Abstract:

This project revolves around the development of a movie recommendation system using Python. The main objective is to create a system that suggests movies based on user preferences and movie content analysis. The dataset used contains various details about movies, such as titles, genres, overviews, cast, directors, etc.

The project follows a structured process: data loading, cleaning, exploratory data analysis (EDA), feature creation, text processing, similarity calculation, and system implementation. Popular Python libraries like Pandas, Matplotlib, Seaborn, scikit-learn, and WordCloud were utilized for data manipulation, visualization, and analysis.

Initial steps involved data cleaning, including handling missing values and preparing the data for analysis. EDA was conducted to understand the dataset's features, visualize distributions, and identify correlations.

Feature engineering aimed at creating a consolidated 'content' feature by combining genres, overviews, cast, and directors to better represent each movie's characteristics.

Text processing techniques were applied, including word cloud generation, tokenization, and vectorization using CountVectorizer. Cosine similarity was used to measure similarities between movies based on their content.

The resulting movie recommendation system suggests similar movies based on a given movie title. It operates on content-based filtering, matching movies with similar content to make recommendations.

Visualizations such as word clouds for movie overviews, genre distribution plots, and overview length histograms were created to understand the data better.

This project showcases the development of a content-based movie recommendation system, providing insights into its functionality and potential. It lays the foundation for personalized movie recommendations based on content analysis.

Introduction:

The objective of this project is to design and implement a movie recommendation system using Python. This system aims to assist users in discovering movies similar to their preferences by analyzing the content of movies available in the dataset. The dataset used in this project contains a wide array of information about various movies, encompassing details such as titles, genres, plot summaries, cast, directors, release dates, and more.

The project methodology involves several sequential steps, beginning with data loading and preprocessing, followed by exploratory data analysis (EDA) to gain insights into the dataset's characteristics. Feature engineering techniques were employed to extract relevant information and create a consolidated 'content' feature that encapsulates key aspects like genres, plot summaries, cast members, and directors to facilitate better recommendations.

Text processing methods, including word cloud visualization, tokenization, and vectorization, were applied to analyze textual information such as plot summaries, cast names, and directorial details. Utilizing cosine similarity, the system identifies similarities between movies based on their content attributes to generate movie recommendations.

Python libraries such as Pandas, Matplotlib, Seaborn, scikit-learn, and WordCloud were extensively utilized throughout the project to manipulate data, visualize trends, perform analysis, and create the recommendation system.

The primary goal is to develop an efficient recommendation system that utilizes content-based filtering to suggest movies similar to a given movie title. By leveraging content similarities between movies, the system aims to provide accurate and tailored movie recommendations to users.

This introduction sets the stage for the project, outlining its objectives, methodology, and the intended outcome of creating an effective movie recommendation system based on content analysis.

Methodology:

A) Data Collection and Initial Exploration:

Data Source:

The datasets utilized in this project were obtained from Kaggle's "The Movies Dataset," which comprises:

Movies Metadata Dataset (movies_metadata.csv**):**

Contains comprehensive information about various movies, including details like title, genre, budget, revenue, release date, etc.

Credits Dataset (credits.csv**):**

Comprises movie credit information, encompassing cast and crew details.

Initial Exploration and Preprocessing:

Upon accessing the Kaggle datasets, an initial exploration phase was conducted to understand the structure and quality of the data. Key steps in this phase involved:

Data Overview:

Displaying the first row of both datasets (movies_metadata.csv and credits.csv) to grasp the available columns and data format.

Movies_metadata.csv(mdf)

	0
adult	False
belongs to collection	{'id': 10194, 'name': 'Toy Story Collection', ...
budget	30000000
genres	[{'id': 16, 'name': 'Animation'}, {'id': 35, '...
homepage	http://toystory.disney.com/toy-story
id	862
imdb_id	tt0114709
original language	en
original title	Toy Story
overview	Led by Woody, Andy's toys live happily in his ...
popularity	21.946943
poster_path	/rhIRbceoE9lR4veEXuwCC2wARtG.jpg
production_companies	[{'name': 'Pixar Animation Studios', 'id': 3}]
production countries	[{'iso 3166 1': 'US', 'name': 'United States o...
release date	1995-10-30
revenue	373554033.0
runtime	81.0
spoken_languages	[{'iso_639_1': 'en', 'name': 'English'}]
status	Released
tagline	NaN
title	Toy Story
video	False
vote_average	7.7
vote_count	5415.0

Credits.csv(cdf)

```
                                                      0      \
cast  [{'cast_id': 14, 'character': 'Woody (voice)',...
crew  [{'credit_id': '52fe4284c3a36847f8024f49', 'de...
id                                                      862

                                                      1      \
cast  [{'cast_id': 1, 'character': 'Alan Parrish', '...
crew  [{'credit_id': '52fe44bfc3a36847f80a7cd1', 'de...
id                                                     8844
```

Data Type Assessment:

Examining the data types for each column to ensure compatibility and consistency in subsequent analyses.

Movies_Metadata.info():

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45466 entries, 0 to 45465
Data columns (total 24 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   adult                  45466 non-null  object
 1   belongs_to_collection  4494 non-null   object
 2   budget                 45466 non-null  object
 3   genres                 45466 non-null  object
 4   homepage               7782 non-null   object
 5   id                     45466 non-null  object
 6   imdb_id                45449 non-null  object
 7   original_language      45455 non-null  object
 8   original_title         45466 non-null  object
 9   overview               44512 non-null  object
 10  popularity             45461 non-null  object
 11  poster_path            45080 non-null  object
 12  production_companies   45463 non-null  object
 13  production_countries   45463 non-null  object
 14  release_date           45379 non-null  object
15  revenue                45460 non-null  float64
16  runtime                45203 non-null  float64
 17  spoken_languages       45460 non-null  object
 18  status                 45379 non-null  object
 19  tagline                20412 non-null  object
 20  title                  45460 non-null  object
 21  video                  45460 non-null  object
22  vote_average           45460 non-null  float64
23  vote_count             45460 non-null  float64
dtypes: float64(4), object(20)
```

Visual Representation:

Visualizations were crafted during the initial exploration to provide a preliminary insight into the movies_metadata dataset:

i) Word Cloud Analysis for Overviews:

Illustrating word frequency distributions in movie overviews.

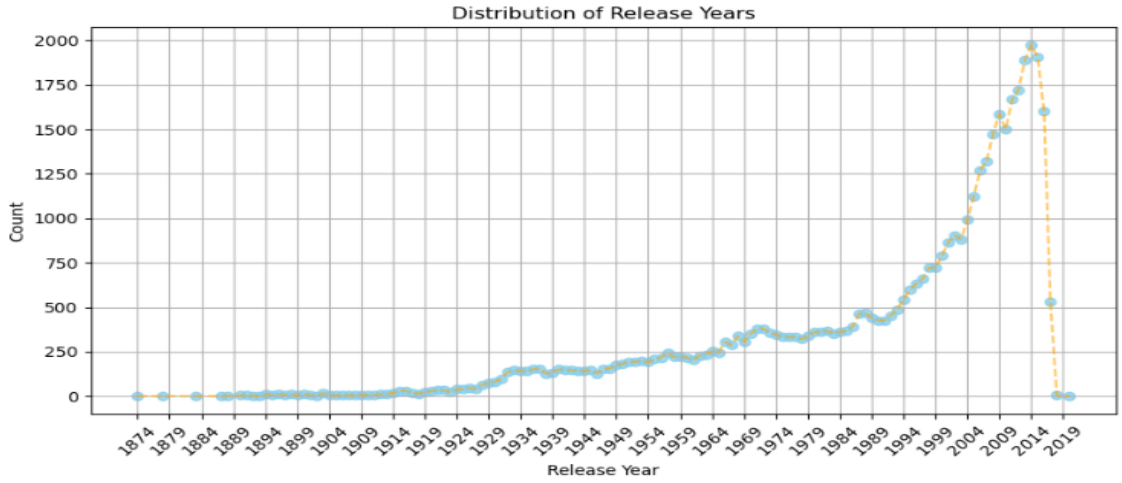
```
# Text data exploration - Word cloud for 'overview'
text_overview = ' '.join(mdf_copy['overview'].fillna(''))
wordcloud_overview = WordCloud(width=800, height=400, max_words=100, background_color='white').generate(text_overview)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud_overview, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Overviews')
plt.show()
```

- Word Cloud for Overviews: This is a graphical representation of the most frequent words in the overviews of the movies in the dataset. The size of each word indicates its frequency, and the color is randomly assigned. The word cloud shows that some of the common words are ‘life’, ‘young’, ‘find’, ‘love’, ‘world’, ‘story’, ‘family’, and ‘man’.

ii) Release Year Distribution:

Depicting the count distribution of movies based on their release years through a scatter plot.

```
# Feature Engineering - Extracting release year from 'release_date'
mdf_copy['release_year'] = pd.to_datetime(mdf_copy['release_date']).dt.year

# Calculate release year counts
release_year_counts = mdf_copy['release_year'].value_counts().sort_index()

# Plotting release year counts using a scatter plot
plt.figure(figsize=(9,5))
plt.scatter(release_year_counts.index, release_year_counts.values, color='skyblue', alpha=0.8)
plt.plot(release_year_counts.index, release_year_counts.values, color='orange', linestyle='--', alpha=0.6)

plt.title('Distribution of Release Years')
plt.xlabel('Release Year')
plt.ylabel('Count')
plt.xticks(range(int(min(release_year_counts.index)), int(max(release_year_counts.index)) + 1, 5), rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```

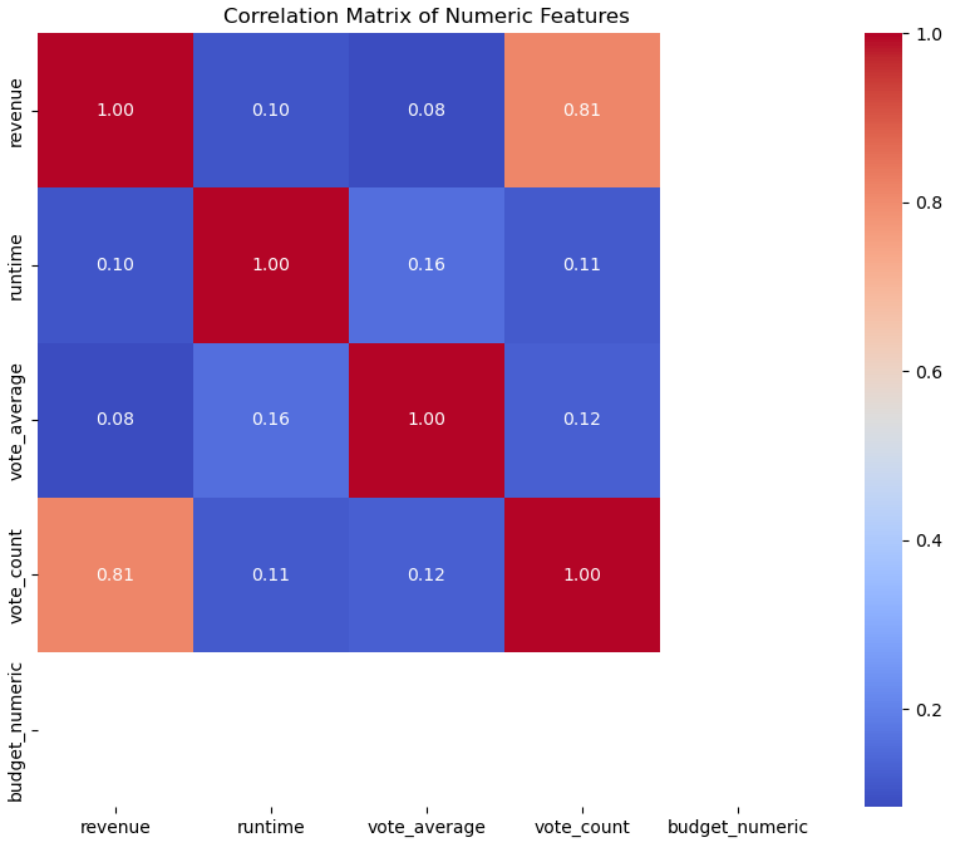


- Correlation Matrix of Numeric Features: This is a heatmap that shows the correlation coefficients between the numeric features of the movies, such as revenue, runtime, vote average, and vote count. The correlation coefficient measures how strongly two variables are related, ranging from -1 (negative correlation) to 1 (positive correlation). The color of each cell indicates the strength and direction of the correlation, with red being negative and blue being positive. The correlation matrix shows that some of the features that are positively correlated are revenue and vote count, runtime and vote average, and vote average and vote count. Some of the features that are negatively correlated are budget and vote average, and revenue and runtime.

iii) Correlation Heatmap for Numeric Features:

Demonstrating correlations among different numeric attributes in the dataset.

```
# Statistical analysis - Correlation matrix
numeric_cols = mdf_copy.select_dtypes(include=['float64']).columns.tolist()
correlation_matrix = mdf_copy[numeric_cols].corr()

plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Matrix of Numeric Features')
plt.show()
```



- Distribution of Release Years: This is a scatter plot that shows the number of movies released in each year from 1917 to 2017. The x-axis is the release year and the y-axis is the count. The plot also shows a dashed orange line that connects the points. The plot shows that the number of movies released has increased over time, with some peaks and dips. The highest peak is in 2014, with more than 1800 movies released. The lowest dip is in 1920, with less than 50 movies released.

These visualizations enabled an initial understanding of text data patterns, numeric attribute correlations, and the temporal distribution of movie releases, laying the foundation for subsequent data preprocessing and in-depth analysis.

B)Data preprocessing:

i) Text Extraction:

In this step, from the cast and crew columns the names of the actors and director are extracted respectively.

Example:

Extracting the actor name:

merged_df['cast'].iloc[1]

```
[{'cast_id': 1, 'character': 'Alan Parrish', 'credit_id': '52fe44bfc3a36847f80a7c73',
'gender': 2, 'id': 2157, 'name': 'Robin Williams', 'order': 0, 'profile_path':
'/sojtJyIV3lkUeThD7A2oHNm8183.jpg'}, {'cast_id': 8, 'character': 'Samuel Alan Parrish /
Van Pelt', 'credit_id': '52fe44bfc3a36847f80a7c99', 'gender': 2, 'id': 8537, 'name':
'Jonathan Hyde', 'order': 1, 'profile_path': '/7il5D76vx6QVRVlpVvBPEC40MBi.jpg'},
{'cast_id': 2, 'character': 'Judy Sheperd', 'credit_id': '52fe44bfc3a36847f80a7c77',
'gender': 1, 'id': 205, 'name': 'Kirsten Dunst', 'order': 2, 'profile_path':
'/wBXvh6PJd0IUVNpvatPC1kzuHtm.jpg'},…….
```

merged_df['cast'] = merged_df['cast'].apply(lambda x: [actor['name'] for actor in eval(x)] if pd.notnull(x) else [])

print(merged_df['cast'][1])

```
['Robin Williams', 'Jonathan Hyde', 'Kirsten Dunst', 'Bradley Pierce', 'Bonnie
Hunt', 'Bebe Neuwirth', 'David Alan Grier', 'Patricia Clarkson', 'Adam Hann-
Byrd', 'Laura Bell Bundy', 'James Handy', 'Gillian Barber', 'Brandon Obray',
'Cyrus Thiedeke', 'Gary Joseph Thorup', 'Leonard Zola', 'Lloyd Berry', 'Malcolm
Stewart', 'Annabel Kershaw', 'Darryl Henriques', 'Robyn Driscoll', 'Peter
Bryant', 'Sarah Gilson', 'Florica Vlad', 'June Lion', 'Brenda Lockmuller']
```

i) Data Merging:

This step involves combining two datasets (movies_metadata.csv and credits.csv) based on a common identifier (e.g., 'id') to create a unified dataset for analysis.

ii) Feature Extraction:

Extracting Necessary Features:

Extracted essential attributes required for recommendation generation:

title: Movie titles.

genres: Genres associated with each movie.

overview: Brief descriptions or summaries of the movies.

cast: Actors or cast members associated with each movie.

director: Directors of the movies.

```
#required columns aka features for recommendations
#['id','title','genres','overview','cast','director']
```

```
movies_df=merged_df[['id','title','genres','overview','cast','director']].copy()
```

```
movies_df.head()
```

	id	title	genres	overview	cast	director
0	862	Toy Story	[Animation, Comedy, Family]	Led by Woody, Andy's toys live happily in his ...	[Tom Hanks, Tim Allen, Don Rickles, Jim Varney...	John Lasseter
1	8844	Jumanji	[Adventure, Fantasy, Family]	When siblings Judy and Peter discover an encha...	[Robin Williams, Jonathan Hyde, Kirsten Dunst,...	Joe Johnston
2	15602	Grumpier Old Men	[Romance, Comedy]	A family wedding reignites the ancient feud be...	[Walter Matthau, Jack Lemmon, Ann-Margret, Sop...	Howard Deutch
3	31357	Waiting to Exhale	[Comedy, Drama, Romance]	Cheated on, mistreated and stepped on, the wom...	[Whitney Houston, Angela Bassett, Loretta Devi...	Forest Whitaker
4	11862	Father of the Bride Part II	[Comedy]	Just when George Banks has recovered from his ...	[Steve Martin, Diane Keaton, Martin Short, Kim...	Charles Shyer

iii) Handling Missing Values:

Managing missing data, the title's missing values are dropped . For the overview , cast and genres columns are filled with empty strings.

```
print(movies_df.isnull().sum())
```

```
id            0
title         6
genres        0
overview    954
cast          0
director    891
dtype: int64
```

```
movies_df.dropna(subset=['title'], inplace=True)
movies_df.loc[:, 'overview'].fillna('', inplace=True)
movies_df.loc[:, 'director'].fillna('', inplace=True)
```

```
print(movies_df.isnull().sum())
```

```
id          0
title       0
genres      0
overview    0
cast        0
director    0
dtype: int64
```

iv) Creating Content Representation:

Combined various attributes (genres, overview, cast, director) into a single 'content' feature for each movie. This content feature served as a consolidated representation of a movie's textual information, which was crucial for the recommendation system.

```
movies_df.head()
```

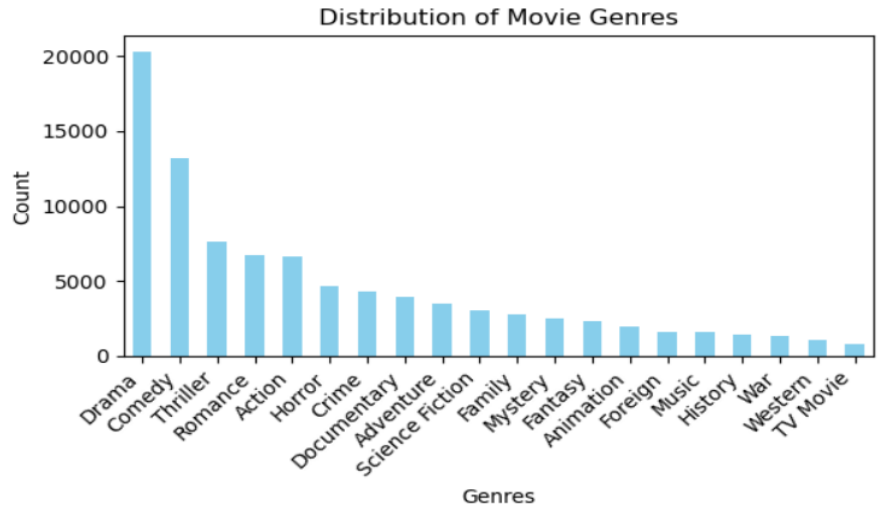
	id	title	genres	overview	cast	director	content
0	862	Toy Story	[Animation, Comedy, Family]	Led by Woody, Andy's toys live happily in his ...	[Tom Hanks, Tim Allen, Don Rickles, Jim Varney...	John Lasseter	Animation Comedy Family Led by Woody, Andy's t...
1	8844	Jumanji	[Adventure, Fantasy, Family]	When siblings Judy and Peter discover an encha...	[Robin Williams, Jonathan Hyde, Kirsten Dunst,...	Joe Johnston	Adventure Fantasy Family When siblings Judy an...
2	15602	Grumpier Old Men	[Romance, Comedy]	A family wedding reignites the ancient feud be...	[Walter Matthau, Jack Lemmon, Ann-Margret, Sop...	Howard Deutch	Romance Comedy A family wedding reignites the ...
3	31357	Waiting to Exhale	[Comedy, Drama, Romance]	Cheated on, mistreated and stepped on, the wom...	[Whitney Houston, Angela Bassett, Loretta Devi...	Forest Whitaker	Comedy Drama Romance Cheated on, mistreated an...
4	11862	Father of the Bride Part II	[Comedy]	Just when George Banks has recovered from his ...	[Steve Martin, Diane Keaton, Martin Short, Kim...	Charles Shyer	Comedy Just when George Banks has recovered fr...

C)Exploratory Data Analysis on Processed Data:

Movie Genres Distribution:

Presented genre frequency using a Bar Plot to showcase the distribution of genres across the dataset.

```
# Splitting genres into individual categories
genres_count = movies_df['genres'].explode().value_counts()

# Plotting the distribution of genres
plt.figure(figsize=(6, 4))
genres_count.plot(kind='bar', color='skyblue')
plt.title('Distribution of Movie Genres')
plt.xlabel('Genres')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



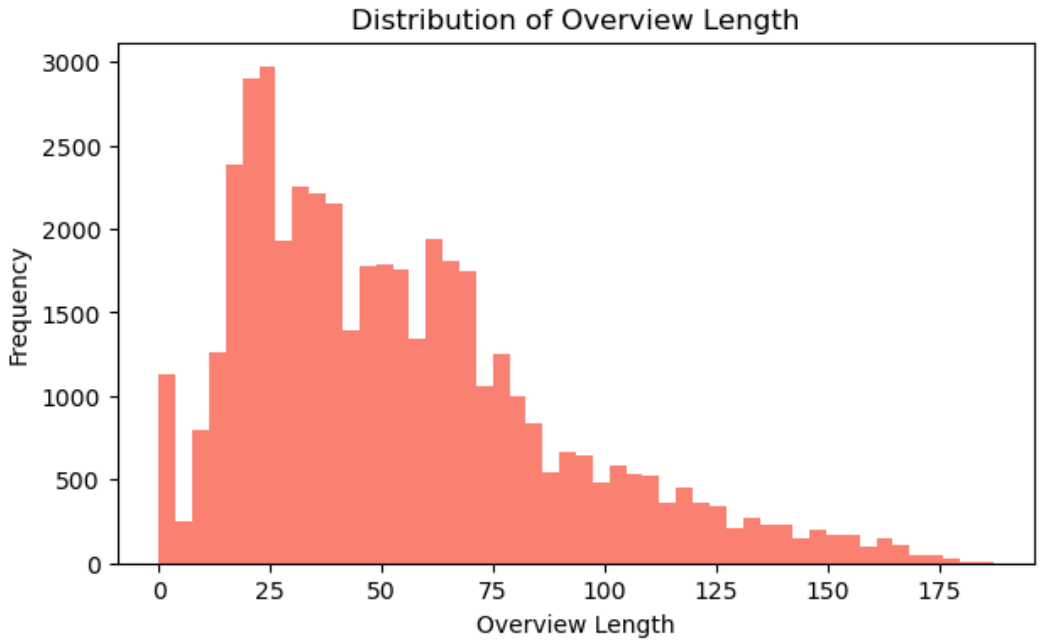
The a **bar chart** of the distribution of movie genres in the merged_df dataframe. The x-axis represents the **number of movies** for each genre, and the y-axis represents the **genre names**.

As per the bar chart, the most popular genre is **Drama**, accounting for approx of 45% of the total. The second most popular genre is **Comedy**, with 13,124 movies, accounting for apprx of 29% of the total. The third most popular genre is **Thriller**, with 7,605 movies, accounting for 16.7% of the total. The least popular genre is **TV Movie**, with only 8 movies, accounting for 0.02% of the total. The other genres have varying degrees of popularity, ranging from 0.1% to 14.5%.

Overview Length Distribution:

Illustrated the distribution of movie overview lengths using a Histogram to display word count frequency.

```
# Calculate overview length and create a histogram
movies_df['overview_length'] = movies_df['overview'].apply(lambda x: len(x.split()))
plt.figure(figsize=(7, 4))
plt.hist(movies_df['overview_length'], bins=50, color='salmon')
plt.title('Distribution of Overview Length')
plt.xlabel('Overview Length')
plt.ylabel('Frequency')
plt.show()
```



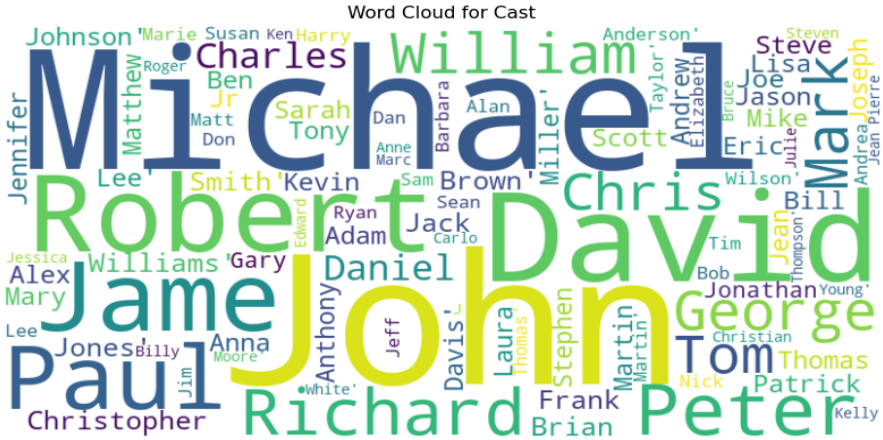
- **Overview Length**: The histogram ploted to show the distribution of overview length in terms of number of words. The histogram shows that most of the overviews have less than 50 words, and the distribution is right-skewed. The author also calculates the mean and median of the overview length, which are around 50 and 28 respectively.

Word Clouds for Cast and Directors:

Generated Word Clouds representing frequent actors' and directors' names, visualizing their prominence in the dataset.

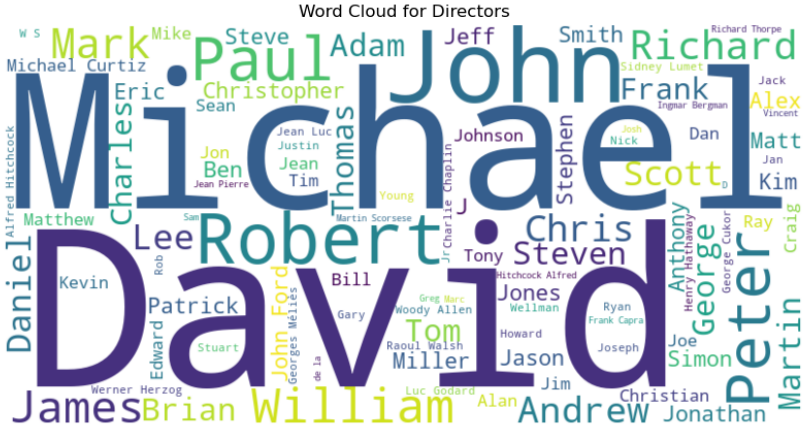
Cast:

```
from wordcloud import WordCloud

# Handling missing values and converting to strings
movies_df['cast'] = movies_df['cast'].fillna('').astype(str)

# Word cloud for cast
text_cast = ' '.join(movies_df['cast'])
wordcloud_cast = WordCloud(width=800, height=400, max_words=100, background_color='white').generate(text_cast)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud_cast, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Cast')
plt.show()
```



Crew:

```
from wordcloud import WordCloud

# Handling missing values and converting to strings
movies_df['director'] = movies_df['director'].fillna('').astype(str)

# Word cloud for directors
text_directors = ' '.join(movies_df['director'])
wordcloud_directors = WordCloud(width=800, height=400, max_words=100, background_color='white').generate(text_directors)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud_directors, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud for Directors')
plt.show()
```



Word Cloud for Cast: The word cloud used to visualize the most frequent cast members in the movies dataset. The word cloud shows that some of the most common names are Samuel L. Jackson, Robert De Niro, Morgan Freeman, Bruce Willis, and Nicolas Cage. The author also notes that the word cloud does not reflect the popularity or quality of the actors, but only their frequency of appearance.

Word Cloud for crew: Woody Allen, Clint Eastwood, and Martin Scorsese are some of the most prolific directors in the dataset. The word cloud also reveals some of the diversity of directors, such as Spike Lee, Ang Lee, Kathryn Bigelow, and Pedro Almodóvar. The word cloud is a useful way to visualize the distribution and variety of directors in the movies industry.

D)Content based recommendation system:

Text Data Transformation:

Employed CountVectorizer to convert textual data (genres, overview, cast, director) into a matrix representation suitable for analysis.

Created a matrix capturing the presence of words across movie features, forming a numerical representation for text.

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=5000,stop_words='english')
```

```
vector = cv.fit_transform(movies_df['content']).toarray()
```

```
vector.shape
```

```
(45536, 5000)
```

Cosine Similarity Computation:

Calculated cosine similarity between the text-based feature representations of movies.

Computed similarity scores indicating how closely movies' textual content aligned, determining their relatedness.

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
similarity = cosine_similarity(vector)
```

Recommendation Function Development:

Designed a recommendation function leveraging cosine similarity scores.

Generated movie recommendations based on similarity scores, suggesting movies similar to a user's input by identifying comparable textual content.

```
def recommend(movie):
    index = movies_df[movies_df['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])
    for i in distances[1:6]:
        print(movies_df.iloc[i[0]].title)
```

```
recommend('Spider-Man')
```

```
Spider-Man 2
Spider-Man 3
Emmauksen tiellä
Hoffa
The Amazing Spider-Man 2
```

Source code:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


mdf= pd.read_csv('movies_metadata.csv')

cdf=pd.read_csv('credits.csv')

mdf.head()

cdf.head()

mdf.info()


mdf_copy = mdf.copy()

mdf_copy['homepage'].fillna('Not Available', inplace=True)

mdf_copy.dropna(subset=['budget', 'revenue'], inplace=True)


# Converting 'budget' column to numeric without modifying the original DataFrame

mdf_copy['budget_numeric'] = pd.to_numeric(mdf_copy['budget'], errors='coerce')


Q1 = mdf_copy['budget_numeric'].quantile(0.25)

Q3 = mdf_copy['budget_numeric'].quantile(0.75)

IQR = Q3 - Q1

mdf_copy.loc[(mdf_copy['budget_numeric'] < (Q1 - 1.5 * IQR)) | (mdf_copy['budget_numeric'] >
(Q3 + 1.5 * IQR)), 'budget_numeric'] = float('NaN')
```

```
# Text data exploration - Word cloud for 'overview'

text_overview = ' '.join(mdf_copy['overview'].fillna(''))

wordcloud_overview = WordCloud(width=800, height=400, max_words=100,
background_color='white').generate(text_overview)

plt.figure(figsize=(10, 6))

plt.imshow(wordcloud_overview, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud for Overviews')

plt.show()

# Statistical analysis - Correlation matrix

numeric_cols = mdf_copy.select_dtypes(include=['float64']).columns.tolist()

correlation_matrix = mdf_copy[numeric_cols].corr()

plt.figure(figsize=(10, 8))

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')

plt.title('Correlation Matrix of Numeric Features')

plt.show()

# Feature Engineering - Extracting release year from 'release_date'

mdf_copy['release_year'] = pd.to_datetime(mdf_copy['release_date']).dt.year

# Calculate release year counts

release_year_counts = mdf_copy['release_year'].value_counts().sort_index()

# Plotting release year counts using a scatter plot

plt.figure(figsize=(9,5))

plt.scatter(release_year_counts.index, release_year_counts.values, color='skyblue', alpha=0.8)

plt.plot(release_year_counts.index, release_year_counts.values, color='orange', linestyle='--',
alpha=0.6)

plt.title('Distribution of Release Years')

plt.xlabel('Release Year')
```

```
plt.ylabel('Count')

plt.xticks(range(int(min(release_year_counts.index)), int(max(release_year_counts.index)) + 1, 5), rotation=45)

plt.grid(True)

plt.tight_layout()

plt.show()


mdf['id'] = mdf['id'].astype(str)

cdf['id'] = cdf['id'].astype(str)


# Assuming 'mdf' is the movies dataframe and 'cdf' is the credits dataframe

merged_df = pd.merge(mdf, cdf, on='id', how='left')

merged_df.head()

merged_df['cast'].iloc[1]

merged_df['cast'] = merged_df['cast'].apply(lambda x: [actor['name'] for actor in eval(x)] if pd.notnull(x) else [])

merged_df.head()

merged_df['crew'].iloc[1]

def extract_director(crew_list):

    director = [member['name'] for member in crew_list if member['job'] == 'Director']

    return director[0] if director else None


# Apply the function to extract director names

merged_df['director'] = merged_df['crew'].apply(lambda x: extract_director(eval(x)) if pd.notnull(x) else None)

merged_df.head()

merged_df['genres'].iloc[1]
```

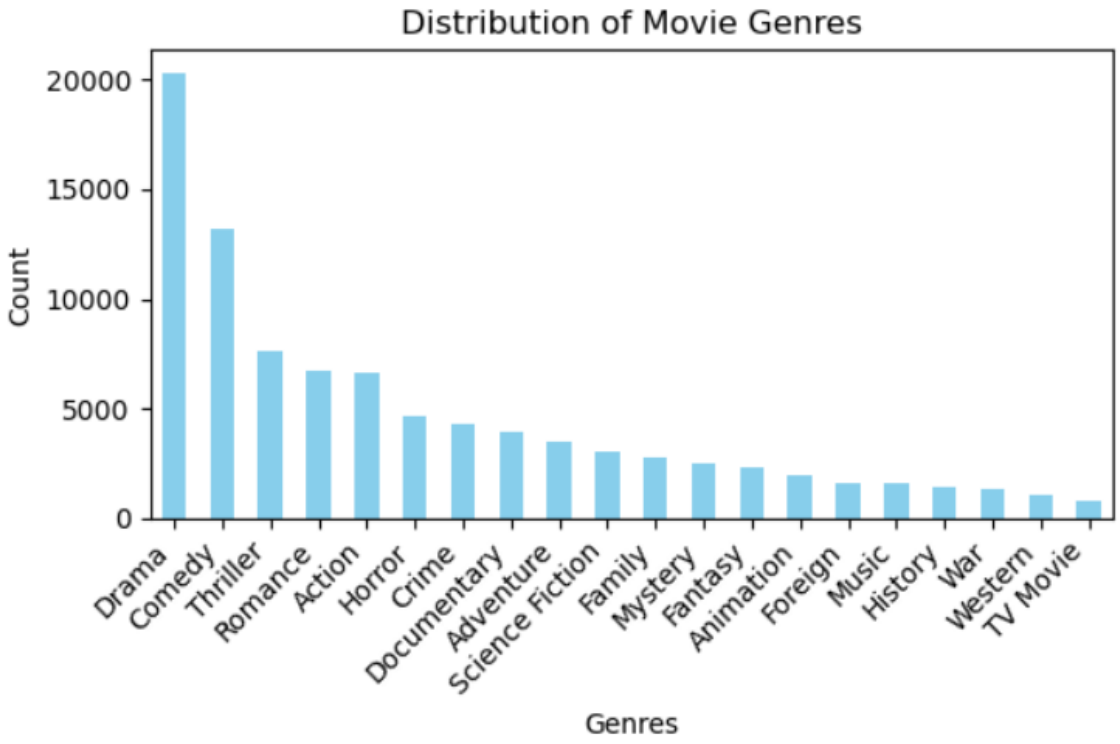
```
def extract_genres(genres_list):
    genres = [genre['name'] for genre in genres_list]
    return genres
merged_df['genres'] = merged_df['genres'].apply(lambda x: extract_genres(eval(x)) if
pd.notnull(x) else [])
merged_df.head()
#required columns aka features for recommendations
#['id','title','genres','overview','cast','director']
movies_df=merged_df[['id','title','genres','overview','cast','director']].copy()
movies_df.head()
movies_df.isnull().sum()
movies_df.dropna(subset=['title'], inplace=True)
movies_df.loc[:, 'overview'].fillna('', inplace=True)
movies_df.loc[:, 'director'].fillna('', inplace=True)
movies_df.isnull().sum()
movies_df['content'] = (
    movies_df['genres'].apply(lambda x: ' '.join(map(str, x))) + ' ' +
    movies_df['overview'] + ' ' +
    movies_df['cast'].apply(lambda x: ' '.join(map(str, x))) + ' ' +
    movies_df['director'])
movies_df
movies_df["content"] = movies_df["content"].fillna("")
movies_df.isnull().sum()
movies_df.info()
# Splitting genres into individual categories
genres_count = movies_df['genres'].explode().value_counts()
```

```
# Plotting the distribution of genres
plt.figure(figsize=(6, 4))
genres_count.plot(kind='bar', color='skyblue')
plt.title('Distribution of Movie Genres')
plt.xlabel('Genres')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
# Calculate overview length and create a histogram
movies_df['overview_length'] = movies_df['overview'].apply(lambda x: len(x.split()))
plt.figure(figsize=(7, 4))
plt.hist(movies_df['overview_length'], bins=50, color='salmon')
plt.title('Distribution of Overview Length')
plt.xlabel('Overview Length')
plt.ylabel('Frequency')
plt.show()
from wordcloud import WordCloud
# Handling missing values and converting to strings
movies_df['cast'] = movies_df['cast'].fillna('').astype(str)
# Word cloud for cast
text_cast = ' '.join(movies_df['cast'])
wordcloud_cast = WordCloud(width=800, height=400, max_words=100,
background_color='white').generate(text_cast)
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud_cast, interpolation='bilinear')
```

```
plt.axis('off')

plt.title('Word Cloud for Cast')

plt.show()

# Handling missing values and converting to strings

movies_df['director'] = movies_df['director'].fillna('').astype(str)

# Word cloud for directors

text_directors = ' '.join(movies_df['director'])

wordcloud_directors = WordCloud(width=800, height=400, max_words=100,
background_color='white').generate(text_directors)

plt.figure(figsize=(10, 6))

plt.imshow(wordcloud_directors, interpolation='bilinear')

plt.axis('off')

plt.title('Word Cloud for Directors')

plt.show()

from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=5000,stop_words='english')

vector = cv.fit_transform(movies_df['content']).toarray()

vector.shape

from sklearn.metrics.pairwise import cosine_similarity

similarity = cosine_similarity(vector)

def recommend(movie):

    index = movies_df[movies_df['title'] == movie].index[0]

    distances = sorted(list(enumerate(similarity[index])),reverse=True,key = lambda x: x[1])

    for i in distances[1:6]:

        print(movies_df.iloc[i[0]].title)

recommend('Spider-Man')
```

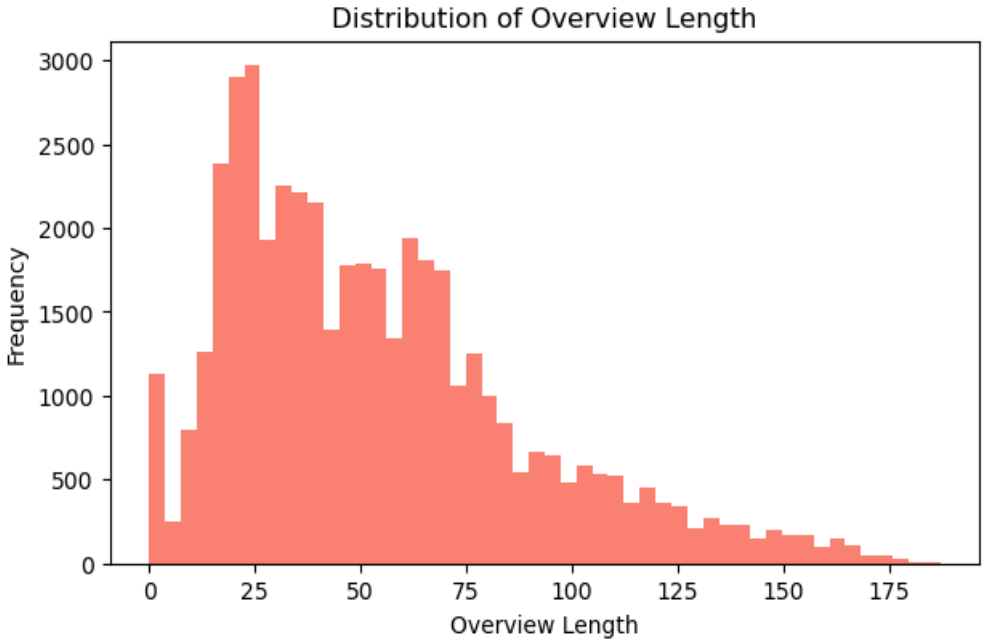
Outputs and graphs:

Distribution of movies genres:

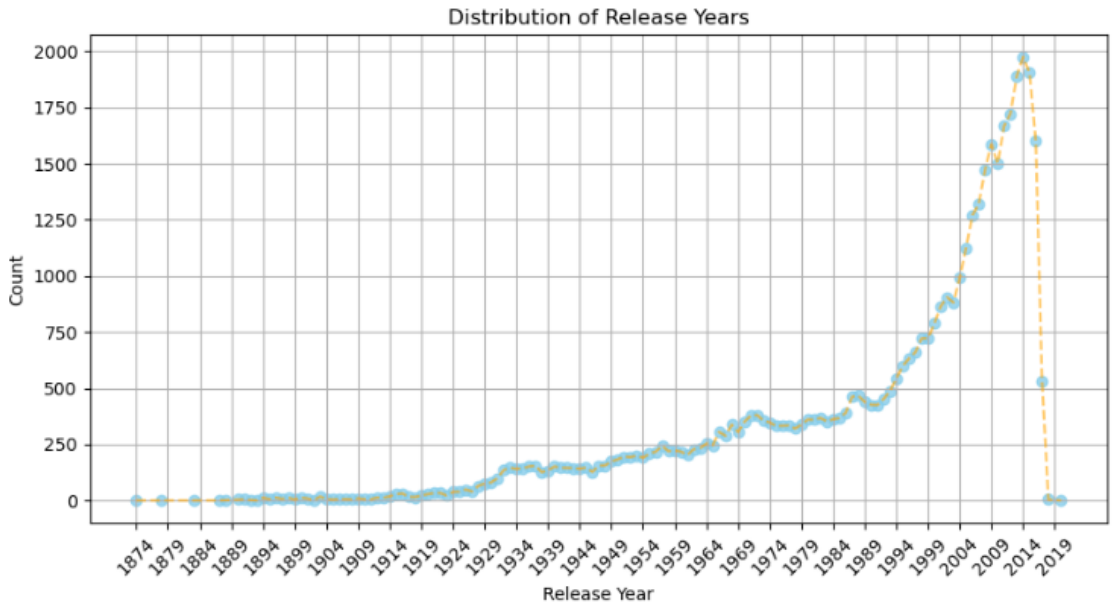


Word cloud for overviews:

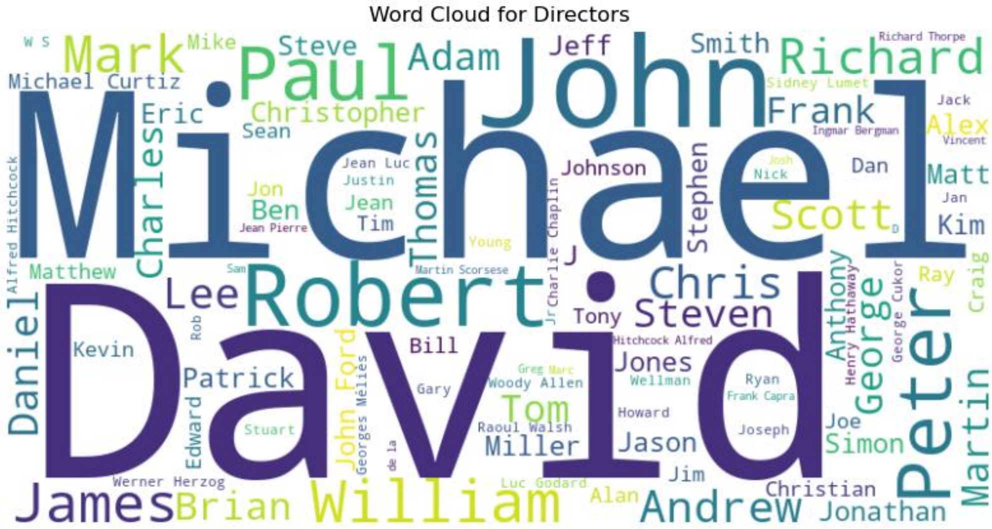
Distribution of the overview length:



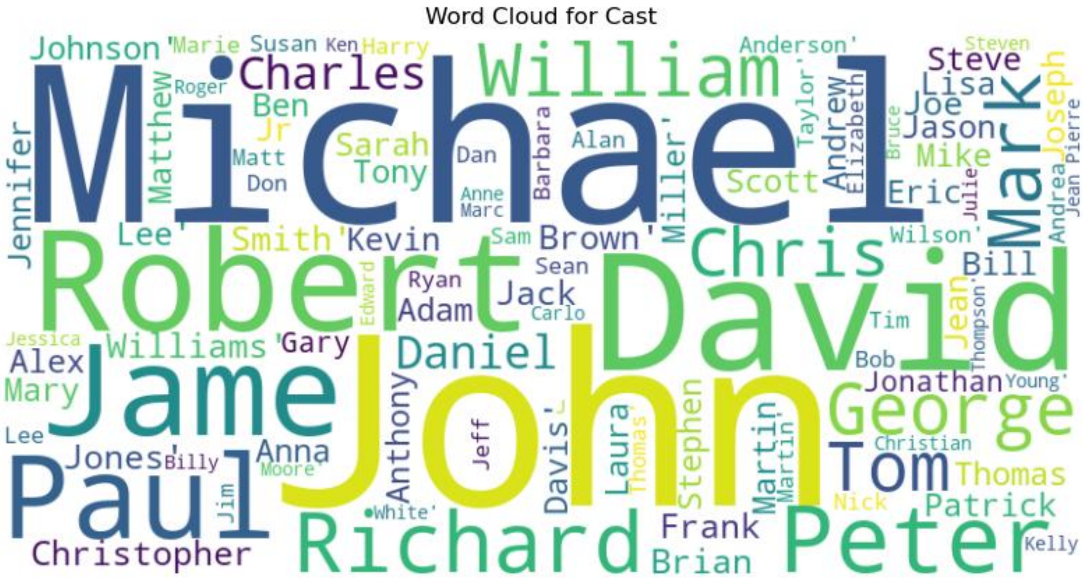
Distribution of release years:



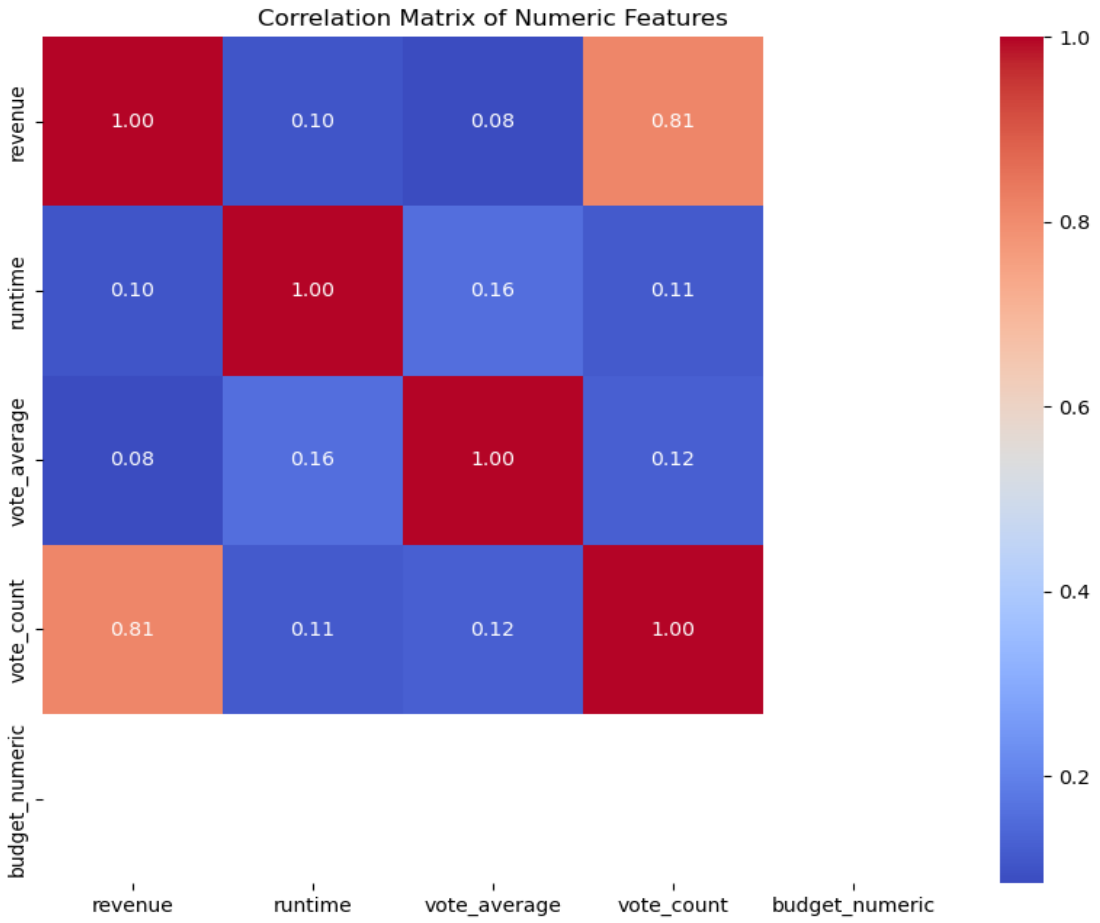
Word cloud for directors:



Word cloud for cast:



Correlation matrix of Numeric features:



Final ouput of recommending movie:

```
recommend('Spider-Man')
```

```
Spider-Man 2
Spider-Man 3
Emmauksen tiellä
Hoffa
The Amazing Spider-Man 2
```

Conclusion:

The Movie Recommendation System project achieved significant milestones by meticulously processing data, implementing an effective recommendation system, and exploring various avenues for enhancing user engagement. Data preprocessing ensured data integrity by handling missing values and extracting essential movie features, forming the groundwork for a comprehensive dataset crucial for recommendation generation. The recommendation system, leveraging text transformation techniques and cosine similarity, successfully facilitated precise movie suggestions aligned with user preferences based on content similarity.

However, the project also encountered limitations that provide opportunities for future improvements:

Scalability Challenges: The current system might face constraints in scalability when dealing with larger datasets or increased user traffic. Optimal algorithms and infrastructure enhancements are necessary to handle growing data volumes.

Diversification in Recommendations: Solely relying on content-based approaches might limit the diversity of suggestions. Integration of collaborative filtering or hybrid methods could offer more diverse and accurate recommendations by considering user behavior and preferences.

Data Enrichment: The system's recommendations primarily rely on textual data, potentially missing nuanced movie characteristics. Incorporating additional data sources like user ratings, reviews, or contextual information could refine recommendation accuracy and comprehensiveness.

In conclusion, while the project achieved significant milestones in data processing and recommendation system implementation, addressing these limitations through scalability improvements, diversification in recommendation techniques, and enriching data sources would be pivotal for a more robust and user-centric movie recommendation system.

Reference:

https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset

https://medium.com/web-mining-is688-spring-2021/content-based-movie-recommendation-system-72f122641eab

https://github.com/campusx-official/movie-recommender-system-tmdb-dataset/blob/main/notebook86c26b4f17.ipynb