Download the dataset from: https://github.com/bellawillrise/Introduction-to-Numerical-Computing-in-Python/

Submit a pdf file, which is a rendered saved version of the jupyter notebook. Make sure to execute all the codes so the output can be viewed in the pdf.

Also include the link to the public github repository where the jupyter notebook for the assignment is uploaded.

Link to the github repository: https://github.com/HelloCigar/CMSC-197-ML-JupyterPractice

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
# %matplotlib inline
```

In [6]:
```
data = pd.read_csv("data/movie_metadata_cleaned.csv")
```

In [7]:
```
data.head(2)
```

Out[7]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	direct
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	

2 rows × 29 columns

Get the top 10 directors with most movies directed and use a boxplot for their gross earnings

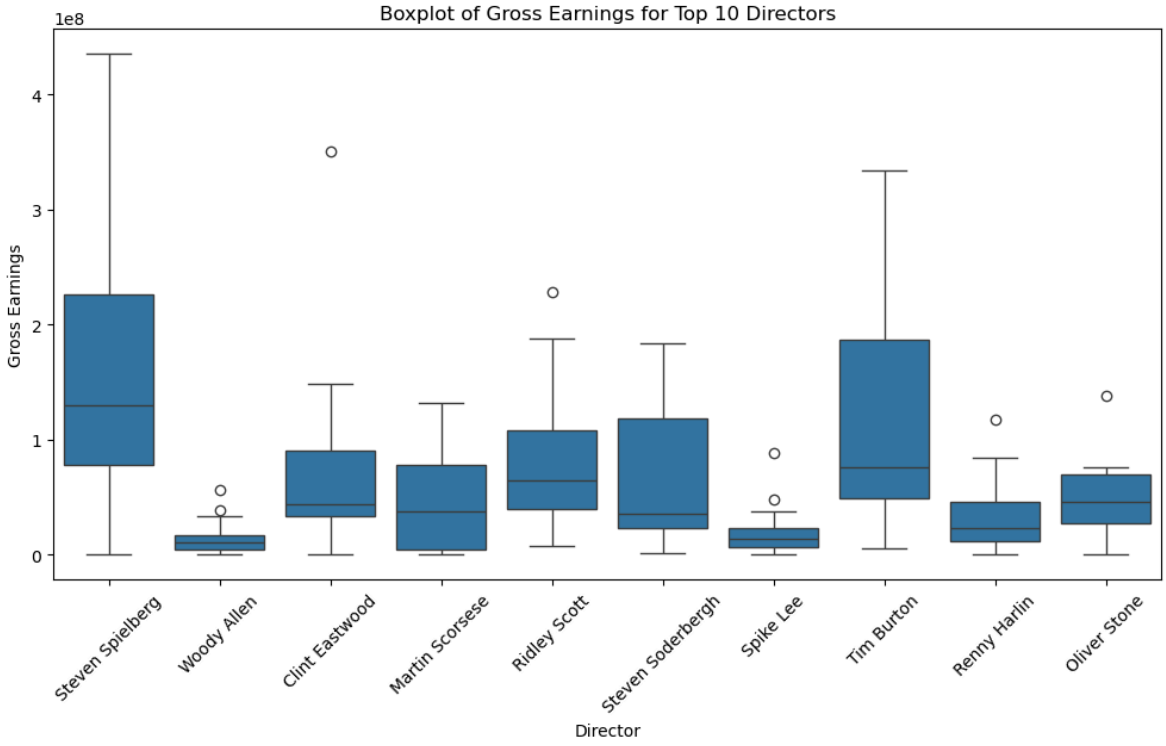
In [9]:
```
#load the data
data = pd.read_csv("data/movie_metadata_cleaned.csv")

director_counts = data.groupby('director_name').size().reset_index(name='movie_c
director_counts = director_counts.sort_values(by='movie_count', ascending=False)

#filter to exclude director_name = '0'
director_counts = director_counts[director_counts['director_name'] != '0']
top_10_directors = director_counts[:10]
```

```
#filter to only include the movies made by the top 10 directors
top_directors_movies = data[data['director_name'].isin(top_10_directors['directo

# show boxplot
plt.figure(figsize=(12, 6))
sns.boxplot(x='director_name', y='gross', data=top_directors_movies, order=top_1
plt.xticks(rotation=45)
plt.title('Boxplot of Gross Earnings for Top 10 Directors')
plt.xlabel('Director')
plt.ylabel('Gross Earnings')
plt.show()
```



Plot the following variables in one graph:

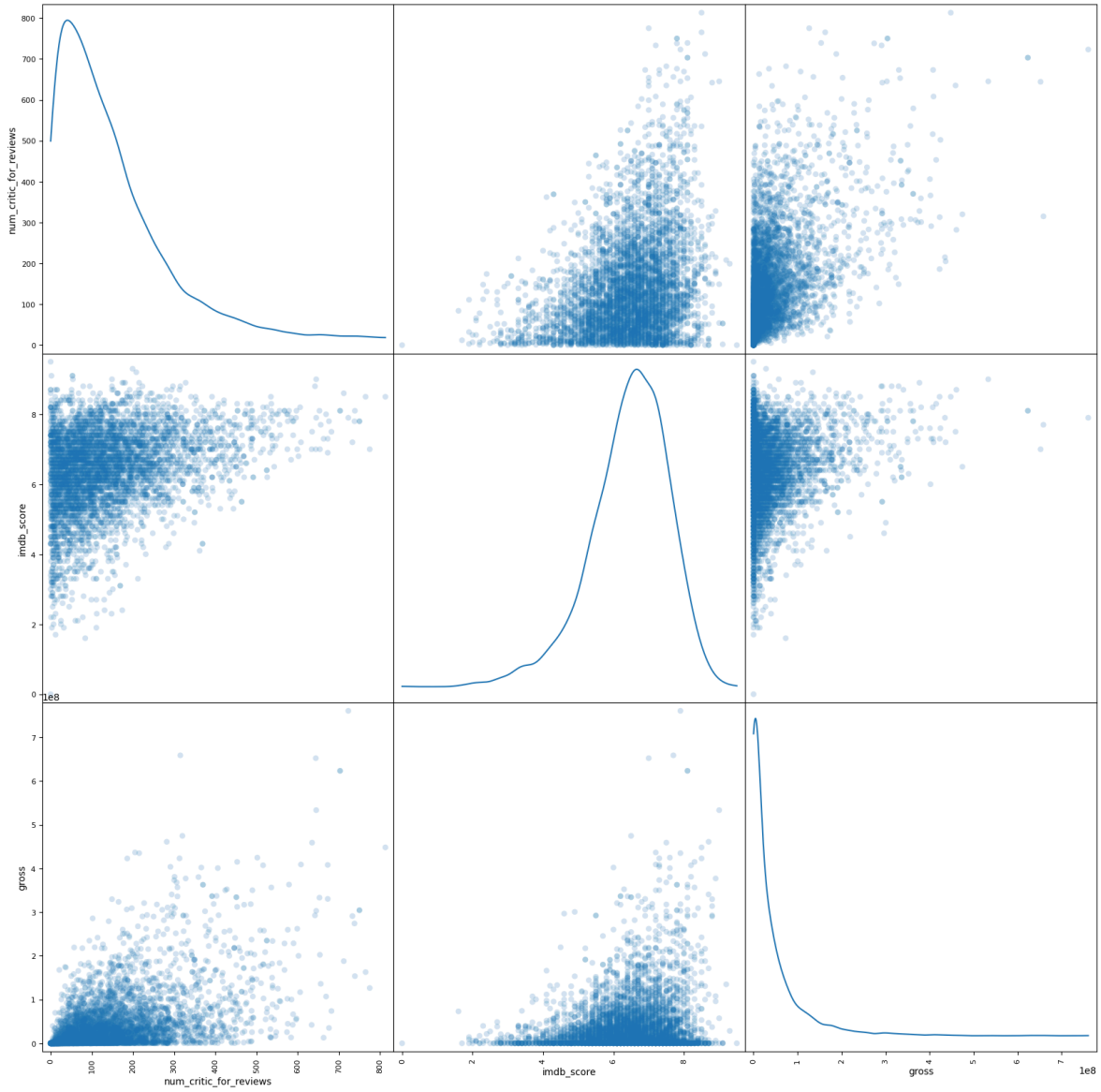
- num_critic_for_reviews
- IMDB score
- gross

In [11]:
```
from pandas.plotting import scatter_matrix

#specify the columns to include in the scatter matrix
cols = ["num_critic_for_reviews", "imdb_score", "gross"]

#generate a scatter matrix plot for the specified columns
scatter_matrix(data[cols], alpha=0.2, figsize=(20, 20), diagonal='kde', marker='

#show the scatter matrix plot
plt.show()
```



Compute Sales (Gross - Budget), add it as another column

In [13]:
```
#compute the 'sales' by subtracting the 'budget' from the 'gross' revenue
data['sales'] = data['gross'] - data['budget']

#view the updated data
data
```

Out[13]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	di
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	
...	...	...	...	...	...	...	
5039	5039	b'The Following '	Color	0	43.0	43.0	
5040	5040	b'A Plague So Pleasant'	Color	Benjamin Roberds	13.0	76.0	
5041	5041	b'Shanghai Calling'	Color	Daniel Hsia	14.0	100.0	
5042	5042	b'My Date with Drew'	Color	Jon Gunn	43.0	90.0	
5043	5043	b'Starting Over Again'	0	Olivia Lamasan	0.0	0.0	

5044 rows × 30 columns

Which directors garnered the most total sales?

In [15]:
```
#group the data by 'director_name' and calculate the total 'sales' for each dire
total_sales = data.groupby('director_name')['sales'].sum().reset_index()

#sort the directors by total sales in descending order
total_sales = total_sales.sort_values(by='sales', ascending=False)

#select the top 10 directors with the highest total sales
top_directors_by_sales = total_sales[:10]

#display the sales data for the top 10 directors
top_directors_by_sales
```

Out[15]:

	director_name	sales
2159	Steven Spielberg	2.451332e+09
765	George Lucas	1.386641e+09
923	James Cameron	1.199626e+09
1219	Joss Whedon	1.000887e+09
335	Chris Columbus	9.417076e+08
1787	Peter Jackson	9.009693e+08
2221	Tim Burton	8.242755e+08
374	Christopher Nolan	8.082276e+08
1158	Jon Favreau	7.693815e+08
695	Francis Lawrence	7.555020e+08

Plot sales and average likes as a scatterplot. Fit it with a line.

In [17]:

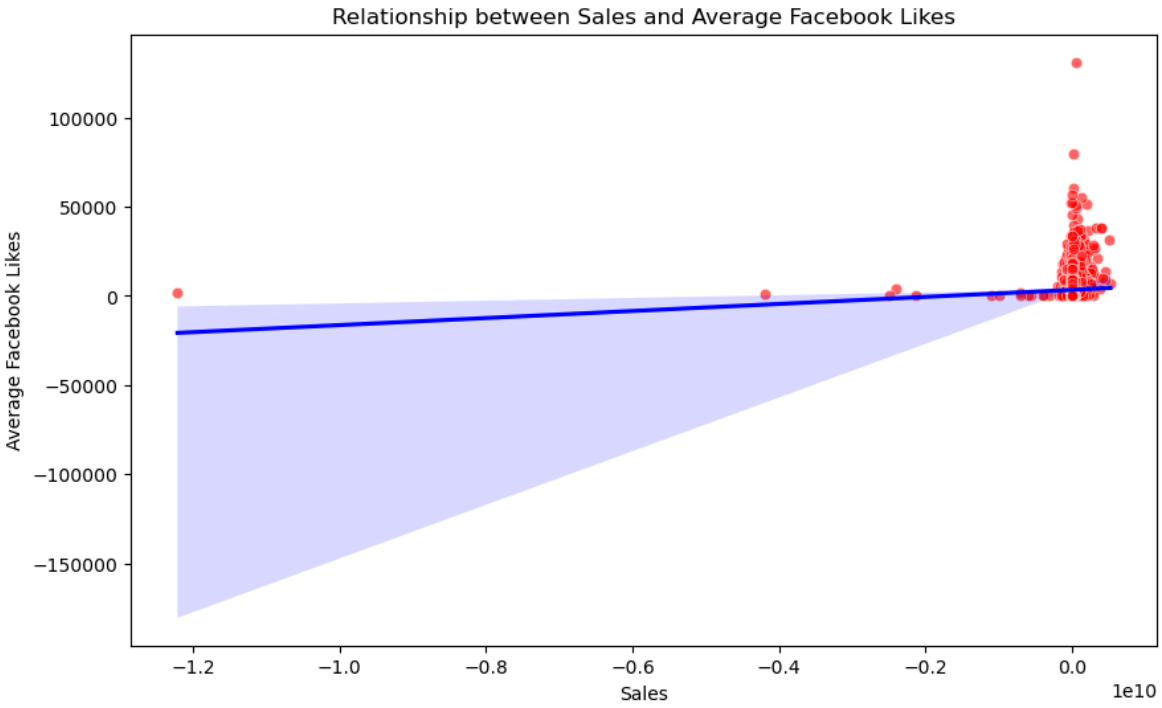
```
#initialize a figure with a specified size for the scatter plot
plt.figure(figsize=(10, 6))

#plot a scatterplot to visualize the relationship between sales and the average
#across directors, main actors
sns.scatterplot(
    x='sales',
    y=data[
            [
                'director_facebook_likes',
                'actor_1_facebook_likes',
                'actor_2_facebook_likes',
                'actor_3_facebook_likes',
                'movie_facebook_likes'
            ]
        ].mean(axis=1),
    data=data, color='red', alpha=0.6
)

#add a regression line
sns.regplot(
    x='sales',
    y=data[
            [
                'director_facebook_likes',
                'actor_1_facebook_likes',
                'actor_2_facebook_likes',
                'actor_3_facebook_likes',
                'movie_facebook_likes'
            ]
        ].mean(axis=1),
    data=data, scatter=False, color='blue'
)
```

```
#set the plot title and axis labels for better readability
plt.title('Relationship between Sales and Average Facebook Likes')
plt.xlabel('Sales')
plt.ylabel('Average Facebook Likes')

#display the scatter plot with the line
plt.show()
```



Which of these genres are the most profitable? Plot their sales using different histograms, superimposed in the same axis.

- Romance
- Comedy
- Action
- Fantasy

In [19]:
```
#plot a histogram showing the distribution of sales for the 'Romance' genre, sto
plot = sns.histplot(
    data[data['genres'] == 'Romance']["sales"],
    color="red",
    label="Romance",
    kde=False,
    stat="density",
    linewidth=0
)

#add a histogram for the 'Comedy' genre sales to the plot
sns.histplot(
    data[data['genres'] == 'Comedy']["sales"],
    color="teal",
    label="Comedy",
    kde=False,
    stat="density",
```

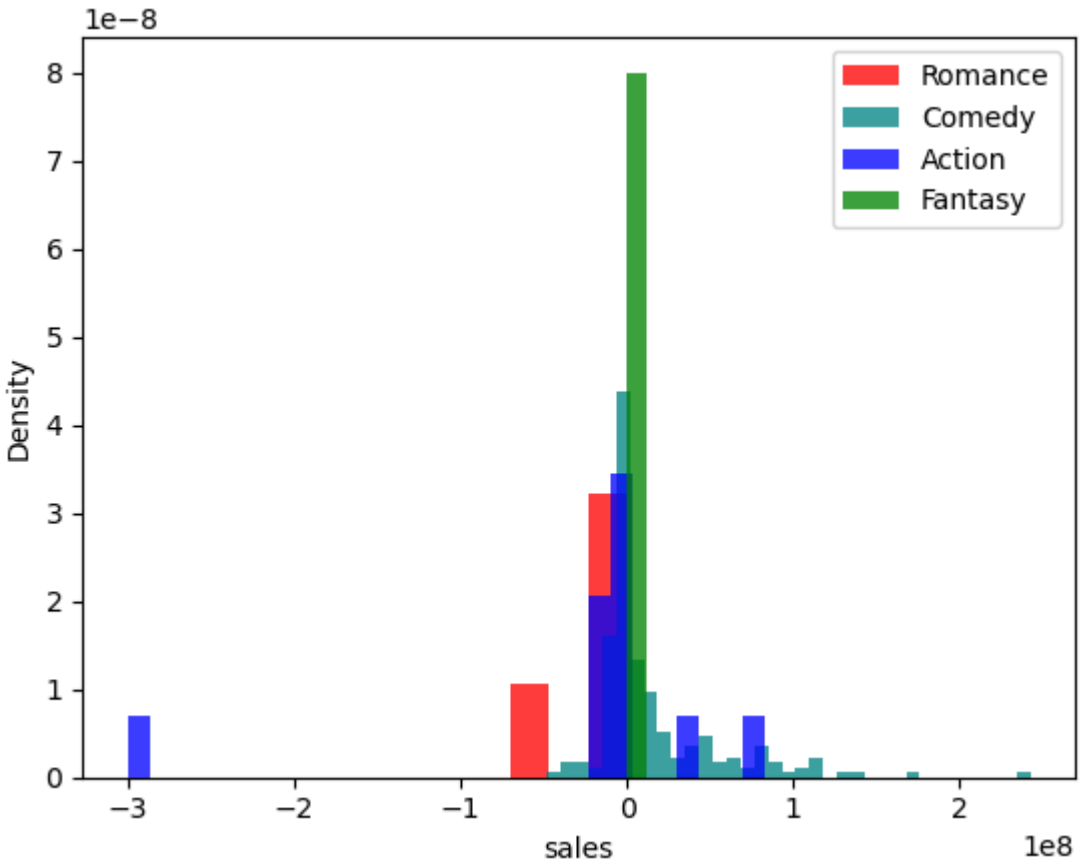
```
    linewidth=0
)

#add a histogram for the 'Action' genre sales to the plot
sns.histplot(
    data[data['genres'] == 'Action']["sales"],
    color="blue",
    label="Action",
    kde=False,
    stat="density",
    linewidth=0
)

#add a histogram for the 'Fantasy' genre sales to the plot
sns.histplot(
    data[data['genres'] == 'Fantasy']["sales"],
    color="green",
    label="Fantasy",
    kde=False,
    stat="density",
    linewidth=0
)

# Display the legend to differentiate between the genres
plot.legend()
```

Out[19]: <matplotlib.legend.Legend at 0x1a8e684d040>



seems that Fantasy is thriving

For each of movie, compute average likes of the three actors and store it as a new variable

Read up on the mean function.

Store it as a new column, average_actor_likes.

In [22]:
```
#calculate the average Facebook likes for the three actors in each movie
data['average_actor_likes'] = data[['actor_1_facebook_likes', 'actor_2_facebook_

#display the updated data with 'average_actor_likes' column added
data
```

Out[22]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	di
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	
...	...	...	...	...	...	...	
5039	5039	b'The Following '	Color	0	43.0	43.0	
5040	5040	b'A Plague So Pleasant'	Color	Benjamin Roberds	13.0	76.0	
5041	5041	b'Shanghai Calling'	Color	Daniel Hsia	14.0	100.0	
5042	5042	b'My Date with Drew'	Color	Jon Gunn	43.0	90.0	
5043	5043	b'Starting Over Again'	0	Olivia Lamasan	0.0	0.0	

5044 rows × 31 columns

Copying the whole dataframe

In [24]:
```
df = data.copy()
df.head()
```

Out[24]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	direct
0	0	b'Avatar'	Color	James Cameron	723.0	178.0	
1	1	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	302.0	169.0	
2	2	b'Spectre'	Color	Sam Mendes	602.0	148.0	
3	3	b'The Dark Knight Rises'	Color	Christopher Nolan	813.0	164.0	
4	4	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.0	0.0	

5 rows × 31 columns

Min-Max Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

The min-max approach (often called normalization) rescales the feature to a hard and fast range of [0,1] by subtracting the minimum value of the feature then dividing by the range. We can apply the min-max scaling in Pandas using the .min() and .max() methods.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Normalize each numeric column (those that have types integer or float) of the copied dataframe (df)

In [27]:
```
# Identify all numeric columns in the DataFrame (integers and floats)
numeric_columns = df.select_dtypes(include=['int64', 'float64']).columns
```

```
# Create a copy of the original DataFrame to store normalized values
normalized_dataframe = df.copy()

# Iterate over each numeric column to perform normalization
for column in numeric_columns:
    # Find the minimum and maximum values of the current column
    min_value = df[column].min()
    max_value = df[column].max()

    # Compute the range of values in the current column
    range_value = max_value - min_value

    # Apply min-max normalization to scale values between 0 and 1
    normalized_dataframe[column] = (df[column] - min_value) / range_value

# Output the DataFrame with normalized numeric values
normalized_dataframe
```

Out[27]:

	Unnamed: 0	movie_title	color	director_name	num_critic_for_reviews	duration	di
0	0.000000	b'Avatar'	Color	James Cameron	0.889299	0.941799	
1	0.000198	b"Pirates of the Caribbean: At World's End"	Color	Gore Verbinski	0.371464	0.894180	
2	0.000397	b'Spectre'	Color	Sam Mendes	0.740467	0.783069	
3	0.000595	b'The Dark Knight Rises'	Color	Christopher Nolan	1.000000	0.867725	
4	0.000793	b'Star Wars: Episode VII - The Force Awakens ...	0	Doug Walker	0.000000	0.000000	
...	...	...	...	...	...	...	
5039	0.999207	b'The Following '	Color	0	0.052891	0.227513	
5040	0.999405	b'A Plague So Pleasant'	Color	Benjamin Roberds	0.015990	0.402116	
5041	0.999603	b'Shanghai Calling'	Color	Daniel Hsia	0.017220	0.529101	
5042	0.999802	b'My Date with Drew'	Color	Jon Gunn	0.052891	0.476190	
5043	1.000000	b'Starting Over Again'	0	Olivia Lamasan	0.000000	0.000000	

5044 rows × 31 columns

In [ ]: