

Big Data in Media Technology

Lab 2: Use Generative AI To Support Advanced Data Analytics



Group 3

Name: Yangle Ma

Group Members: JingWen Qin YuTing Huang

Task

1. Exploratory Data Analysis (EDA)

1.1 Correlation between numerical features in the dataset

Prompt:

1. I have a wine dataset, the features are Name Variety Origin Attr_1 Attr_2 Alcohol_vol Alcohol_percentage Winemaker_notes Review Avg_rating N_ratings Price_Out-of-stock Price 13 features all. I wanna visualize Correlation between numerical features in the dataset
2. could not convert string to float: '1,199.99'
3. AttributeError: Can only use .str accessor with string values!
4. ValueError: could not convert string to float: "

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

# Ensure the 'Price' and 'Price (Out-of-stock)' columns are strings
df['Price'] = df['Price'].astype(str).str.replace(',', '').astype(float)
df['Price_Out-of-stock'] = df['Price_Out-of-stock'].replace('',
np.nan).astype(str).str.replace(',', '').astype(float)

# Select numerical features
numerical_features = df[['Alcohol_vol', 'Alcohol_percentage', 'Avg_rating', 'N_ratings', 'Price',
'Price_Out-of-stock']]

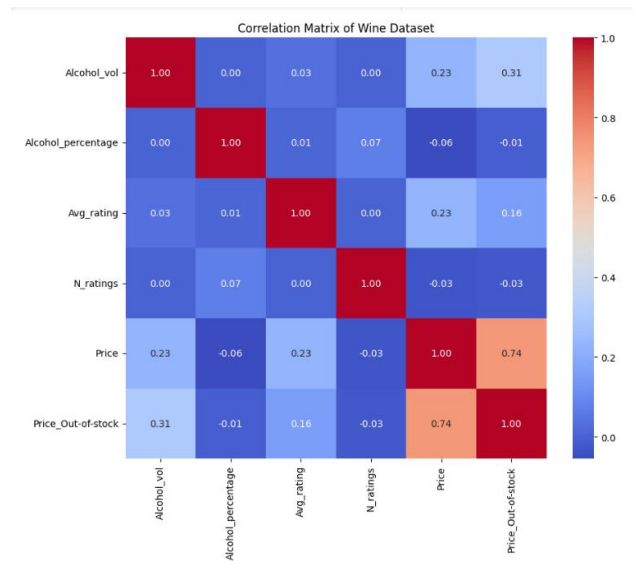
# Calculate the correlation matrix
correlation_matrix = numerical_features.corr()

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Create a heatmap
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm', square=True)

# Add title
plt.title('Correlation Matrix of Wine Dataset')

# Show the plot
plt.show()
```



1.2 Top ten countries with the most wines in the dataset

Prompt:

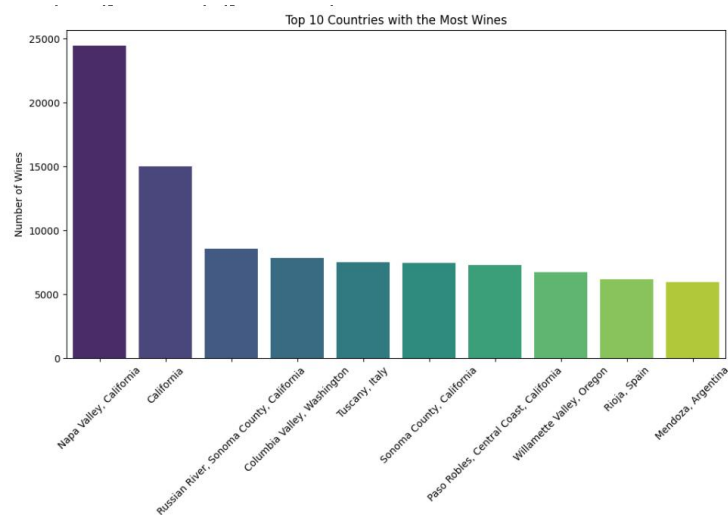
5. Visualize Top ten countries with the most wines in the dataset

```
# Count wines by country
country_counts = df['Origin'].value_counts()

# Select the top 10 countries
top_countries = country_counts.head(10)

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=top_countries.index, y=top_countries.values, palette='viridis')

# Add labels and title
plt.xlabel('Country')
plt.ylabel('Number of Wines')
plt.title('Top 10 Countries with the Most Wines')
plt.xticks(rotation=45)
plt.show()
```



1.3 Top ten grape varieties with the most entries in the dataset

Prompt:

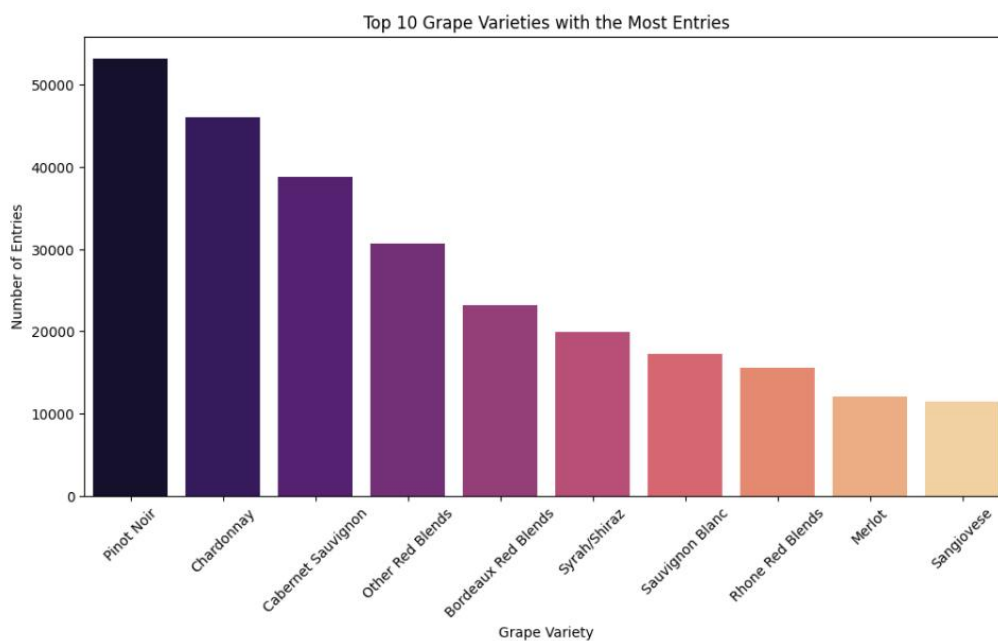
6. Top ten grape varieties with the most entries in the dataset

```
# Count entries by grape variety
variety_counts = df['Variety'].value_counts()

# Select the top 10 grape varieties
top_varieties = variety_counts.head(10)

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=top_varieties.index, y=top_varieties.values, palette='magma')

# Add labels and title
plt.xlabel('Grape Variety')
plt.ylabel('Number of Entries')
plt.title('Top 10 Grape Varieties with the Most Entries')
plt.xticks(rotation=45)
plt.show()
```



1.4 Distribution of unique values among user review scores

Prompt:

7. Visualize Distribution of unique values among user review scores

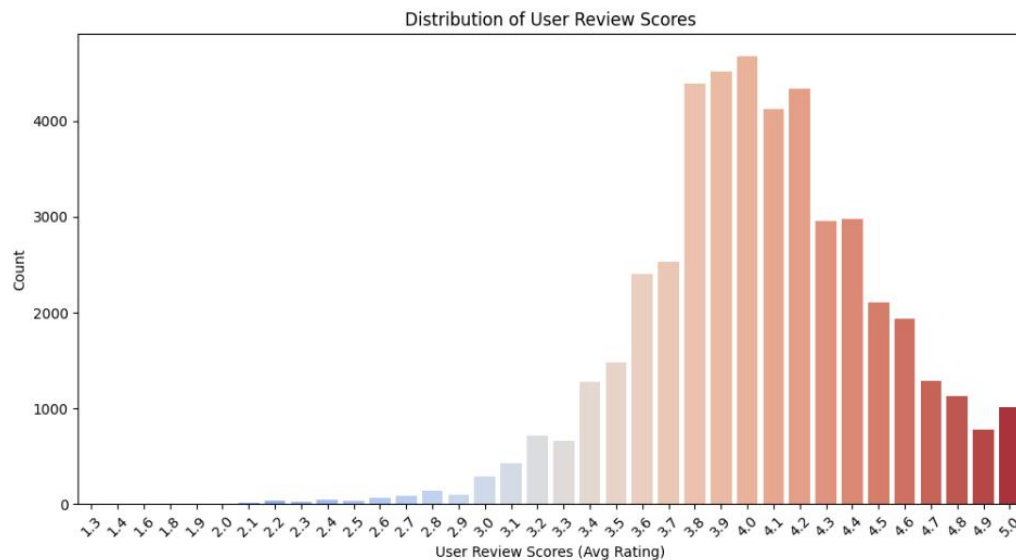
```
plt.figure(figsize=(12, 6))

# Using a count plot for unique review scores
```

```

sns.countplot(x='Avg_rating', data=df, palette='coolwarm')
# Add labels and title
plt.xlabel('User Review Scores (Avg Rating)')
plt.ylabel('Count')
plt.title('Distribution of User Review Scores')
plt.xticks(rotation=45)
plt.show()

```



1.5 Average user rating from the top ten countries

Prompt:

8. Visualize Average user rating from the top ten countries

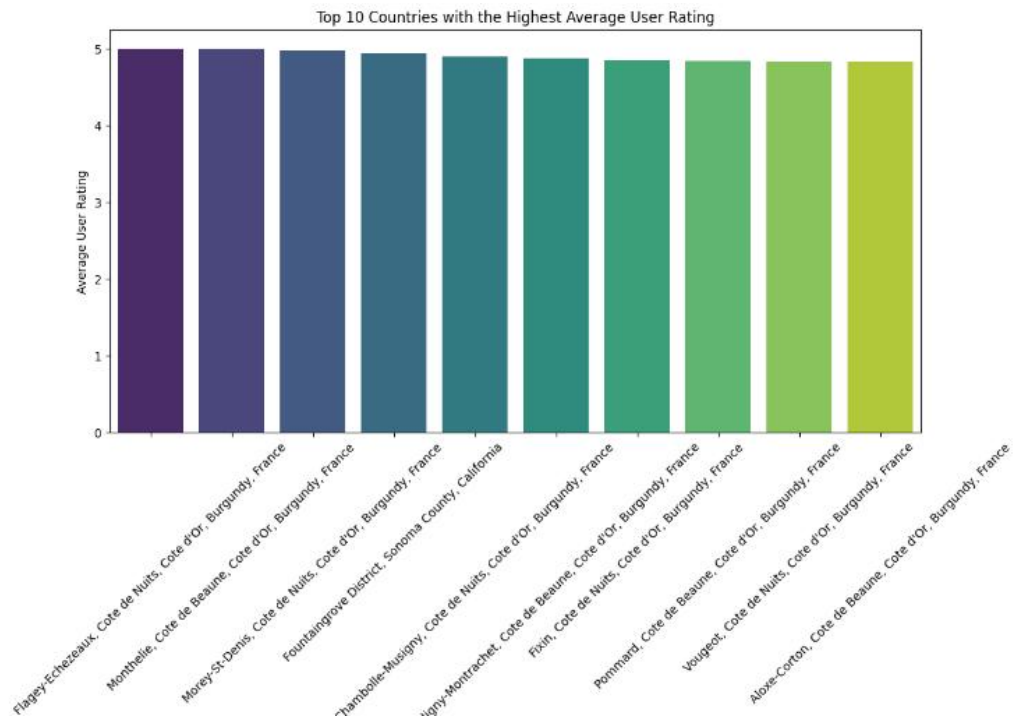
```

# Calculate average rating by country
average_rating = df.groupby('Origin')['Avg_rating'].mean()

# Sort and select the top 10 countries by average rating
top_countries = average_rating.sort_values(ascending=False).head(10)

# Create a bar plot
plt.figure(figsize=(12, 6))
sns.barplot(x=top_countries.index, y=top_countries.values, palette='viridis')
# Add labels and title
plt.xlabel('Country')
plt.ylabel('Average User Rating')
plt.title('Top 10 Countries with the Highest Average User Rating')
plt.xticks(rotation=45)
plt.show()

```



It can be seen that we couldn't get expected results without cleaning the raw data.

2. Data Cleaning and Feature Engineering

2.1 Remove Null Values

Prompt:

1. I have a wine dataset, the features are Name Variety Origin Attr_1 Attr_2 Alcohol_vol Alcohol_percentage Winemaker_notes Review Avg_rating N_ratings Price_Out-of-stock Price 13 features all.

checking for any entries that are missing a winemaker note or have null values for the wine's provenance. discard any records that do not contain a review.

```
# Step 1: Remove records with missing or null values in 'Winemaker_notes' or 'Origin'
df = df.dropna(subset=['Winemaker_notes', 'Origin'])
# Step 2: Discard any records that do not contain a review
df = df.dropna(subset=['Review'])
# Optional: Reset the index for the cleaned dataframe
df = df.reset_index(drop=True)
```

2.2 Remove and Build new feature

Prompt:

2. remove any non-numeric characters (in this case, "the") from the "Price-Out-of-Stock" and "Price" columns.

```
df['Price_Out-of-stock'] = df['Price_Out-of-stock'].astype(str).str.replace(r'^\0-9.', '',
regex=True)
df['Price'] = df['Price'].astype(str).str.replace(r'^\0-9.', '', regex=True)

# Step 4: Convert cleaned columns to numeric data types (if necessary)
df['Price_Out-of-stock'] = pd.to_numeric(df['Price_Out-of-stock'], errors='coerce')
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
```

3. Convert the values to float types, then combine them into a single feature named "Price_Feature."

```
df['Price_Feature'] = df['Price'].combine_first(df['Price_Out-of-stock'])
```

4. 'Origin' entries are formatted inconsistently, some only list the country, some list the state. For example "Rhône, France" Split any entries containing multiple locations in the 'Origin' column into separate columns, labeling each appropriately as 'Country' or 'Region'. For wines originating from the United States that only list the state, assign "United States" as the country of origin.

```
# Step 6: Split 'Origin' into 'Country' and 'Region'
```

```

us_states = [
    'Alabama', 'Alaska', 'Arizona', 'Arkansas', 'California', 'Colorado', 'Connecticut',
    'Delaware', 'Florida', 'Georgia', 'Hawaii', 'Idaho', 'Illinois', 'Indiana', 'Iowa',
    'Kansas', 'Kentucky', 'Louisiana', 'Maine', 'Maryland', 'Massachusetts', 'Michigan',
    'Minnesota', 'Mississippi', 'Missouri', 'Montana', 'Nebraska', 'Nevada', 'New Hampshire',
    'New Jersey', 'New Mexico', 'New York', 'North Carolina', 'North Dakota', 'Ohio',
    'Oklahoma', 'Oregon', 'Pennsylvania', 'Rhode Island', 'South Carolina', 'South Dakota',
    'Tennessee', 'Texas', 'Utah', 'Vermont', 'Virginia', 'Washington', 'West Virginia',
    'Wisconsin', 'Wyoming'
]

def split_origin(origin):
    parts = [part.strip() for part in origin.split(',')]
    if len(parts) == 1:
        return pd.Series([parts[0], None])

    elif len(parts) >= 2:
        country = parts[len(parts)-1]
        region = parts[len(parts)-2]
        zone = parts[len(parts)-3] if len(parts) >= 3 else None
        # Assign "United States" if the region is a known US state
        if country in us_states:
            country = "United States"
        return pd.Series([country, region, zone])
    return pd.Series([None, None]) # Handle any unexpected cases

# Apply the split_origin function to the 'Origin' column
df[['Country', 'Region', 'Zone']] = df['Origin'].apply(split_origin)

```

2.3 Extract Feature

Prompt:

5. Extract the year from the wine's name and create a new feature, "Year," to store this information. Do not retain decimals

```

import re

def extract_year(name):
    if pd.isna(name):
        return None

    # Find a four-digit number (year) in the name
    match = re.search(r'\b(20\d{2}|19\d{2})\b', name)
    return int(match.group(0)) if match else None

# Apply the extract_year function to the 'Name' column
df['Year'] = df['Name'].apply(extract_year)

```


For the Final Results , we can see we satisfy the requirements

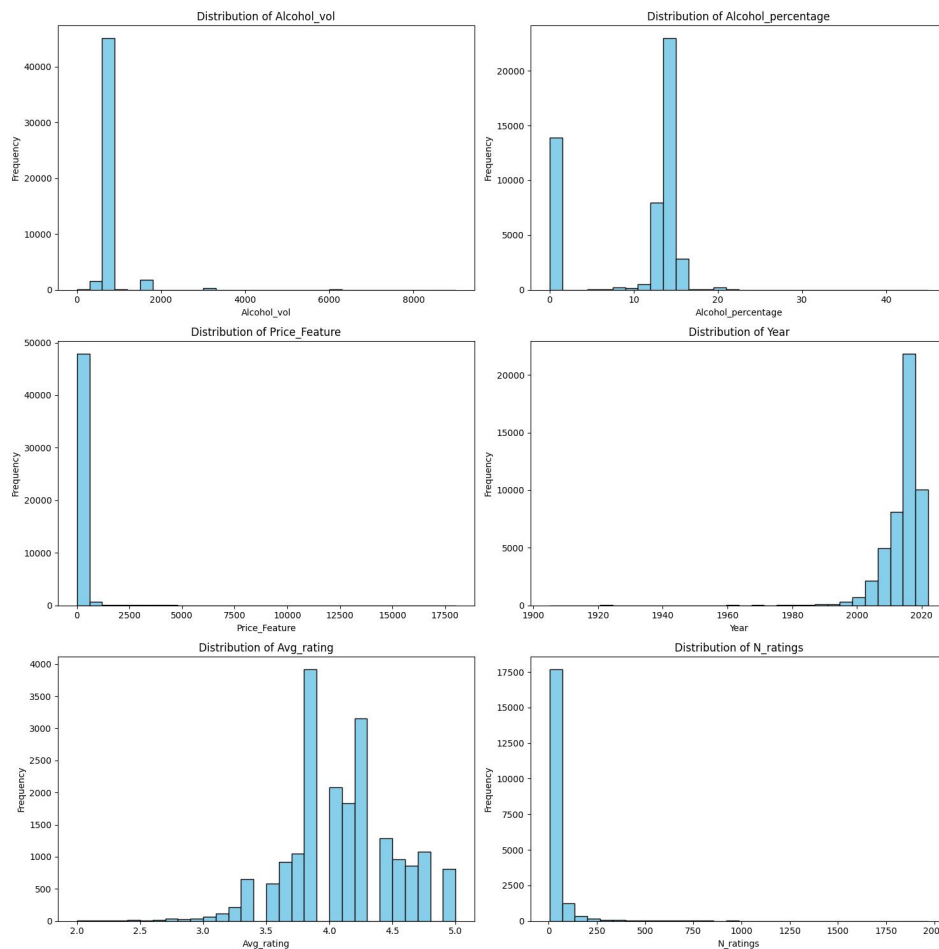
```
[52] df.columns  
  
Index(['Name', 'Variety', 'Attr_1', 'Attr_2', 'Alcohol_vol',  
      'Alcohol_percentage', 'Winemaker_notes', 'Review', 'Avg_rating',  
      'N_ratings', 'Price_Feature', 'Country', 'Region', 'Zone', 'Year'],  
      dtype='object')
```

2.4 Create 6 Histograms

Prompt:

6. Create 6 histograms to display the distribution of each numerical feature.

```
# List of numerical features to plot  
numerical_features = [  
    'Alcohol_vol',  
    'Alcohol_percentage',  
    'Price_Feature',  
    'Year',  
    'Avg_rating',  
    'N_ratings' # Ensure to include all relevant numerical features  
]  
  
# Set up the plotting area  
plt.figure(figsize=(15, 15))  
  
# Create histograms for each numerical feature  
for i, feature in enumerate(numerical_features):  
    plt.subplot(3, 2, i + 1) # Adjust the grid size as needed  
    plt.hist(df[feature].dropna(), bins=30, color='skyblue', edgecolor='black')  
    plt.title(f'Distribution of {feature}')  
    plt.xlabel(feature)  
    plt.ylabel('Frequency')  
  
# Adjust layout for better spacing  
plt.tight_layout()  
plt.show()
```



2.5 Re-draw Figures 2, 3, and 5

Prompt:

7. Visualize Top ten countries with the most wines in the dataset

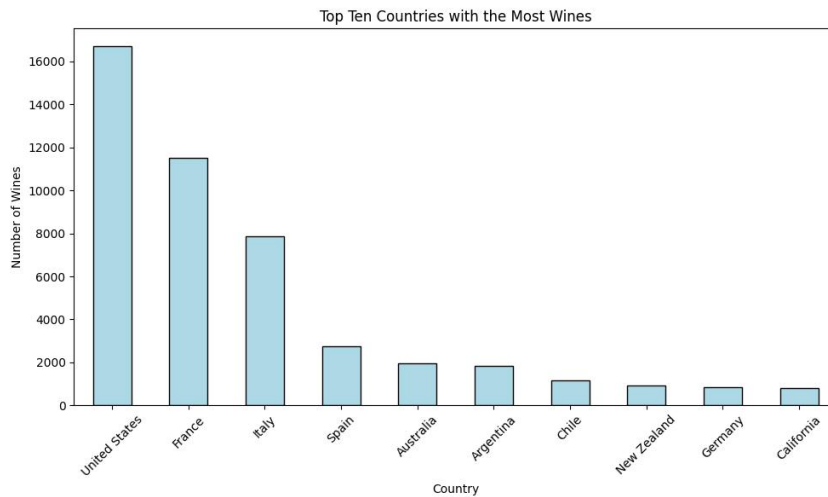
```
# Step 1: Count the number of wines for each country
country_counts = df['Country'].value_counts()

# Step 2: Select the top ten countries
top_ten_countries = country_counts.head(10)

# Step 3: Plot the bar chart
plt.figure(figsize=(10, 6))
top_ten_countries.plot(kind='bar', color='lightblue', edgecolor='black')

# Step 4: Add titles and labels
plt.title('Top Ten Countries with the Most Wines')
plt.xlabel('Country')
plt.ylabel('Number of Wines')
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()
```



8. Visualize Top ten countries with the most wines in the dataset

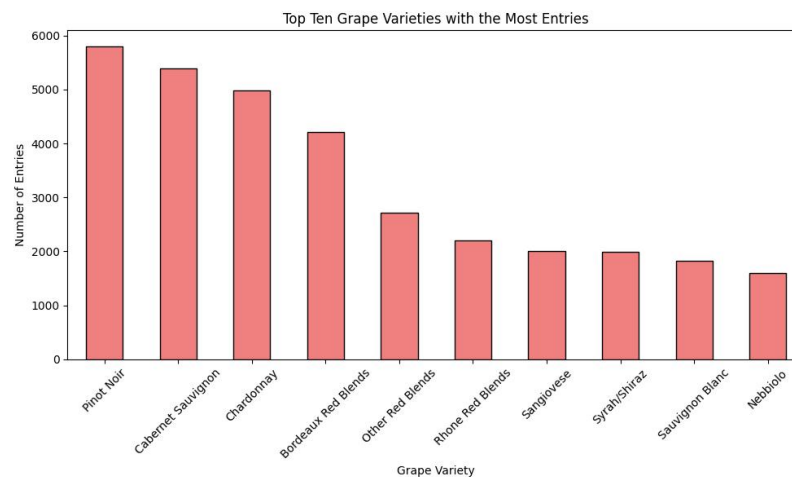
```
# Step 1: Count the number of entries for each grape variety
variety_counts = df['Variety'].value_counts()

# Step 2: Select the top ten grape varieties
top_ten_varieties = variety_counts.head(10)

# Step 3: Plot the bar chart
plt.figure(figsize=(10, 6))
top_ten_varieties.plot(kind='bar', color='lightcoral', edgecolor='black')

# Step 4: Add titles and labels
plt.title('Top Ten Grape Varieties with the Most Entries')
plt.xlabel('Grape Variety')
plt.ylabel('Number of Entries')
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()
```



9. Visualize Average user rating from the top ten countries

```
# Step 1: Count the number of wines for each country
country_counts = df['Country'].value_counts()

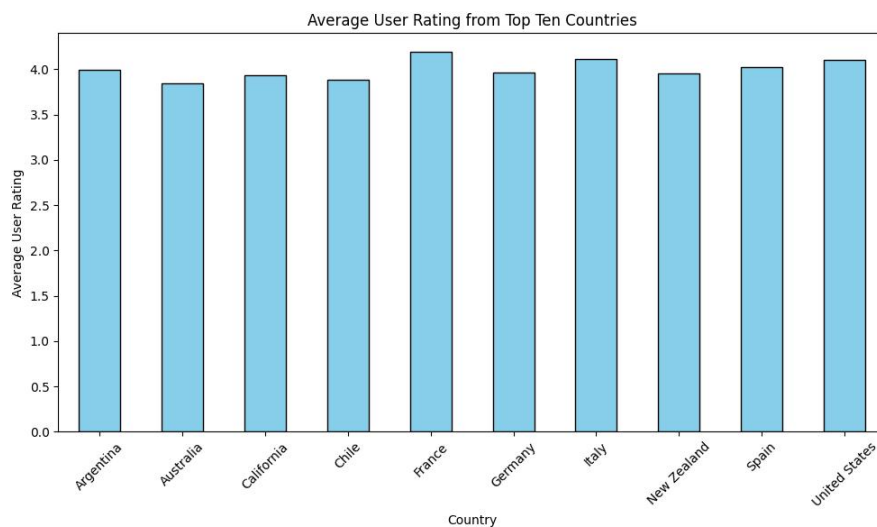
# Step 2: Get the top ten countries
top_ten_countries = country_counts.head(10).index

# Step 3: Calculate the average rating for these countries
average_ratings =
df[df['Country'].isin(top_ten_countries)].groupby('Country')['Avg_rating'].mean()

# Step 4: Plot the bar chart
plt.figure(figsize=(10, 6))
average_ratings.plot(kind='bar', color='skyblue', edgecolor='black')

# Step 5: Add titles and labels
plt.title('Average User Rating from Top Ten Countries')
plt.xlabel('Country')
plt.ylabel('Average User Rating')
plt.xticks(rotation=45)

# Show the plot
plt.tight_layout()
plt.show()
```



You can compare these three graphs with the graphs created by the raw data. We can now see the differences about the average user ratings and the most wines.

Code

[2024 KTH AUS/Big Data/Lab2/Big Data lab2.ipynb](#) at [main](#) • [MaYangle/2024 KTH AUS \(github.com\)](#)