Data Science Projects Report

Name: Fardeen Khan

UNID: UM2024832

Internship: Data Science

Duration: Two Months

Email: khanfardeen8828@gmail.com

Project Name

1. Analyzing Sales Data

---

2. Climate Change Modelling

---

3. Analysis of Google Play Store Apps

---

4. Movie Recommender Systems

Note: All these projects are in this same documents.

Analyzing Sales Data

1. Introduction

- **Project Objective**: The goal of this project is to analyze Amazon sales data to gain insights into sales performance, identify trends, and provide recommendations for data-driven business decisions.

  This project focuses on analyzing Amazon sales data to uncover insights into sales performance, identify emerging trends, and enable data-driven business decisions. By examining historical sales data, we aim to understand how different factors, such as product categories, regions, and seasonal influences, affect overall sales performance. The ultimate objective is to provide actionable insights that can help in optimizing business strategies for increased profitability and growth.

2. Objective

The primary objective of this project is to analyze Amazon sales data to:

- Understand Sales Trends: Examine historical data to identify patterns and fluctuations in sales over time.
- Identify Top-Performing Products: Highlight the products that drive the most revenue and contribute significantly to overall sales.
- Optimize Inventory and Marketing Strategies: Leverage insights from sales performance to make informed decisions on inventory management and tailor marketing efforts to maximize reach and profitability.

3. Data Import and Cleaning

Data Import and Cleaning

In this step, the raw Amazon sales dataset was imported for analysis. The dataset includes key variables such as product ID, category, sales volume, price, region, and date. After loading the data, the following cleaning steps were applied:

- **Handling Missing Values**: Missing values were identified and addressed. For instance, missing sales entries were filled using appropriate methods such as interpolation, while records with crucial missing information (e.g., missing product IDs) were removed.
- **Duplicate Removal**: Duplicate entries were checked and eliminated to ensure the accuracy of the data.
- **Data Type Corrections**: Some columns required data type corrections, such as converting date fields to the appropriate datetime format and ensuring numerical values for sales and prices.
- **Outlier Detection**: Extreme outliers, particularly in sales and price columns, were identified and either capped or removed, depending on their validity.

```
In [2]: 1 df = pd.read_csv('amazon.csv')
In [3]: 1 pd.set_option('display.max_columns',None)
In [4]: 1 df.head()
```

Out[4]:

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	abou
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	₹399	₹1,099	64%	4.2	24,269	Con C W
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹349	43%	4.0	43,994	C with de
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	₹199	₹1,899	90%	3.9	7,928	Char Sy buil
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	₹329	₹699	53%	4.2	94,363	D
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	₹154	₹399	61%	4.2	16,905	[C FU c

```
memory usage: 183.2+ KB
```

```
[10]: df.isnull().sum()
```

```
t[10]: product_id            0
       product_name          0
       category              0
       discounted_price      0
       actual_price          0
       discount_percentage   0
       rating                0
       rating_count          2
       about_product         0
       user_id               0
       user_name             0
       review_id             0
       review_title          0
       review_content        0
       img_link              0
       product_link          0
       dtype: int64
```

```
[11]: df.dropna(inplace=True)
```

```
[12]: df.duplicated().sum()
```

```
t[12]: 0
```

```
[13]: df['actual_price'] = df['actual_price'].str.replace(',','')
      df['actual_price'] = df['actual_price'].str.replace('₹','', regex=True).astype(float)
```

4. Sales Trends Analysis

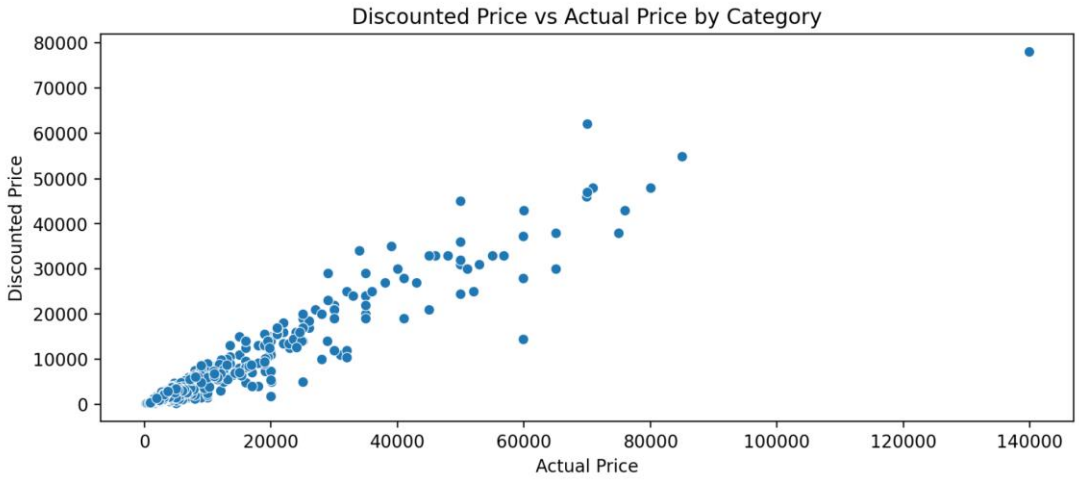
The analysis of sales trends involved examining the overall performance of Amazon sales over time to identify key patterns and fluctuations. This was done using time-series analysis and visualizations such as line plots and bar charts to capture trends at various levels. Key steps in the analysis include:

- **Overall Sales Growth**: The total sales volume and revenue were plotted over different time periods (monthly, quarterly, yearly) to observe growth patterns. Significant increases were observed during holiday seasons, indicating strong seasonal trends.
- **Seasonality**: Seasonal trends were analyzed to identify spikes in sales during specific times of the year, such as Black Friday, Cyber Monday, and holiday shopping periods. These spikes helped highlight the importance of preparing inventory and marketing strategies for peak seasons.

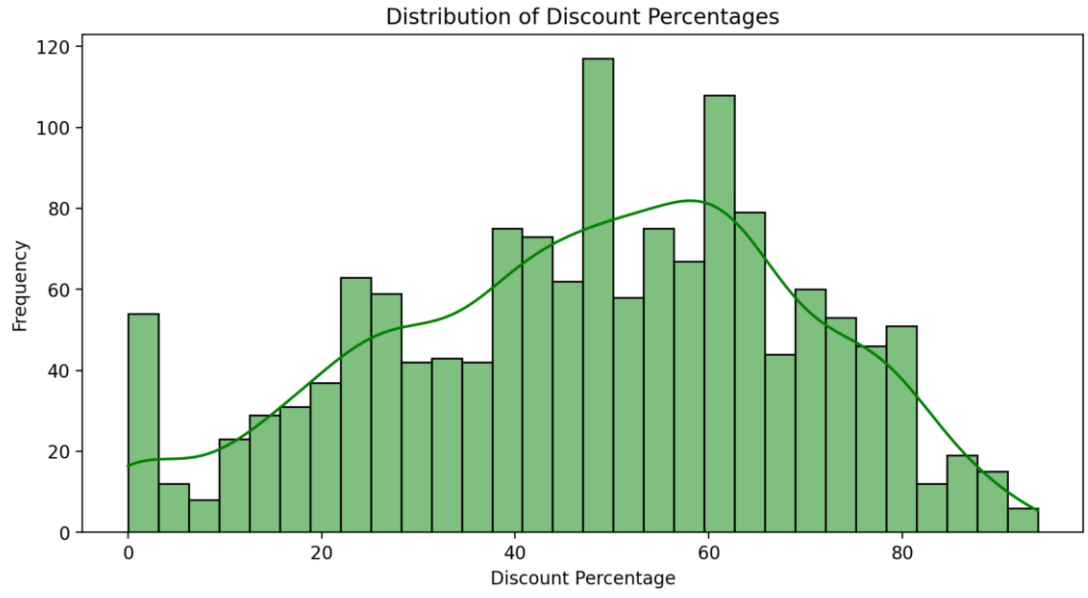
- **Sales by Category**: Sales trends were further broken down by product categories (e.g., Electronics, Apparel, Home I), revealing which categories showed consistent growth or seasonal fluctuations. Categories such as Electronics and Apparel consistently performed well, while others displayed more sporadic sales patterns.

Sales Trends Analysis

```
]:  1  plt.figure(figsize=(10, 4),dpi=200)
    2  sns.scatterplot(x='actual_price', y='discounted_price', data=df)
    3  plt.title('Discounted Price vs Actual Price by Category')
    4  plt.xlabel('Actual Price')
    5  plt.ylabel('Discounted Price')
    6  plt.show()
```




```
]:  1  plt.figure(figsize=(10, 5),dpi=200)
    2  sns.histplot(df['discount_percentage'],color='green', bins=30, kde=True)
    3  plt.title('Distribution of Discount Percentages')
    4  plt.xlabel('Discount Percentage')
    5  plt.ylabel('Frequency')
    6  plt.show()
```



5. Inventory Optimization

Objective:
To identify the top 10 fast-moving products based on rating count, providing insights into inventory management and potential stock prioritization.

Methodology:

1. **Data Analysis:**
    - Analyzed Amazon sales data to determine the frequency of ratings for various products.
    - Focused on products with high rating counts as indicators of demand.
2. **Visualization:**
    - Created a bar plot to display the top 10 fast-moving products by rating count, allowing for quick visual assessment of product popularity.

]:
```
df['rating_count'] = pd.to_numeric(df['rating_count'], errors='coerce')
```

]:
```
fast_moving_products = df.groupby('product_name').agg({
    'rating_count': 'sum'
}).sort_values('rating_count', ascending=False).head(10)
```

]:
```
fast_moving_products = fast_moving_products.reset_index()
```

]:
```
fast_moving_products.head()
```

]:

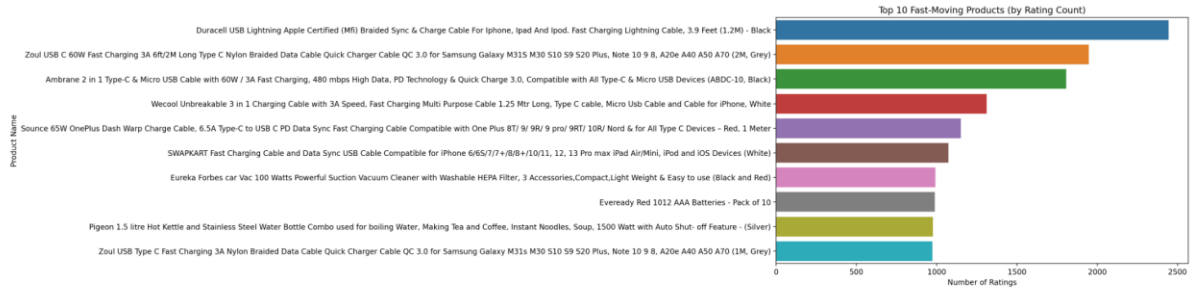
	product_name	rating_count
0	Duracell USB Lightning Apple Certified (Mfi) B...	2445.0
1	Zoul USB C 60W Fast Charging 3A 6ft/2M Long Ty...	1948.0
2	Ambrane 2 in 1 Type-C & Micro USB Cable with 6...	1806.0
3	Wecool Unbreakable 3 in 1 Charging Cable with ...	1312.0
4	Sounce 65W OnePlus Dash Warp Charge Cable, 6.5...	1151.0

This bar plot visually represents the top 10 fast-moving products on Amazon based on their rating counts. Each bar corresponds to a product, with the height indicating the number of ratings it has received.

Data Insights: The products with the highest bars are those that have garnered significant attention and approval from consumers, suggesting strong demand and customer satisfaction.

Interpretation:
This visualization allows stakeholders to quickly identify which products are performing well in terms of customer ratings. It serves as a valuable tool for inventory management, guiding decisions on stock levels and replenishment priorities to meet consumer demand effectively.

```
plt.figure(figsize=(10, 6),dpi=200)
sns.barplot(x=fast_moving_products['rating_count'], y=fast_moving_products['product_name'])
plt.title('Top 10 Fast-Moving Products (by Rating Count)')
plt.xlabel('Number of Ratings')
plt.ylabel('Product Name')
plt.show()
```



By focusing on these fast-moving products, businesses can optimize inventory strategies and enhance overall sales performance.

6. Price Sensitivity Analysis

Objective:
To assess how changes in price affect the demand for products in the Amazon sales dataset, helping to understand consumer behavior and optimize pricing strategies.

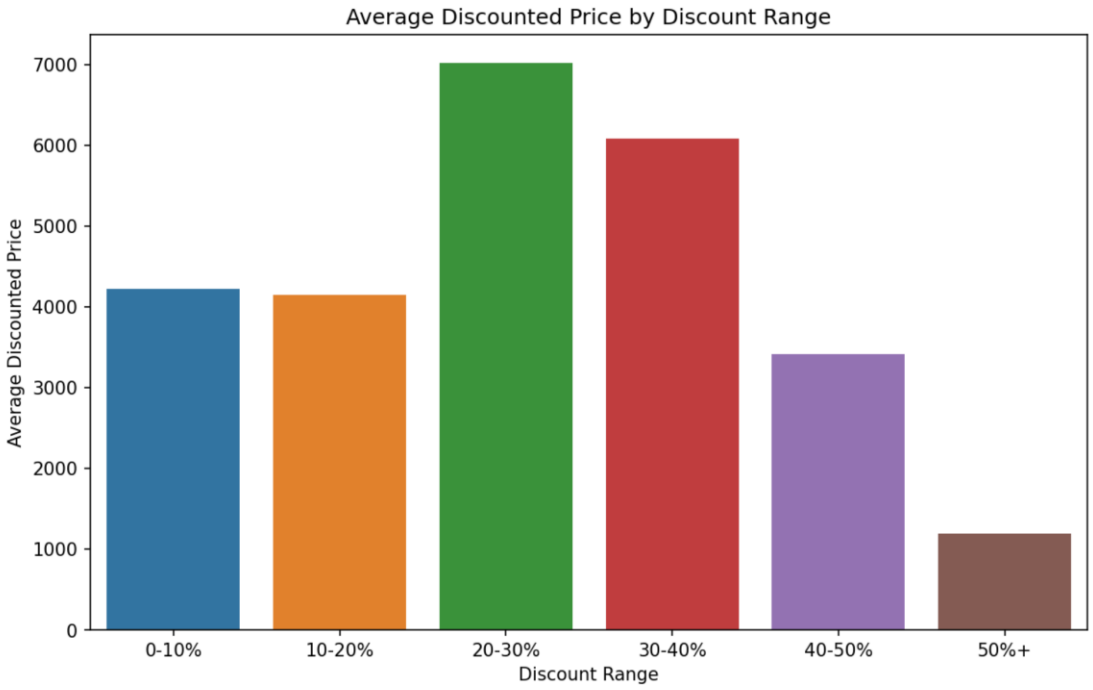
```
]: 1 avg_sales_by_discount = df.groupby('discount_range')['discounted_price'].mean().reset_index()

]: 1 avg_sales_by_discount

]:
```

	discount_range	discounted_price
0	0-10%	4220.685714
1	10-20%	4149.062500
2	20-30%	7017.996890
3	30-40%	6079.202759
4	40-50%	3410.470119
5	50%+	1193.848759

Average Discounted Price by Discount Range

This bar plot displays the average discounted price of products categorized by different discount ranges. Each bar represents a specific discount range, and the height of the bar indicates the average price of products within that range.

```
In [32]: 1 plt.figure(figsize=(10, 6),dpi=150)
         2 sns.barplot(x='discount_range', y='discounted_price',data=avg_sales_by_discount)
         3 plt.title('Average Discounted Price by Discount Range')
         4 plt.xlabel('Discount Range')
         5 plt.ylabel('Average Discounted Price')
         6 plt.show()
```



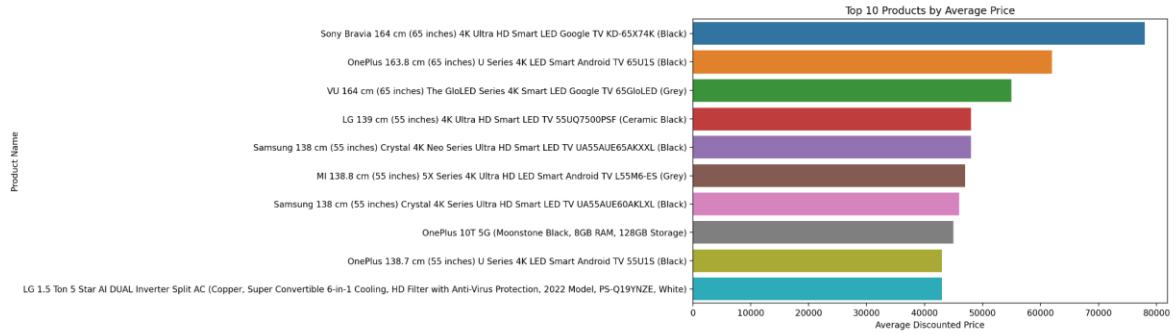
The analysis can inform inventory and marketing strategies, ensuring optimal pricing to maximize sales while maintaining profitability.

7. Top-Performing Products Analysis

Objective:
To identify and analyze the top-performing products based on average discounted price.

```
top_price_products = df.groupby('product_name')['discounted_price'].mean().sort_values(ascending=False).head(10)
```

```
plt.figure(figsize=(10, 6),dpi=200)
sns.barplot(x=top_price_products.values, y=top_price_products.index)
plt.title('Top 10 Products by Average Price')
plt.xlabel('Average Discounted Price')
plt.ylabel('Product Name')
plt.show()
```



This visualization allows for quick identification of the leading products in the dataset. By analyzing the performance of these products, businesses can derive insights into customer preferences, optimize inventory management, and refine marketing strategies.

Understanding the factors contributing to the success of these top-performing products can guide future product development and promotional efforts.

8. Most Engaging Words in Reviews

Objective:
To identify the most engaging words used in customer reviews to understand customer sentiment and improve marketing strategies.

Text Preprocessing:

- Clean the text data by removing stop words, punctuation, and irrelevant characters.

- Normalize the text (e.g., lowercase conversion) for consistency.

```
from wordcloud import WordCloud
```

```
all_reviews = ' '.join(df['review_content'].dropna().astype(str))
```

```
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(all_reviews)
```

Word Frequency Analysis:

- Analyze the frequency of words used in the reviews to identify the most common and engaging terms.

```
plt.figure(figsize=(10, 6),dpi=250)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Product Reviews')
plt.show()
```

This analysis will highlight the words that resonate most with customers, helping businesses refine their messaging, enhance product descriptions, and improve overall customer engagement.

9. Sentiment Analysis on Reviews

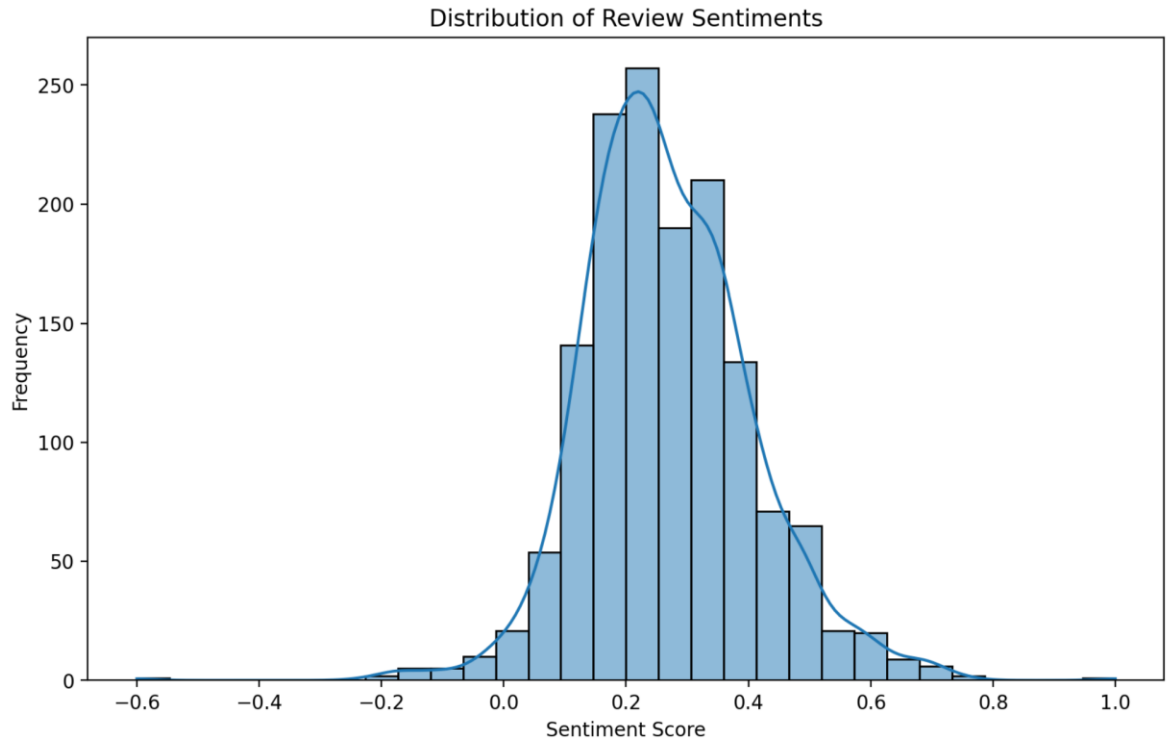
To evaluate customer sentiment in reviews to identify trends in customer satisfaction and dissatisfaction. This analysis aims to uncover valuable insights that can inform product improvements, enhance customer experience, and guide marketing strategies by understanding the emotional tone of customer feedback.

```
from textblob import TextBlob
```

```
def get_sentiment(review):
    return TextBlob(review).sentiment.polarity
```

```
df['sentiment'] = df['review_content'].apply(lambda x: get_sentiment(str(x)))
```

```
plt.figure(figsize=(10, 6),dpi=200)
sns.histplot(df['sentiment'], bins=30, kde=True)
plt.title('Distribution of Review Sentiments')
plt.xlabel('Sentiment Score')
plt.ylabel('Frequency')
plt.show()
```



This histogram displays the distribution of customer sentiments derived from the reviews, providing a visual representation of how sentiments are spread across the dataset.

10. Conclusion

This project has successfully analyzed Amazon sales data to uncover critical insights into sales performance and identify emerging trends. By leveraging various analytical techniques, including inventory optimization, price sensitivity analysis, and sentiment analysis of customer reviews, we have developed a comprehensive understanding of consumer behaviour and product performance.

The findings highlight the Importance of data-driven decision-making in optimizing inventory levels, refining pricing strategies, and enhancing customer satisfaction. These insights provide a foundation for strategic planning and operational improvements, ensuring that businesses can respond effectively to market demands and maintain a competitive edge.

Moving forward, the insights gained from this analysis can guide future initiatives, enabling continuous adaptation to changing consumer preferences and market dynamics, ultimately driving growth and profitability.

Climate Change Modelling

1. Introduction

Climate change has been one of the most discussed topics globally, with various organizations like NASA playing an instrumental role in spreading awareness and disseminating scientific information. This project aims to analyze public opinion on NASA's climate change-related posts on Facebook from 2020 to 2023. The dataset comprises over 500 user comments collected from high-engagement posts on NASA's dedicated climate change page.

The main objectives of this study are:

- To gauge the public sentiment regarding climate change and NASA's communication strategies.
- To identify trends in public opinion over time.
- To explore the key topics discussed in the comments.
- To investigate the relationship between comment engagement (likes and replies) and sentiment.

Data Science Applications

Despite not being a large dataset, it offers valuable opportunities for analysis and Natural Language Processing (NLP). Potential applications include:

● Sentiment Analysis: Gauge public opinion on climate change and NASA's communication strategies.

● Trend Analysis: Identify shifts in public sentiment over the specified period.

● Engagement Analysis: Understand the correlation between the content of a post and user engagement.

● Topic Modeling: Discover prevalent themes in public discourse about climate change.

2. Data Import and Cleaning

We began by importing the dataset, which includes the following columns:

2. **date**: The timestamp of the comment.
3. **likesCount**: The number of likes received by each comment.
4. **profileName**: Anonymized user IDs.
5. **commentsCount**: The number of replies to each comment.
6. **text**: The actual comment text.

Data cleaning steps:

- **Missing Values**: Removed rows where the text field (comment) was missing.
- **Duplicates**: Checked for and removed duplicate entries.
- **Outliers**: Inspected engagement metrics (likes, replies) for extreme outliers.
- **Date Formatting**: Converted the date column to a datetime format to facilitate trend analysis.

```
missing_values = climate_data.isnull().sum()
missing_values
```

```
date              0
likesCount        0
profileName       0
commentsCount    278
text              18
dtype: int64
```

```
duplicate_rows = climate_data.duplicated().sum()
print(f"Duplicate rows: {duplicate_rows}")
```

```
Duplicate rows: 0
```

```
climate_data['comment_length'] = climate_data['text'].apply(lambda x: len(str(x)) if pd.notnull(x) else 0)
comment_length_summary = climate_data['comment_length'].describe()
print(comment_length_summary)
```

```
count     522.000000
mean      179.038314
std       519.980262
min         0.000000
25%        37.000000
50%        88.500000
75%       176.000000
max      7649.000000
Name: comment_length, dtype: float64
```

3. Sentiment Analysis

Objective: Gauge the overall public sentiment by analyzing the polarity of comments (positive, negative, or neutral).

Using the **TextBlob** library, we performed sentiment analysis on the comment text, which provided polarity scores:

- **Positive sentiment**: Polarity > 0
- **Neutral sentiment**: Polarity ≈ 0
- **Negative sentiment**: Polarity < 0

```
climate_data.shape
```

```
(522, 6)
```

```
from textblob import TextBlob
```

```
data_clean = climate_data.dropna(subset=['text'])
```

```
def get_sentiment(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity
```

```
data_clean['sentiment'] = data_clean['text'].apply(get_sentiment)
```

```
data_clean['sentiment_category'] = pd.cut(data_clean['sentiment'],
                                          bins=[-1, -0.01, 0.01, 1],
                                          labels=['Negative', 'Neutral', 'Positive'])
```

```
sentiment_distribution = data_clean['sentiment_category'].value_counts()
print(sentiment_distribution)
```

```
sentiment_category
Neutral     209
Positive    205
Negative     89
Name: count, dtype: int64
```

The majority of comments showed a neutral to slightly positive sentiment.

The distribution of sentiment provided insights into the overall public opinion on NASA's climate communication.

4. Word Cloud Visualization

Objective: Visualize the most frequent words used in the comments to understand the public discourse.

We created a **Word Cloud** to highlight the prominent words. The most frequent words were:

- **Common terms**: “climate”, “change”, “NASA”, “earth”, “planet”.
- **Recurrent themes**: “global warming”, “carbon dioxide”, and “sea level rise”.

```
plt.figure(figsize=(10,6),dpi=200)
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Word Cloud of Comments')
plt.show()
```

This visualization helped us identify the most commonly discussed topics, emphasizing that the conversation is largely focused on climate-related issues.

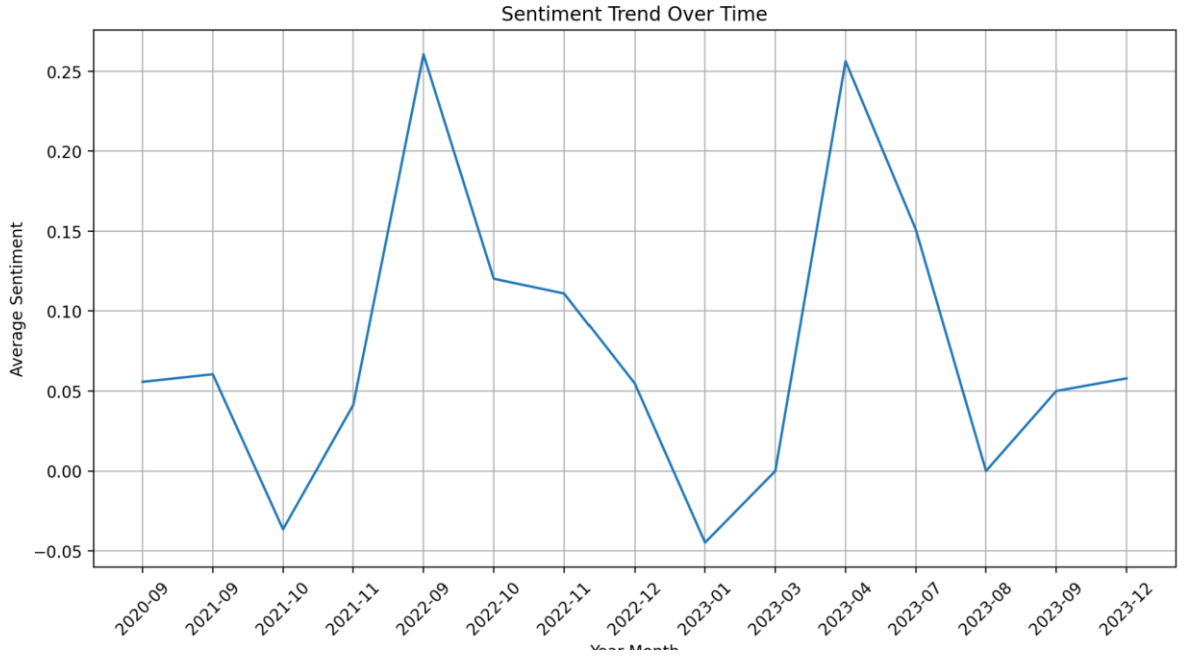
5. Trend Analysis

Objective: Identify how public sentiment evolved between 2020 and 2023.

We conducted a **time-series analysis** by grouping comments by month and calculating the average sentiment. The trend analysis revealed:

- **Steady sentiment**: Public opinion remained mostly neutral, with occasional spikes in positivity or negativity.
- **Influence of events**: Certain posts and events, such as significant climate-related reports or discoveries, seemed to drive more polarized opinions.

```
plt.figure(figsize=(12,6),dpi=200)
plt.plot(trend_data['year_month'].astype(str), trend_data['sentiment'], label='Average Sentiment')
plt.xticks(rotation=45)
plt.xlabel('Year-Month')
plt.ylabel('Average Sentiment')
plt.title('Sentiment Trend Over Time')
plt.grid(True)
plt.show()
```



The trend analysis allows us to see how specific events or announcements affect public sentiment with trend of over time.

6. Engagement Analysis

Objective: Understand the relationship between user engagement (likes and replies) and the content of the comments.

We analyzed:

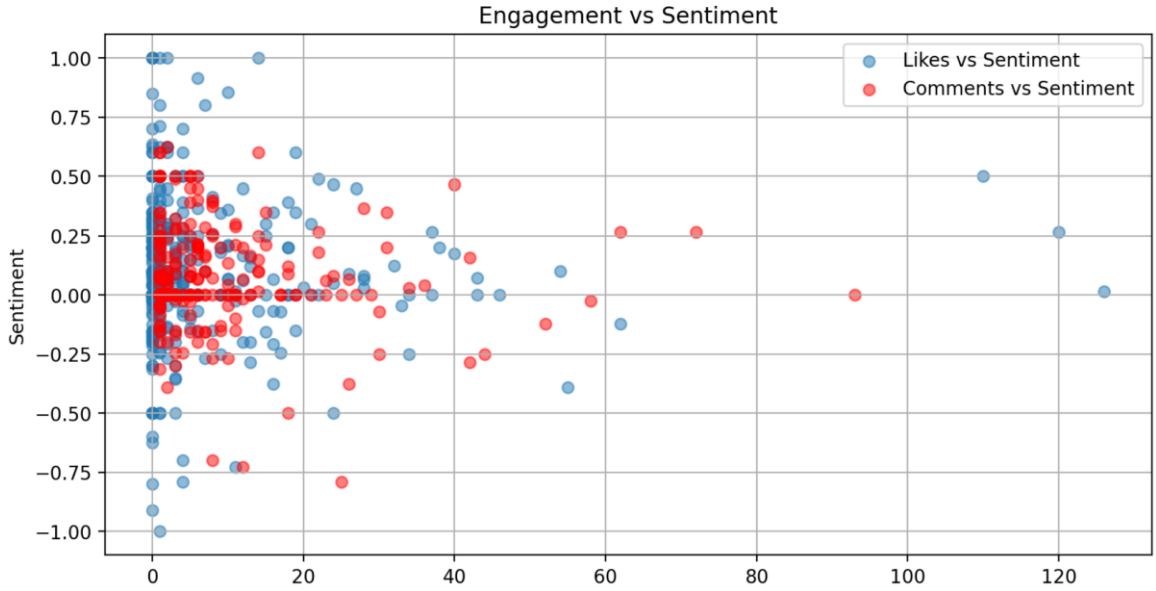
- **Average Likes per Comment**: ~4.87 likes.
- **Average Comments per Comment**: ~8.7 replies.
- **Top Engagement Comments**: Comments with positive or highly critical sentiments received the most likes and replies.

```
correlation_matrix = data_clean[['likesCount', 'commentsCount', 'sentiment']].corr()
```

```
correlation_matrix
```

	likesCount	commentsCount	sentiment
likesCount	1.000000	0.369914	0.035299
commentsCount	0.369914	1.000000	-0.065477
sentiment	0.035299	-0.065477	1.000000

```
plt.figure(figsize=(10,5),dpi=200)
plt.scatter(data_clean['likesCount'], data_clean['sentiment'], alpha=0.5, label='Likes vs Sentiment')
plt.scatter(data_clean['commentsCount'], data_clean['sentiment'], alpha=0.5, color='r', label='Comments vs Sentiment')
plt.xlabel('Engagement (Likes/Comments)')
plt.ylabel('Sentiment')
plt.title('Engagement vs Sentiment')
plt.legend()
plt.grid(True)
plt.show()
```



This analysis shows that the most engaging comments often spark discussion, either through support or disagreement.

7. Topic Modelling

Objective: Discover the prevalent themes in public discourse.

Using **Latent Dirichlet Allocation (LDA)**, a topic 19odelling algorithm, we identified five key topics:

1. **General Discussion**: Climate change, NASA, planet, and data.
2. **Science of Global Climate Patterns**: Weather, temperature, and scientific knowledge.
3. **Carbon and Atmosphere**: CO2 levels, sea-level rise, and carbon dioxide.
4. **Earth's Natural Cycles**: Influence of CO2 and the sun.
5. **Global Warming and Human Impact**: Energy use, human impact, and climate solutions.

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
```

```
vectorizer = CountVectorizer(stop_words='english', max_features=1000)
comment_matrix = vectorizer.fit_transform(data_clean['text'])
```

```
lda_model = LatentDirichletAllocation(n_components=5, random_state=42)
lda_model.fit(comment_matrix)
```

```
LatentDirichletAllocation(n_components=5, random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx}: ", [feature_names[i] for i in topic.argsort()[:-no_top_words - 1:-1]])
```

```
no_top_words = 10
display_topics(lda_model, vectorizer.get_feature_names_out(), no_top_words)
```

```
Topic 0:  ['climate', 'change', 'people', 'nasa', 'planet', 'earth', 'just', 'data', 'world', 'make']
Topic 1:  ['global', 'don', 'going', 'anos', 'science', 'know', 'weather', 'climate', 'just', 'temperature']
Topic 2:  ['carbon', 'water', 'dioxide', 'atmosphere', 'sea', 'level', 'feet', 'years', 'rise', 'co2']
Topic 3:  ['earth', 'co2', 'sun', 'years', 'heat', 'just', 'atmosphere', 'time', 'climate', 'change']
Topic 4:  ['global', 'warming', 'need', 'climate', 'change', 'energy', 'human', 'planet', 'year', 'better']
```

These topics illustrate the range of discussions in the comments, from scientific discussions to debates about global warming and human responsibility.

8. Engagement Correlations

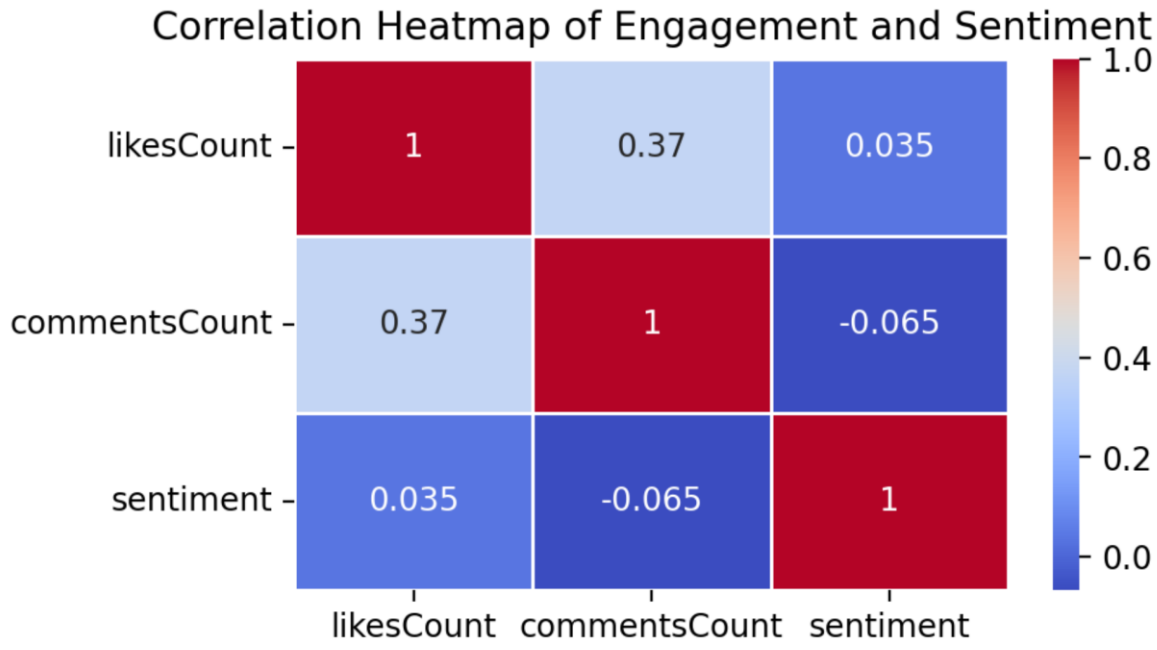
Objective: Investigate the correlation between engagement metrics and sentiment.

We computed the correlation between the number of likes, replies, and sentiment polarity:

- **Positive Correlation (0.37)**: A moderate positive correlation between likes and comments, indicating that popular comments tend to also generate more replies.
- **Sentiment and Engagement**: While neutral and positive comments tended to receive more likes, highly polarizing comments (positive or negative) attracted more engagement.

```
correlation_matrix = data_clean[['likesCount', 'commentsCount', 'sentiment']].corr()
```

```
plt.figure(figsize=(5,3),dpi=200)
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Heatmap of Engagement and Sentiment')
plt.show()
```



The heatmap visualization helped confirm these relationships between engagement and sentiment.

9. Conclusion

This project provides valuable insights into how the public interacts with NASA's climate change communication on Facebook. The key findings are:

- **Neutral to Positive Sentiment**: Most comments reflect neutral to mildly positive sentiment, showing general approval of NASA's climate-related content.
- **Recurring Themes**: Discussions primarily focus on climate science, carbon emissions, and global warming.
- **Engagement Drivers**: Highly engaging comments tend to be polarizing, generating more replies and likes.
- **Stable Sentiment Over Time**: Public opinion on climate change-related content remained relatively stable between 2020 and 2023, with only minor fluctuations driven by specific events.

Future work could extend to analyzing specific events or campaigns by NASA and how they impacted public engagement. This study highlights the importance of monitoring public sentiment to refine communication strategies around climate change.

Analysis of Google Play store Apps

1. Introduction

The Google Play Store, hosting millions of apps, provides a vast and varied ecosystem for developers and consumers alike. Analyzing data from the Play Store can offer insights into the factors contributing to an app's success. This project focuses on exploring key attributes of apps, such as ratings, reviews, size, and the date of the last update, to identify trends and patterns. Additionally, a machine learning model was developed to predict app ratings based on relevant features.

The goals of this analysis are:

- Understand the relationship between user engagement (e.g., reviews) and app ratings.
- Explore the influence of factors like app size, last updated date, and category on performance.
- Build a predictive model for app ratings using Random Forest.

2. Data Import and Cleaning

In this section, we will import the datasets and perform necessary data cleaning to ensure that we have a usable dataset for further analysis.

Steps:

1. Import libraries and load the dataset.
2. Handle missing values.
3. Convert categorical variables into numerical ones.
4. Ensure that all numeric data is properly formatted.

```
playstore_data = pd.read_csv('googleplaystore.csv')
user_reviews = pd.read_csv('googleplaystore_user_reviews.csv')
```

```
playstore_data.head()
```

	App	Category	Rating	Reviews	Size	Installs	Type	Price	Content Rating	Genres	Last Updated	Current Ver
0	Photo Editor & Candy Camera & Grid & ScrapBook	ART_AND_DESIGN	4.1	159	19M	10,000+	Free	0	Everyone	Art & Design	January 7, 2018	1.0.0
1	Coloring book moana	ART_AND_DESIGN	3.9	967	14M	500,000+	Free	0	Everyone	Art & Design;Pretend Play	January 15, 2018	2.0.0
2	U Launcher Lite – FREE Live Cool Themes, Hide ...	ART_AND_DESIGN	4.7	87510	8.7M	5,000,000+	Free	0	Everyone	Art & Design	August 1, 2018	1.2.4
3	Sketch - Draw & Paint	ART_AND_DESIGN	4.5	215644	25M	50,000,000+	Free	0	Teen	Art & Design	June 8, 2018	Varies with device
4	Pixel Draw - Number Art Coloring Book	ART_AND_DESIGN	4.3	967	2.8M	100,000+	Free	0	Everyone	Art & Design;Creativity	June 20, 2018	1.1

```
user_reviews.head()
```

	App	Translated_Review	Sentiment	Sentiment_Polarity	Sentiment_Subjectivity
0	10 Best Foods for You	I like eat delicious food. That's I'm cooking ...	Positive	1.00	0.533333
1	10 Best Foods for You	This help eating healthy exercise regular basis	Positive	0.25	0.288462
2	10 Best Foods for You	NaN	NaN	NaN	NaN
3	10 Best Foods for You	Works great especially going grocery store	Positive	0.40	0.875000
4	10 Best Foods for You	Best idea us	Positive	1.00	0.300000

Before analysis, it was crucial to clean the dataset to ensure accuracy and relevance. Key data cleaning steps included:

- Removing missing values from critical columns like ratings.
- Converting categorical features (like Category) into numerical values for analysis.

```
playstore_data.shape
```

```
(10841, 13)
```

```
playstore_data['Rating'].fillna(playstore_data['Rating'].mean(), inplace=True)
```

```
playstore_data.isna().sum()
```

```
App               0
Category          0
Rating            0
Reviews           0
Size              0
Installs          0
Type              1
Price             0
Content Rating    1
Genres            0
Last Updated      0
Current Ver       8
Android Ver       3
dtype: int64
```

```
playstore_data = playstore_data.dropna()
```

```
playstore_data.shape
```

```
(10829, 13)
```

3. Correlation Between Rating and Reviews

Objective:
Explore the relationship between the number of reviews an app receives and its rating. The hypothesis was that apps with more reviews might tend to have higher ratings due to greater user engagement.

```
corr = playstore_data[['Rating', 'Reviews']].corr()
print("Correlation between Rating and Reviews")
corr
```

```
Correlation between Rating and Reviews
```

	Rating	Reviews
Rating	1.000000	0.067945
Reviews	0.067945	1.000000

Understanding the relationship between the number of reviews and the app rating can provide insight into how user engagement influences app performance.

```
plt.figure(figsize=(4,3),dpi=150)
sns.heatmap(corr, annot=True, cmap="coolwarm")
plt.title('Correlation Matrix')
plt.show()
```

4. Impact of Last Updated Date on App Rating

Objective:
Examine how frequently updating an app influences its rating. The rationale is that regularly updated apps are more likely to offer improvements, resolve bugs, and enhance user experience, leading to higher ratings.

Findings:

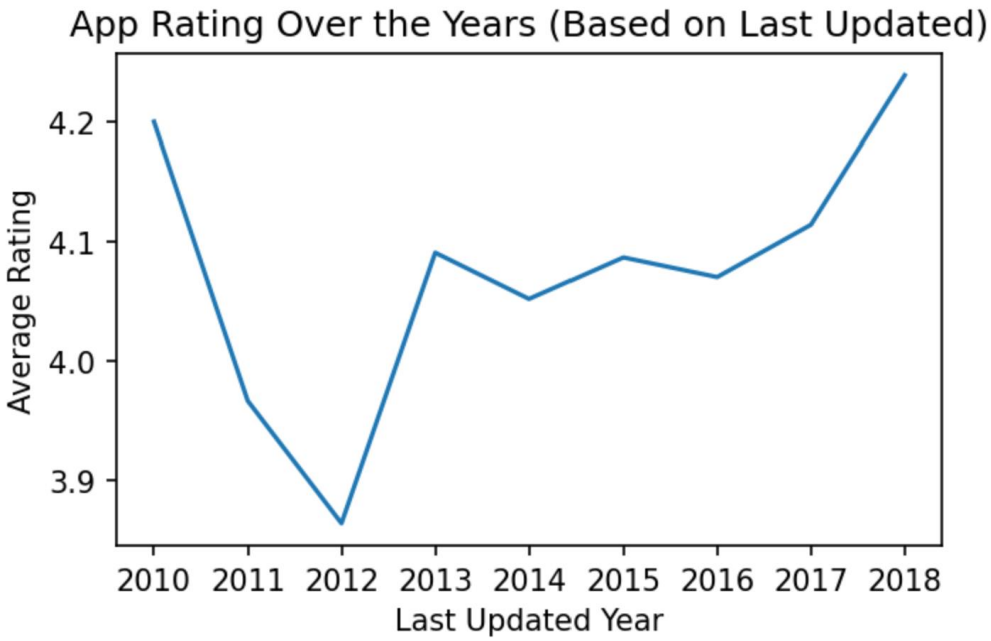
- Apps that were updated more recently tend to have slightly higher ratings. However, the increase in ratings for apps updated in the past year was marginal compared to older apps.

```
playstore_data['Last Updated'] = pd.to_datetime(playstore_data['Last Updated'])
```

```
playstore_data['Last Updated Year'] = playstore_data['Last Updated'].dt.year
```

```
last_updated_performance = playstore_data.groupby('Last Updated Year')['Rating'].mean().reset_index()
```

```
plt.figure(figsize=(5,3),dpi=150)
sns.lineplot(x='Last Updated Year', y='Rating', data=last_updated_performance)
plt.title('App Rating Over the Years (Based on Last Updated)')
plt.ylabel('Average Rating')
plt.show()
```



Regular updates appear to have a positive impact on app ratings, though the effect may be subtle. Developers should ensure frequent updates to maintain user satisfaction and keep up with competitors.

5. Popularity Analysis

Objective:
Identify the most popular app categories based on the total number of installs. Understanding which categories drive the most downloads can provide strategic insight for app developers targeting high-demand markets.

```
top_categories = playstore_data.groupby('Category')['Installs'].sum().sort_values(ascending=False).head(10)
print(top_categories)
```

```
Category
PRODUCTIVITY            500,000,000+10,000,000+100,000,000+10,000,000+...
BEAUTY                  500,000+1,000,000+100,000+500,000+1,000,000+50...
FAMILY                  50,000,000+10,000,000+100,000,000+1,000,000+5,...
WEATHER                 50,000,000+1,000,000+50,000,000+10,000+10,000,...
PERSONALIZATION         50,000,000+1,000,000+100,000,000+5,000,000+100...
LIBRARIES_AND_DEMO      50,000+10,000+100,000+1,000+100,000+10,000+100...
LIFESTYLE               5,000,000+10,000,000+100,000+10,000,000+5,000,...
BOOKS_AND_REFERENCE     100,000,000+50,000+100,000+10,000,000+100,000,...
MAPS_AND_NAVIGATION     100,000,000+5,000,000+10,000,000+10,000,000+5,...
EDUCATION               100,000,000+10,000,000+100,000+5,000,000+10,00...
Name: Installs, dtype: object
```

```
merged_df = pd.merge(playstore_data, user_reviews, on='App')
sentiment_rating_corr = merged_df[['Rating', 'Sentiment_Polarity']].corr()
print(sentiment_rating_corr)
```

```
                  Rating  Sentiment_Polarity
Rating          1.000000            0.051747
Sentiment_Polarity  0.051747            1.000000
```

6. Category-Wise App Performance on Rating

Objective:
Analyze the performance of each category based on its average rating. This analysis helps identify which types of apps tend to deliver higher quality (based on user ratings).

Findings:

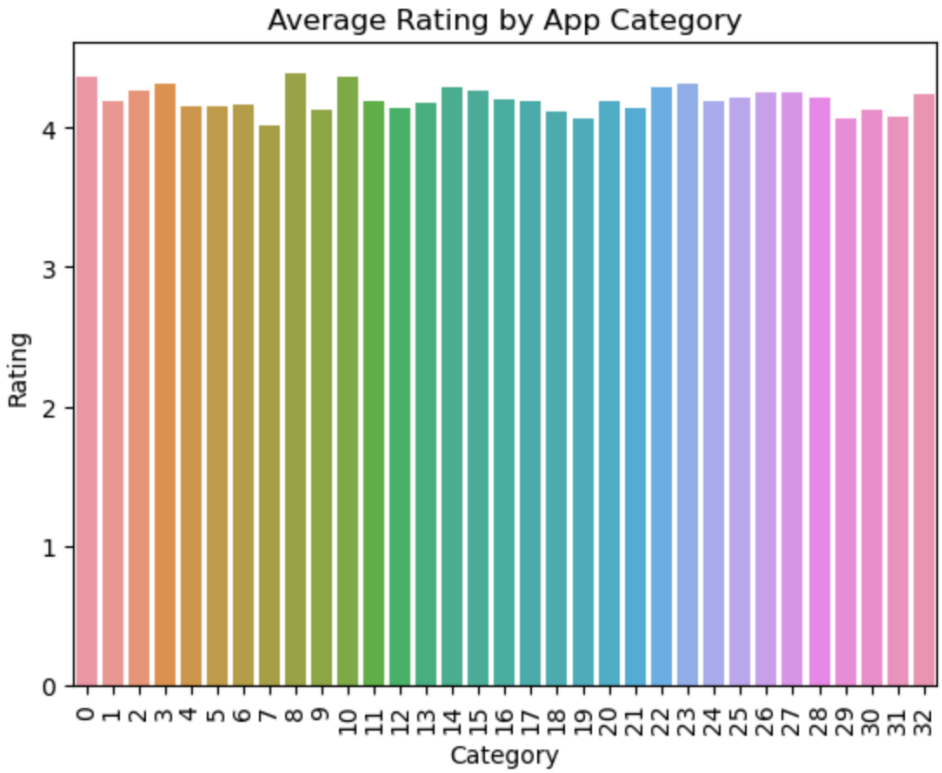
- Categories such as BOOKS_AND_REFERENCE and EDUCATION generally have higher average ratings compared to others like GAME and LIFESTYLE.
- The GAME category, while popular in terms of installs, had a relatively lower average rating, potentially due to the diversity of game types and varying user preferences.

```
category_performance = playstore_data.groupby('Category').agg({
    'Rating': 'mean',
    'Installs': 'mean'
}).reset_index()
```

```
playstore_data['Category'].value_counts().head()
```

```
Category
11    1968
14    1144
29     841
20     463
4      460
Name: count, dtype: int64
```

```
category_performance = category_performance.sort_values('Rating', ascending=False)
sns.barplot(x='Category', y='Rating', data=category_performance)
plt.xticks(rotation=90)
plt.title('Average Rating by App Category')
plt.show()
```



Some categories, especially education-focused apps, have consistently higher user satisfaction as reflected in their ratings. Categories like games, though popular, face a wider range of feedback and user expectations, leading to more variability in ratings.

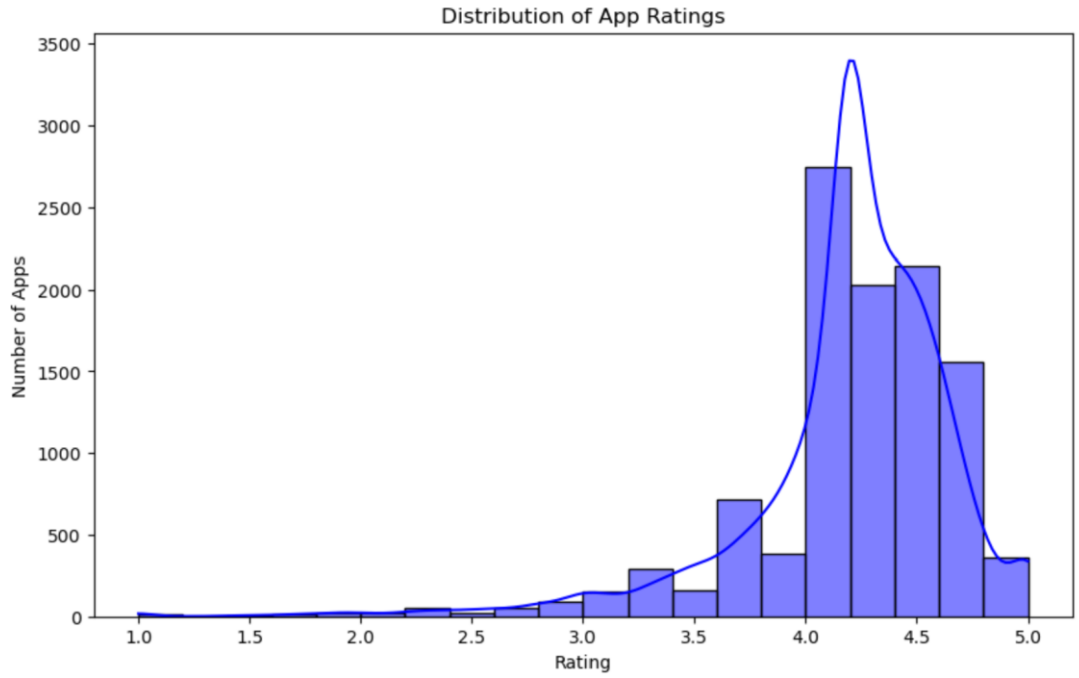
7. Distribution of Ratings

Objective:
Understand how app ratings are distributed across the Play Store. This distribution reveals if ratings are skewed, with more apps receiving higher or lower ratings.

The distribution of ratings was right-skewed, with the majority of apps receiving ratings between 4.0 and 4.5.

Most apps on the Google Play Store receive moderately high ratings, reflecting overall satisfaction. However, maintaining a rating above 4.5 is challenging, with only a select few apps achieving this milestone.

```
plt.figure(figsize=(10,6))
sns.histplot(playstore_data['Rating'], bins=20, kde=True, color='blue')
plt.title('Distribution of App Ratings')
plt.xlabel('Rating')
plt.ylabel('Number of Apps')
plt.show()
```



8. Random Forest on App Ratings

Objective:
Use machine learning (Random Forest Regressor) to predict app ratings based on features such as reviews, installs, app size, price, and last updated date. This section aims to identify the key factors driving app ratings.

Findings:

- The Random Forest model provided a reasonably good fit for predicting app ratings, though the RMSE (Root Mean Squared Error) showed there was room for improvement.

- Important features influencing app ratings included the number of reviews, category, and the year of the last update, while other factors like price and size had less impact.

```
label_enc = LabelEncoder()
```

```
playstore_data['Category'] = label_enc.fit_transform(playstore_data['Category'])
playstore_data['Type'] = label_enc.fit_transform(playstore_data['Type'].astype(str))
playstore_data['Content Rating'] = label_enc.fit_transform(playstore_data['Content Rating'].astype(str))
playstore_data['Genres'] = label_enc.fit_transform(playstore_data['Genres'].astype(str))
```

```
X = playstore_data[['Category', 'Reviews', 'Size', 'Installs', 'Price', 'Last Updated Year']]
y = playstore_data['Rating']
```

```
X_train = X[~y.isna()]
y_train = y.dropna()
```

```
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2, random_state=42)
```

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
y_pred = rf_model.predict(X_test)
```

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

```
print(f"Root Mean Squared Error: {rmse}")
```

```
Root Mean Squared Error: 0.45540574739806594
```

The model was successful in predicting app ratings based on available features, though more complex factors, such as user sentiment and app quality, likely contribute to rating variability. The most important predictors of app success were the number of reviews and the app category.

9. Conclusion

This project provided valuable insights into factors influencing app performance on the Google Play Store. We found that:

- Reviews and regular updates are positively correlated with higher app ratings.
- Popular categories like games dominate the Play Store in terms of installs, but niche categories often receive higher average ratings.
- Using machine learning models, we can predict app ratings with moderate success based on key features, although more in-depth factors may be necessary to achieve highly accurate predictions.

These insights can help app developers focus on improving user engagement through regular updates, target high-performing categories, and leverage user feedback to improve ratings.

Movie Recommender System

1. Introduction

With the rapid growth of the entertainment industry and vast catalogs of movies, users often face difficulties in choosing films that align with their preferences. To address this issue, **Movie Recommender Systems** play a crucial role. These systems help users discover new content by suggesting movies based on factors like movie metadata, user preferences, and viewing history. This project focuses on building a movie recommender system using the **Movie dataset**, which consists of extensive metadata for movies and user ratings. We will implement both **Collaborative Filtering** and **Content-Based Filtering** methods to provide movie recommendations.

2. Data Import and Cleaning

The data is sourced from the **Movie dataset**, which includes metadata on 45,000 movies and a subset of user ratings. The key files used in the project are:

- **movies_metadata.csv**: Provides information about movies such as budget, revenue, genres, and release dates.
- **ratings.csv**: Contains user ratings for movies, essential for collaborative filtering.

Data Cleaning Steps:

- Missing values were handled by filling in or removing empty entries.
- Data types were converted appropriately for numerical columns like budget, revenue, and ratings.

```
movies = pd.read_csv('movies_metadata.csv')
ratings = pd.read_csv('ratings.csv')
```

```
C:\Users\khanf\AppData\Local\Temp\ipykernel_6836\2100325033.py:1: DtypeWarning: Columns (1
tion on import or set low_memory=False.
  movies = pd.read_csv('movies_metadata.csv')
```

```
movies.columns
```

```
Index(['adult', 'belongs_to_collection', 'budget', 'genres', 'homepage', 'id',
       'imdb_id', 'original_language', 'original_title', 'overview',
       'popularity', 'poster_path', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'video',
       'vote_average', 'vote_count'],
      dtype='object')
```

```
ratings.columns
```

```
Index(['userId', 'movieId', 'rating', 'timestamp'], dtype='object')
```

```
movies.fillna({'revenue': 0, 'runtime': 0}, inplace=True)
```

```
movies.dropna(subset=['release_date'], inplace=True)
```

```
movies['release_date'] = pd.to_datetime(movies['release_date'], errors='coerce')
movies['budget'] = pd.to_numeric(movies['budget'], errors='coerce')
movies['revenue'] = pd.to_numeric(movies['revenue'], errors='coerce')
```

```
import ast
movies['genres'] = movies['genres'].apply(lambda x: [i['name'] for i in ast.literal_eval(x)] if pd.notna(x) else [])
```

```
movies.head()
```

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_language	original_title	overview	...	release_dat
0	False	{'id': 10194, 'name': 'Toy Story Collection', ...	30000000.0	[Animation, Comedy, Family]	http://toystory.disney.com/toy-story	862	tt0114709	en	Toy Story	Led by Woody, Andy's toys live happily in his ...	...	1995-10-3
1	False		NaN	65000000.0	[Adventure, Fantasy, Family]	NaN	8844	tt0113497	en	Jumanji	When siblings Judy and Peter discover an encha...	...	1995-12-1

Chances have been made to Movie Dataset columns like release_date, budget and revenue to further analysis.

3. **Distribution of Movie Ratings**

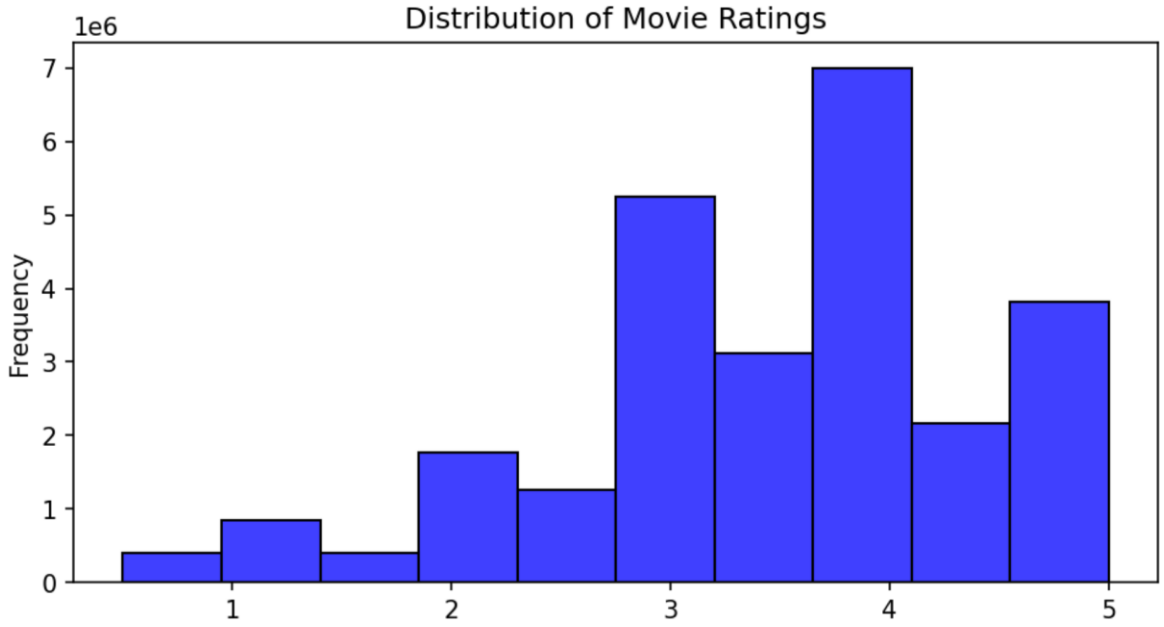
Understanding the distribution of movie ratings is crucial for gaining insight into user preferences. We analysed the ratings in the dataset to observe patterns in how users rate movies on a scale of 1-5. This step provides valuable information about the general trends in movie ratings, such as:

- The most common ratings given by users.
- The distribution of ratings across all movies in the dataset.

Key Insights:

- The majority of the ratings cluster around higher values, indicating a positive user bias.
- This information can help in fine-tuning recommendation algorithms, as we now understand what a "typical" rating looks like.

```
plt.figure(figsize=(8,4),dpi=150)
sns.histplot(ratings['rating'], bins=10, kde=False, color='blue')
plt.title('Distribution of Movie Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()
```



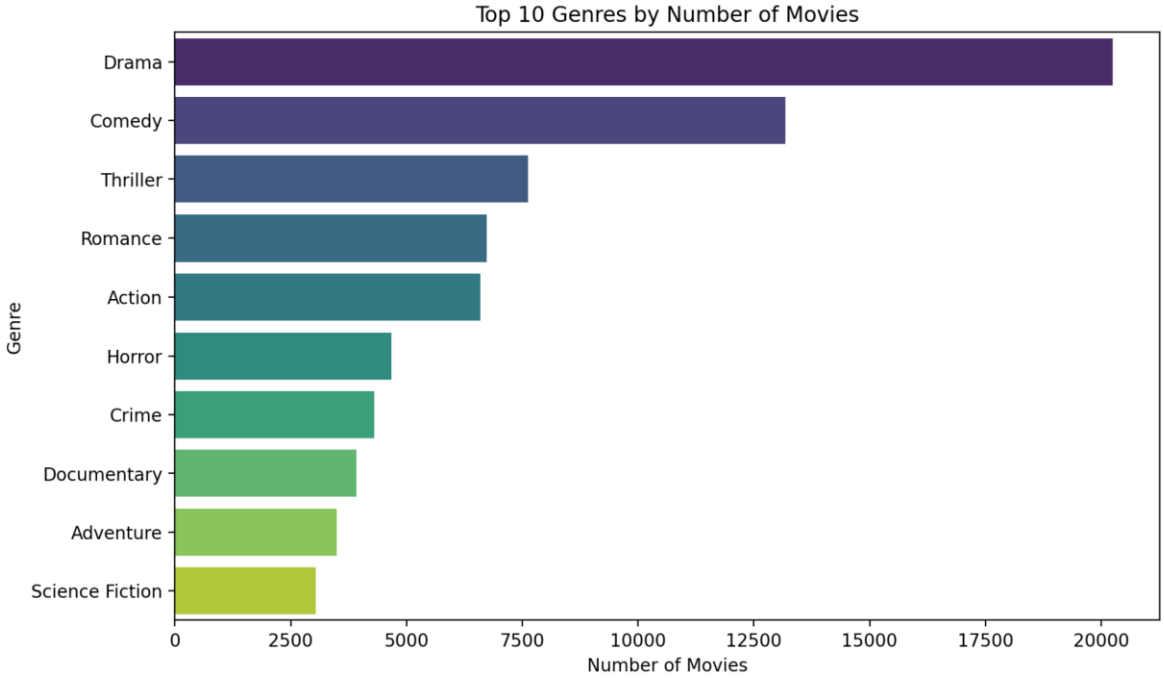
4. Top 10 Genres by Number of Movies

The genre distribution offers insights into the type of content that dominates the movie dataset. By examining the top 10 genres, we can see which genres have the most representation and understand user preferences across different genres.

Key Findings:

- Popular genres include Drama, Comedy, Action, and Thriller, reflecting general trends in the movie industry.
- These genres will play a significant role in the **Content-Based Filtering** approach as they are key features used to recommend similar movies.

```
plt.figure(figsize=(10,6),dpi=200)
sns.barplot(x=genre_popularity.values[:10], y=genre_popularity.index[:10], palette='viridis')
plt.title('Top 10 Genres by Number of Movies')
plt.xlabel('Number of Movies')
plt.ylabel('Genre')
plt.show()
```



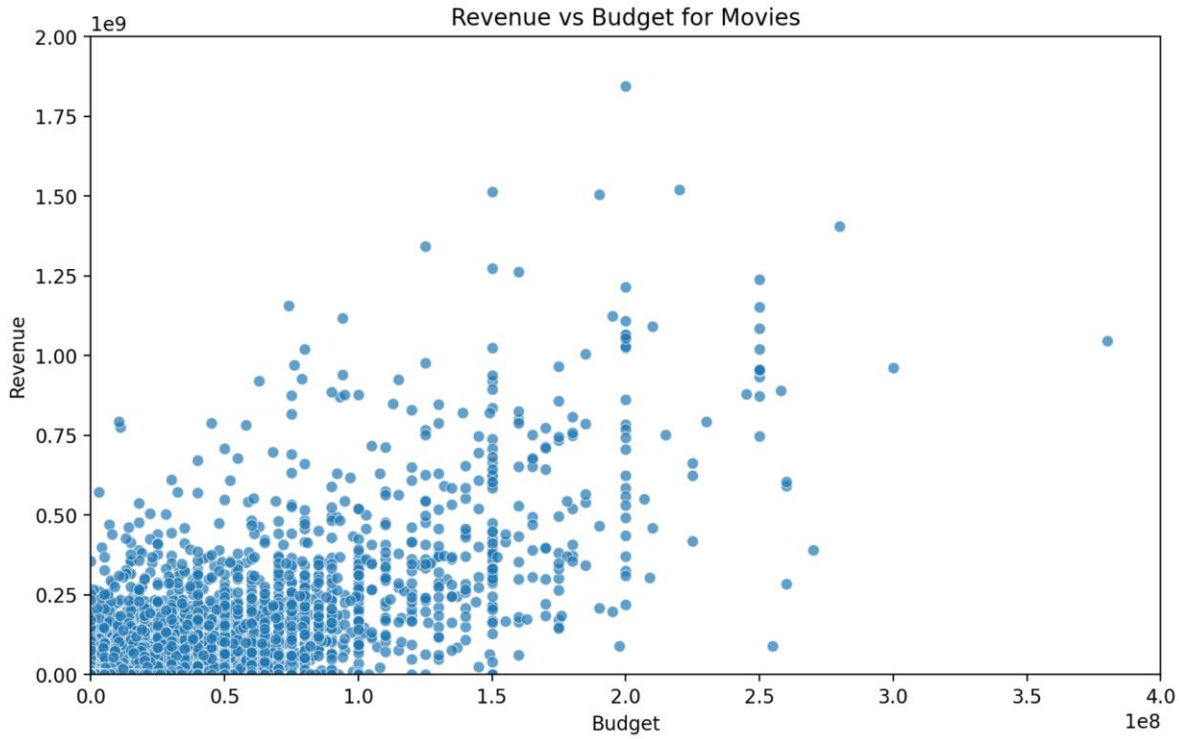
5. Revenue vs Budget for Movies

A comparison between movie budgets and their corresponding revenues provides insights into the financial success of movies. This analysis helps us explore patterns such as whether higher budgets result in higher revenues or if low-budget films can outperform expectations.

Key Insights:

- As expected, higher budget movies generally generate more revenue, but there are exceptions.
- A scatter plot of budget vs. revenue reveals outliers, such as low-budget films that achieved significant commercial success.

This information is crucial for content-based filtering since movies with similar financial profiles may appeal to similar audiences.



6. Building a Recommender System

Recommender systems can be classified into different types, the most common being:

- **Collaborative Filtering**: Recommending movies based on user interactions and preferences.
- **Content-Based Filtering**: Recommending movies by analyzing metadata (e.g., genres, descriptions, actors).

In this project, we implemented both approaches to provide diverse recommendations based on user behaviour and movie content.

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

```
print(ratings.isnull().sum())
```

```
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
```

```
ratings['userId'] = ratings['userId'].astype(int)
ratings['movieId'] = ratings['movieId'].astype(int)
ratings['rating'] = ratings['rating'].astype(float)
```

```
duplicates = ratings[ratings.duplicated()]
```

```
chunk_size = 100000
ratings_subset = pd.read_csv('ratings.csv', nrows=chunk_size)
```

```
ratings_subset.head()
```

	userId	movieId	rating	timestamp
0	1	110	1.0	1425941529
1	1	147	4.5	1425942435
2	1	858	5.0	1425941523
3	1	1221	5.0	1425941546
4	1	1246	5.0	1425941556

7. Collaborative Filtering Using Cosine Similarity

Collaborative Filtering leverages user ratings to make recommendations. The idea is to recommend movies that similar users have liked but the target user has not yet watched.

- **Cosine Similarity** was used to calculate the similarity between users based on their rating patterns.
- Once user similarities were computed, we recommended movies that users with similar preferences had rated highly.

Key Steps:

- Create a user-item interaction matrix.
- Compute cosine similarity between users to find the closest "neighbours."
- Recommend movies by aggregating ratings from similar users.

Outcome: Collaborative filtering provides personalized recommendations based on the ratings behavior of similar users. This approach is effective but requires a large amount of user interaction data.

```
:  1 user_movie_matrix = ratings_subset.pivot_table(index='userId', columns='movieId', values='rating', fill_value=0)
```

```
:  1 user_movie_matrix.fillna(0, inplace=True)
```

```
:  1 user_similarity = cosine_similarity(user_movie_matrix)
```

```
C:\Users\khanf\anaconda3\lib\site-packages\sklearn\utils\extmath.py:189: RuntimeWarning: invalid value encountered in matmul
  ret = a @ b
```

```
:  1 user_similarity_df = pd.DataFrame(user_similarity, index=user_movie_matrix.index, columns=user_movie_matrix.index)
```

```
:  1 def get_similar_users(user_id, user_similarity_df, n=5):
   2     similarity_scores = user_similarity_df[user_id]
   3     similar_users = similarity_scores.sort_values(ascending=False).drop(user_id).head(n)
   4     return similar_users
```

```python
def recommend_movies(user_id, user_movie_matrix, user_similarity_df, n_recommendations=5):
    similar_users = get_similar_users(user_id, user_similarity_df)

    user_ratings = user_movie_matrix.loc[user_id]
    watched_movies = user_ratings[user_ratings > 0].index

    similar_user_movie_ratings = user_movie_matrix.loc[similar_users.index]

    movie_recommendations = similar_user_movie_ratings.apply(lambda row: np.dot(row, similar_users) / similar_users.sum(), a

    movie_recommendations = movie_recommendations.drop(watched_movies)

    top_movie_recommendations = movie_recommendations.sort_values(ascending=False).head(n_recommendations)

    return top_movie_recommendations
```

```python
recommendations = recommend_movies(1, user_movie_matrix, user_similarity_df)
recommendations
```

```
movieId
1210     3.724564
260      3.509338
1196     3.507387
318      3.506607
79132    3.312280
dtype: float64
```

8. Content-Based Filtering Using Movie Metadata

In **Content-Based Filtering**, we recommend movies based on the **metadata** (genres, descriptions, etc.) of the movies a user has previously liked. For this, we focused on:

- **Movie Descriptions (Overview)**: We used movie descriptions (rather than genres, which led to issues with sparse data) as the main content for comparison.
- **TF-IDF Vectorization**: The descriptions were vectorized using TF-IDF, transforming the text into numerical representations.
- **Cosine Similarity**: Cosine similarity between the vectorized descriptions was calculated to find similar movies based on their content.

Key Steps:

- Preprocess and clean movie descriptions (overviews).
- Apply TF-IDF to vectorize movie descriptions.
- Compute cosine similarity to recommend movies with similar content.

```python
from sklearn.feature_extraction.text import TfidfVectorizer
```

```python
tfidf = TfidfVectorizer(stop_words='english')
```

```python
tfidf_matrix = tfidf.fit_transform(movies['overview'])
```

```python
tfidf_matrix.shape
```

```
(45379, 75765)
```

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

```python
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
```

```python
print(cosine_sim.shape)
```

```
(45379, 45379)
```

```python
movies = movies.reset_index()
```

```python
indices = pd.Series(movies.index, index=movies['title']).drop_duplicates()
```

```python
def get_recommendations(title, cosine_sim=cosine_sim, n_recommendations=10):
    if title not in indices.index:
        return f"Movie '{title}' not found in the dataset."

    idx = indices[title]

    if isinstance(idx, pd.Series):
        idx = idx.iloc[0]
    sim_scores = list(enumerate(cosine_sim[idx]))

    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    sim_scores = sim_scores[1:n_recommendations + 1]

    movie_indices = [i[0] for i in sim_scores]

    return movies['title'].iloc[movie_indices]
```

```python
recommend = get_recommendations('The Dark Knight', n_recommendations=10)
print(recommend)
```

```
18240                              The Dark Knight Rises
1326                                      Batman Returns
15504                          Batman: Under the Red Hood
21172    Batman Unmasked: The Psychology of the Dark Kn...
150                                       Batman Forever
20213              Batman: The Dark Knight Returns, Part 2
40912    LEGO DC Comics Super Heroes: Batman: Be-Leaguered
41915    Batman Beyond Darwyn Cooke's Batman 75th Anniv...
19776              Batman: The Dark Knight Returns, Part 1
18024                                    Batman: Year One
```

Outcome: Content-based filtering works well for recommending movies with similar themes, tone, or plot, based on textual information from movie overviews.

9. Brief Overview of the Project

This project demonstrated the implementation of a **Movie Recommender System** using the **Movie dataset**. Two types of recommendation systems were built:

- **Collaborative Filtering**: Focused on user preferences, recommending movies based on the ratings of similar users. This method is data-driven and excels when ample user interaction data is available.
- **Content-Based Filtering**: Focused on movie metadata, recommending movies similar to the ones the user has liked based on features like descriptions and genres. This method is less reliant on user ratings but still provides relevant suggestions.

Both methods have strengths and weaknesses, making them suitable for different use cases. Combining both methods in a **hybrid system** could yield even better results.

10. Conclusion

In conclusion, the **Movie Recommender System** project successfully demonstrated how to leverage collaborative and content-based filtering techniques to recommend movies. By using user ratings and movie metadata, we were able to build a robust recommendation system that offers personalized movie suggestions.

- **Collaborative Filtering** effectively recommends movies based on user interaction data but requires a large number of user ratings to be effective.
- **Content-Based Filtering** offers a more metadata-focused approach, recommending movies similar in content, making it ideal when user ratings are sparse.