -0.417347

                                          beer_name  beer_abv  \
```

```
9634     Jeffersons Reserve Big Fella Bourbon Barrel Stout  3.856729
9635     Jeffersons Reserve Big Fella Bourbon Barrel Stout  3.856729
12911                                          Schorschbock  3.856729
12912                                          Schorschbock  3.856729
12913                                          Schorschbock  3.856729
…                                                       …       …
1453199                                  Farriers' Beer  3.512282
1483002                               Warning Sign Eisbock  4.717845
1498776                                  Barrel Of Monkeys  4.244231
1498777                                  Barrel Of Monkeys  4.244231
1498778                                  Barrel Of Monkeys  4.244231

         beer_beerid
9634           45774
9635           45774
12911          47421
12912          47421
12913          47421
…                …
1453199        32949
1483002        40821
1498776        56134
1498777        56134
1498778        56134

[9587 rows x 12 columns]
```

[49]:
```
#Filtrando datos con el beer_abv para cervezas estandar
Cervezas_Estandar = df[df['beer_abv'] <=5.5]
```

[50]:
```
Cervezas_Estandar
```

[50]:
```
         brewery_id                brewery_name  review_time  review_overall  \
0             10325             Vecchio Birraio   1234817823       -3.239994
1             10325             Vecchio Birraio   1235915097       -1.148720
2             10325             Vecchio Birraio   1235916604       -1.148720
3             10325             Vecchio Birraio   1234725145       -1.148720
4              1075     Caldera Brewing Company   1293735206        0.245463
…               …                           …            …               …
1586609       14359  The Defiant Brewing Company   1162684892        1.639646
1586610       14359  The Defiant Brewing Company   1161048566        0.245463
1586611       14359  The Defiant Brewing Company   1160702513        0.942555
1586612       14359  The Defiant Brewing Company   1160023044        0.245463
1586613       14359  The Defiant Brewing Company   1160005319        1.639646

         review_aroma  review_appearance                    beer_style  \
0          -2.511302          -2.198210                    Hefeweizen
```

```
1           -1.792233          -1.384289              English Strong Ale
2           -1.792233          -1.384289          Foreign / Export Stout
3           -1.073164          -0.570368                German Pilsener
4            1.084042           0.243553  American Double / Imperial IPA
…                …                  …                             …
1586609      0.364974          -0.570368                     Pumpkin Ale
1586610      1.803111          -2.198210                     Pumpkin Ale
1586611     -0.354095          -1.384289                     Pumpkin Ale
1586612      1.084042           1.057473                     Pumpkin Ale
1586613      1.084042           1.057473                     Pumpkin Ale

         review_palate  review_taste               beer_name  beer_abv  \
0            -3.317561     -3.162309            Sausa Weizen -0.879410
1            -1.109519     -1.103587                Red Moon -0.362740
2            -1.109519     -1.103587  Black Horse Black Beer -0.233573
3            -1.845533     -1.103587              Sausa Pils -0.879410
4             0.362510      0.955134           Cauldron DIPA  0.283097
…                …             …                       …       …
1586609       0.362510      0.268894      The Horseman's Ale -0.793298
1586610      -2.581547      0.268894      The Horseman's Ale -0.793298
1586611      -0.373505      0.268894      The Horseman's Ale -0.793298
1586612       1.098524      0.955134      The Horseman's Ale -0.793298
1586613       1.098524      0.955134      The Horseman's Ale -0.793298

         beer_beerid
0              47986
1              48213
2              48215
3              47969
4              64883
…                …
1586609        33061
1586610        33061
1586611        33061
1586612        33061
1586613        33061

[1517722 rows x 12 columns]
```

[51]:
```
#Filtrando datos con el beer_abv para cervezas fuertes
Cervezas_Fuertes = df[df['beer_abv'] >5.5]
```

[52]:
```
Cervezas_Fuertes
```

[52]:
```
         brewery_id           brewery_name  review_time  review_overall  \
12918          6513           Schorschbräu   1248785936       -0.451628
12919          6513           Schorschbräu   1316780901        0.245463
```

```
12934          6513            Schorschbräu   1264684153        0.245463
12939          6513            Schorschbräu   1309974178        0.245463
12940          6513            Schorschbräu   1274469798       -0.451628
…               …                       …           …              …
1386403        1924  DuClaw Brewing Company   1265957224       -1.845811
1386404        1924  DuClaw Brewing Company   1243749469       -1.148720
1386405        1924  DuClaw Brewing Company   1238893861       -0.451628
1386406        1924  DuClaw Brewing Company   1238768944        0.942555
1386407        1924  DuClaw Brewing Company   1238680017        0.245463

         review_aroma  review_appearance           beer_style  review_palate  \
12918        0.364974           0.243553              Eisbock       0.362510
12919        0.364974           0.243553              Eisbock       0.362510
12934        1.084042           1.057473              Eisbock       0.362510
12939        0.364974          -0.570368              Eisbock       0.362510
12940        0.364974           0.243553              Eisbock       0.362510
…                 …                  …                    …             …
1386403      0.364974           0.243553  American Strong Ale       0.362510
1386404     -0.354095           0.243553  American Strong Ale       0.362510
1386405     -0.354095           0.243553  American Strong Ale       0.362510
1386406      1.084042           1.057473  American Strong Ale       1.098524
1386407     -0.354095           0.243553  American Strong Ale       1.098524

         review_taste                      beer_name   beer_abv  beer_beerid
12918        0.955134  Schorschbräu Schorschbock 31%  10.254821        51466
12919       -0.417347  Schorschbräu Schorschbock 57%  21.810999        73368
12934        0.955134  Schorschbräu Schorschbock 40%  13.949009        55712
12939        0.268894  Schorschbräu Schorschbock 43%  15.481796        57856
12940        0.955134  Schorschbräu Schorschbock 43%  15.481796        57856
…                 …                             …          …            …
1386403     -0.417347                      Colossus   6.405632        48881
1386404      0.268894                      Colossus   6.405632        48881
1386405      0.268894                      Colossus   6.405632        48881
1386406      0.268894                      Colossus   6.405632        48881
1386407      0.268894                      Colossus   6.405632        48881

[756 rows x 12 columns]
```

14 Filtracion de las cervezas por review general

[53]:
```
# Filtrar las filas donde 'review_overall' es mayor o igual a 4.5
filtered_df = df[df['review_overall'] >= 4.5]
```

[54]:
```
filtered_df
```

[54]:
```
Empty DataFrame
Columns: [brewery_id, brewery_name, review_time, review_overall, review_aroma,
review_appearance, beer_style, review_palate, review_taste, beer_name, beer_abv,
beer_beerid]
Index: []
```

Para el segundo filtrado de los datos tomamos en cuenta sólo los datos que sean mayor o igual a 5.0, ya que queríamos ver cuales eran las mejores cervezas que estaban clasificadas en los datos.

[55]:
```
# Filtrar las filas donde 'review_overall' es mayor o igual a 5
filtered_df_2 = df[df['review_overall'] >=5.0]
```

[56]:
```
filtered_df_2
```

[56]:
```
Empty DataFrame
Columns: [brewery_id, brewery_name, review_time, review_overall, review_aroma,
review_appearance, beer_style, review_palate, review_taste, beer_name, beer_abv,
beer_beerid]
Index: []
```

#CSV de la segunda filtración de datos. En esta parte colocamos una posible descarga de la filtración de datos que hemos hecho anteriormente, debido a que los mismos pueden ser ocupados en un futuro.

[57]:
```
# Guardar el DataFrame filtrado en un nuevo archivo CSV
#filtered_df_2.to_csv('filtered_review_overall_2.csv', index=False)
```

[58]:
```
# Descargar el archivo CSV filtrado
#files.download('filtered_review_overall_2.csv')
```

15 Resumen de la primera parte del proyecto

En este dataset tomamos los datos de la cervezeria Kross para poder hacer un analisis de mercado. Mientras observamos los datos encontramos que hay varios factores que pueden afectar en la clasificacion de una cerveza entre las variables que más influyen son el sabor, aroma, paladar y apariencia. Estas 4 variables son las que mas impactan las calificaciones.

Pudimos identificar cuales de estas variables se correlacionan en mayor cantidad gracias a la Matriz de correlacion. Debido a esto, identificamos las variales que más afectan el puntaje de una cerveza.

En cuanto a la metodologia que utilizada es CRISP-DM, debido a las ventajas que esta trae como :

-Estructura clara: Define un proceso ordenado para manejar grandes volúmenes de datos.

-Completitud: Aborda todo el ciclo de vida del proyecto, desde el análisis del negocio hasta la implementación.

Debido a los factores mencionado anteriormente podemos identificar que cervezas se venderan mejor en el mercado chileno sin sufrir perdidas por posibles productos defectuosos.

16 Fase 4: modelar los datos

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: