CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY

DATA ANALYSIS AND VISUALIZATION (22ADE01)

A Course End Project on

YOUTUBE VIDEO ANALYTICS

Submitted by:

CHANDANA D 1601-23-737-012

Submitted to:

Dr Ramakrishna Kolikipogu Professor, Dept of IT

Class IT 1, Sem IV

CERTIFICATE OF COMPLETION

This is to certify that the course end project work entitled "YouTube Video Analytics" is submitted by Chandana Devanaboyina (160123737012) in partial fulfillment of the requirements for the award of CIE Marks of DATA ANALYSIS AND VISUALIZATION (22ADE01) of B.E., IV-SEM, INFORMATION TECHNOLOGY to CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A) affiliated to OSMANIA UNIVERSITY, HYDERABAD. This report is a record of bonafide work carried out by them under my supervision and guidance. The results embodied in this report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Signature of Course Faculty

Dr Ramakrishna Kolikipogu

Professor, Dept of IT, CBIT

Acknowledgment

I would like to extend my heartfelt gratitude to everyone who has supported me throughout the process of completing this project. Firstly, I would like to thank my mentor, whose valuable guidance, feedback, and unwavering support have been instrumental in shaping the outcome of this report. Their expertise and constant encouragement pushed me to think critically and adopt a structured approach towards data analysis.

I am also deeply grateful to the Kaggle community, whose datasets, tutorials, and resources served as the foundation of my work. In particular, the "YouTube New Dataset" provided rich and comprehensive metadata, which enabled the analysis to be as meaningful and informative as possible. Lastly, I would like to express my appreciation to the creators and contributors of the Python programming language and its associated libraries, such as Pandas, Matplotlib, and Seaborn, which played a crucial role in my ability to manipulate and visualize the data effectively.

Abstract

This report explores the application of data mining techniques to YouTube video metadata in an effort to analyze video performance metrics. Using a dataset sourced from Kaggle, the study investigates various aspects of video engagement, including views, likes, dislikes, comments, and upload times. The report employs several key data mining techniques such as data preprocessing, correlation analysis, and data visualization to uncover patterns and relationships within the dataset. Specifically, the analysis examines the correlation between engagement metrics, the distribution of views across videos, and the optimal time for uploading videos. In addition, the report provides insights into the most common video categories, as well as the relationship between the number of views and other performance metrics. The findings of this study are aimed at content creators seeking to optimize their strategies for increasing engagement and improving video performance on YouTube.

Table of Contents

1. Introduction

The rapid growth of YouTube has transformed the platform into one of the largest and most influential sources of content, with millions of users globally engaging in video consumption, creation, and sharing. As of recent years, content creators have been striving to increase the visibility and engagement of their videos, utilizing analytics to optimize their strategies. However, with an overwhelming volume of videos uploaded daily, it becomes increasingly important to understand the key factors that drive video success.

This report aims to provide insights into the performance of YouTube videos using data mining techniques. By leveraging a comprehensive dataset containing attributes such as views, likes, dislikes, comments, and categories, this study investigates how various factors such as engagement metrics and upload times impact a video's success. The dataset, which was sourced from Kaggle, consists of video metadata from the United States and offers a wide variety of features that allow for in-depth analysis.

The ultimate goal of this report is to uncover actionable insights that can assist content creators in tailoring their videos for improved performance on YouTube. The findings can also be beneficial for marketers and analysts looking to understand the factors influencing video engagement.

Objectives of the Project

1. **Analyze YouTube Video Engagement:** Evaluate the correlation between various engagement metrics such as views, likes, dislikes, and comments.
2. **Visualize Video Upload Patterns:** Identify trends in video uploads based on time-series analysis and determine peak hours for publishing content.
3. **Categorize Video Content:** Map category IDs to their respective names and analyze the distribution of different video categories on YouTube.
4. **Evaluate Engagement by Category:** Compute and compare average engagement metrics (likes, dislikes, and comments) across different video categories.
5. **Perform Sentiment Analysis:** Analyze video titles using sentiment analysis to classify them as Positive, Neutral, or Negative.

6. **Study the Impact of Clickbait Titles:** Assess whether clickbait words in video titles influence video popularity and engagement.
7. **Track Time-Series Trends of YouTube Views:** Examine how viewership changes over time using time-series visualizations.

Expected Outcomes

1. **Understanding of Audience Engagement:** Identify key factors that drive video popularity and engagement on YouTube.
2. **Optimal Publishing Strategies:** Provide insights into the best time to upload videos based on historical data.
3. **Category-Specific Insights:** Determine which content categories receive the highest engagement and viewer interaction.
4. **Influence of Clickbait Titles:** Evaluate the effectiveness of clickbait in increasing viewership.
5. **Sentiment Trends in Video Titles:** Discover how the sentiment of video titles correlates with audience engagement.
6. **Data-Driven Decision Making:** Enable content creators to optimize their video strategies based on data analytics.
7. **Interactive Visualizations:** Provide meaningful graphical representations to help interpret complex datasets effectively.

2. Methodology

Data Collection

The dataset used for this analysis was sourced from Kaggle, specifically the "YouTube New Dataset" contributed by the user datasnaek. The dataset comprises a wide range of features, including video titles, descriptions, categories, views, likes, dislikes, comment counts, and publication time. This dataset provides a comprehensive view of YouTube video performance, making it an ideal source for analysis. The data is publicly available on Kaggle, a leading platform for data science competitions and datasets, which ensured that the dataset was both reliable and high-quality.

The dataset contains metadata for thousands of videos, allowing for a robust exploration of video performance metrics across different categories. Given the variety of metrics available, it was possible to explore relationships between views, likes, and other engagement indicators, providing a holistic view of video success.

```python
""" Download the latest YouTube dataset from Kaggle and print its file path. """
import kagglehub
path = kagglehub.dataset_download("datasnaek/youtube-new")
print("Path to dataset files:", path)
```

```
Path to dataset files: /root/.cache/kagglehub/datasets/datasnaek/youtube-new/versions/115
```

Data Cleaning and Preprocessing

Before diving into the analysis, it was essential to clean and preprocess the data. The raw dataset, like many real-world datasets, contained missing values, incorrect data types, and other inconsistencies. The first step in preprocessing involved handling missing data. Missing values for the columns 'likes' and 'dislikes' were filled with the median of the respective columns. This imputation technique was chosen because the median is less sensitive to extreme values compared to the mean, ensuring that the replacement values did not distort the data distribution.

For other columns, such as 'views' and 'title', rows with missing values were dropped. These columns are critical to the analysis, and any row without this information would be unusable for further computations.

Another important preprocessing step involved converting the data types of certain columns. Columns like 'views', 'likes', and 'dislikes' were initially represented as floating-point numbers. Since these columns represent count data, it was necessary to convert them into integers. This was achieved using the .astype(int) method, which ensured that all numerical operations (such as summing or averaging) were performed accurately.

3. System Architecture

This analysis was conducted using **Google Colab**, an interactive cloud-based environment that allows users to run Python code. Colab provides the advantage of access to high-performance computational resources (including GPUs) without the need for complex hardware setups, making it an ideal choice for running data analysis and machine learning tasks.

The Python libraries utilized in this project included:

- **Pandas**: For data manipulation and analysis, including data cleaning, transformation, and aggregation.
- **Matplotlib** and **Seaborn**: For data visualization, enabling the creation of various plots and graphs to analyze trends and patterns in the dataset.
- **NumPy**: For numerical operations and handling of large datasets.

Google Colab was selected because of its ease of use, integration with Google Drive, and collaborative features, which allowed for seamless collaboration and sharing of code and results.

```python
# importing all the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

4. Detailed Explanation of Code Snippets

Data Loading and Cleaning

The first step in any data analysis project is to load the dataset into a structured format. Using Pandas, the dataset was loaded from the CSV file provided by Kaggle into a DataFrame using the pd.read_csv() function. This enabled easy manipulation of the data and allowed for initial inspection of the columns and rows. The .head() method was used to view the first few rows, and the .columns attribute was used to print out the column names.

Once the data was loaded, the next step involved identifying and handling missing values. Missing values were first checked using the .isnull().sum() method, which displayed the number of missing values in each column. For columns such as 'likes' and 'dislikes', missing values were filled using the median, a robust method for dealing with missing data in numerical fields. For columns critical to the analysis, such as 'views' and 'title', rows with missing data were removed entirely.

```
""" Download the latest YouTube dataset from Kaggle and print its file path. """
import kagglehub
path = kagglehub.dataset_download("datasnaek/youtube-new")
print("Path to dataset files:", path)
```

```
Path to dataset files: /root/.cache/kagglehub/datasets/datasnaek/youtube-new/versions/115
```

```
# Load the dataset (modify the file name if necessary)
df = pd.read_csv(f"{path}/USvideos.csv")  # Adjust based on your dataset

# Display column names
print(df.columns)
print(df.head())
```

```
Index(['video_id', 'trending_date', 'title', 'channel_title', 'category_id',
       'publish_time', 'tags', 'views', 'likes', 'dislikes', 'comment_count',
       'thumbnail_link', 'comments_disabled', 'ratings_disabled',
       'video_error_or_removed', 'description'],
      dtype='object')
      video_id trending_date  \
0  2kyS6SvSYSE      17.14.11
1  1ZAPwfrtAFY      17.14.11
2  5qpjK5DgCt4      17.14.11
3  puqaWrEC7tY      17.14.11
4  d380meD0W0M      17.14.11

                                              title          channel_title  \
0                   WE WANT TO TALK ABOUT OUR MARRIAGE           CaseyNeistat
1  The Trump Presidency: Last Week Tonight with J...        LastWeekTonight
2  Racist Superman | Rudy Mancuso, King Bach & Le...           Rudy Mancuso
3                     Nickelback Lyrics: Real or Fake?  Good Mythical Morning
4                             I Dare You: GOING BALD!?               nigahiga

   category_id              publish_time  \
0           22  2017-11-13T17:13:01.000Z
1           24  2017-11-13T07:30:00.000Z
2           23  2017-11-12T19:05:24.000Z
3           24  2017-11-13T11:00:04.000Z
4           24  2017-11-12T18:01:41.000Z
```

```
[20] # Check for missing values
print(df.isnull().sum())
# Fill missing values with appropriate values or drop them
df.fillna({"likes": df["likes"].median(), "dislikes": df["dislikes"].median()}, inplace=True)
df.dropna(subset=["views", "title"], inplace=True)
```

```
video_id                    0
trending_date               0
title                       0
channel_title               0
category_id                 0
publish_time                0
tags                        0
views                       0
likes                       0
dislikes                    0
comment_count               0
thumbnail_link              0
comments_disabled           0
ratings_disabled            0
video_error_or_removed      0
description               570
dtype: int64
```

Data Type Conversion

One of the common issues in real-world datasets is that some columns may not have the correct data types. In this case, the 'views', 'likes', and 'dislikes' columns were initially represented as floating-point numbers. Since these columns represent counts, it was important to convert them to integers to prevent issues in subsequent calculations. The conversion was performed using the .astype(int) method, which allowed for precise numerical operations.

```
""" Display column data types and convert 'views', 'likes', and 'dislikes' to integers. """
print(df.dtypes)
df["views"] = df["views"].astype(int)
df["likes"] = df["likes"].astype(int)
df["dislikes"] = df["dislikes"].astype(int)
```

```
video_id                  object
trending_date             object
title                     object
channel_title             object
category_id                int64
publish_time              object
tags                      object
views                      int64
likes                      int64
dislikes                   int64
comment_count              int64
thumbnail_link            object
comments_disabled           bool
ratings_disabled            bool
video_error_or_removed      bool
description               object
dtype: object
```

Exploratory Data Analysis (EDA)

Descriptive Statistics

Descriptive statistics play a crucial role in understanding the distribution and summary of key metrics. By calling the .describe() method on the DataFrame, we generated a summary of statistics for all numerical columns, including 'views', 'likes', 'dislikes', and 'comment_count'. This provided insights into the spread of the data, helping to identify any outliers, trends, or anomalies.

```
[22] df.describe()
```

	category_id	views	likes	dislikes	comment_count
count	40949.000000	4.094900e+04	4.094900e+04	4.094900e+04	4.094900e+04
mean	19.972429	2.360785e+06	7.426670e+04	3.711401e+03	8.446804e+03
std	7.568327	7.394114e+06	2.288853e+05	2.902971e+04	3.743049e+04
min	1.000000	5.490000e+02	0.000000e+00	0.000000e+00	0.000000e+00
25%	17.000000	2.423290e+05	5.424000e+03	2.020000e+02	6.140000e+02
50%	24.000000	6.818610e+05	1.809100e+04	6.310000e+02	1.856000e+03
75%	25.000000	1.823157e+06	5.541700e+04	1.938000e+03	5.755000e+03
max	43.000000	2.252119e+08	5.613827e+06	1.674420e+06	1.361580e+06

Top 10 Videos by Views

To identify which videos garnered the most views, the dataset was sorted by the 'views' column in descending order. The top 10 videos were then extracted and displayed. This analysis revealed which videos were the most popular and allowed for further investigation of their characteristics, such as the number of likes and dislikes.

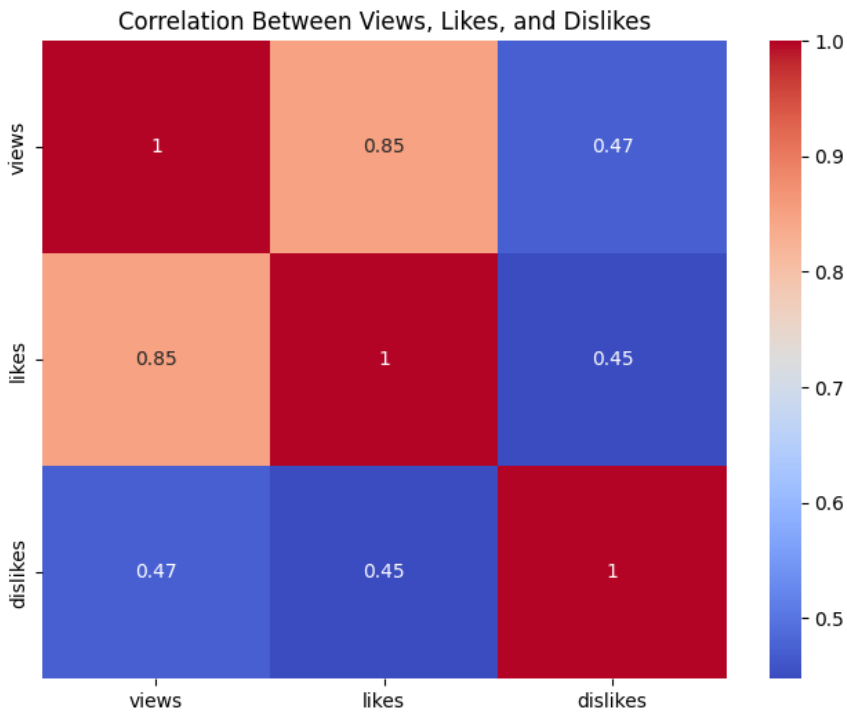
```python
# Find top 10 videos by views
top_videos = df.sort_values(by="views", ascending=False).head(10)
print(top_videos[["title", "views", "likes"]])
```

```
                                                   title      views    likes
38547  Childish Gambino - This Is America (Official V...  225211923  5023450
38345  Childish Gambino - This Is America (Official V...  220490543  4962403
38146  Childish Gambino - This Is America (Official V...  217750076  4934188
37935  Childish Gambino - This Is America (Official V...  210338856  4836448
37730  Childish Gambino - This Is America (Official V...  205643016  4776680
37531  Childish Gambino - This Is America (Official V...  200820941  4714942
37333  Childish Gambino - This Is America (Official V...  196222618  4656929
37123  Childish Gambino - This Is America (Official V...  190950401  4594931
36913  Childish Gambino - This Is America (Official V...  184446490  4512326
36710  Childish Gambino - This Is America (Official V...  179045286  4437175
```

Correlation Analysis Between Views, Likes, and Dislikes

A key part of the analysis was understanding the relationships between different engagement metrics. A heatmap was generated to visualize the correlation matrix between 'views', 'likes', and 'dislikes'. This heatmap displayed the strength of the relationships between these metrics, with positive correlations indicating that videos with higher views typically also had more likes.

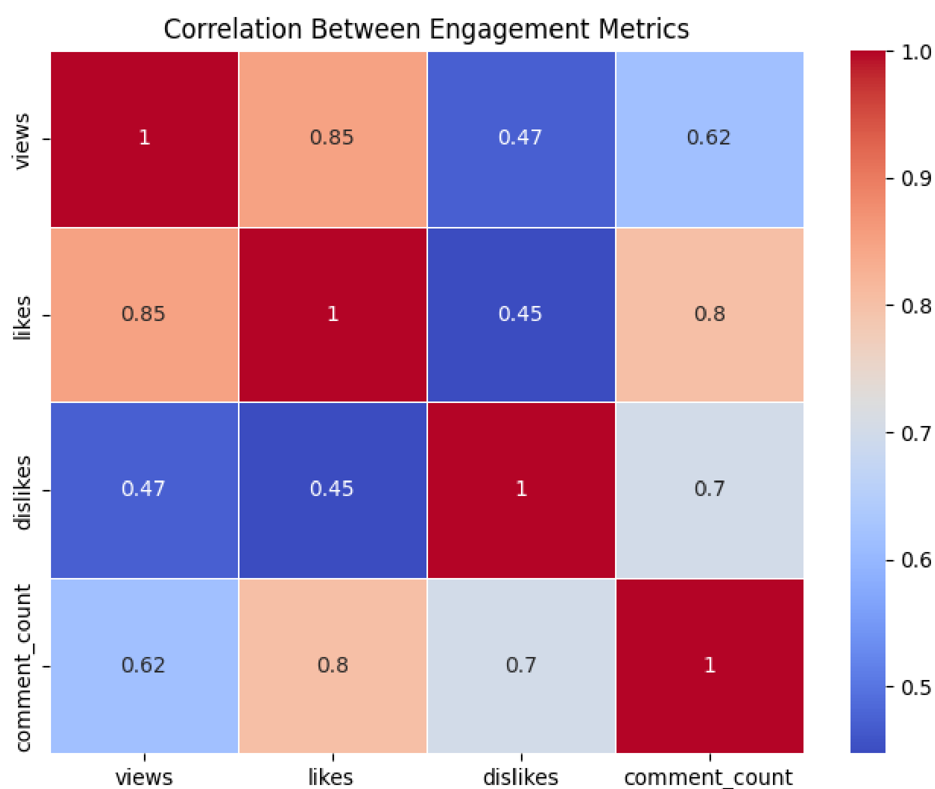
```
#heatmap to analyse the correlation between likes dislikes and views
plt.figure(figsize=(8,6))
sns.heatmap(df[["views", "likes", "dislikes"]].corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Between Views, Likes, and Dislikes")
plt.show()
```



Engagement Correlation with Comment Count

In addition to views and likes, the 'comment_count' column was also included in the correlation analysis. This provided a broader view of video engagement, considering not just passive views and likes, but also active engagement via comments. The heatmap revealed strong correlations between comments and other metrics, emphasizing the importance of user interaction in determining video success.

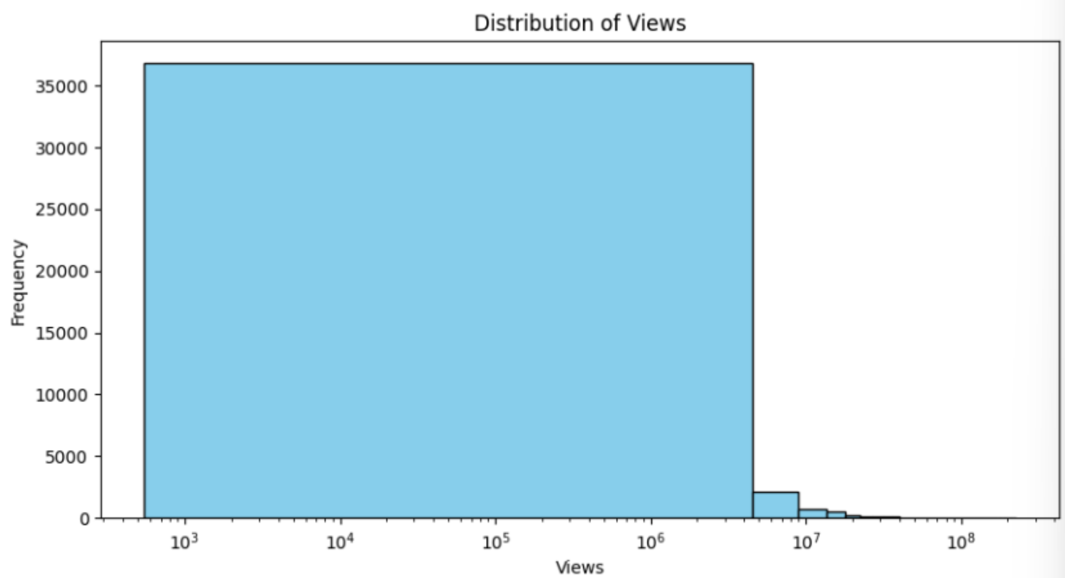
```
[37] # correlation heatmap for video engagement
     import seaborn as sns

     # Select relevant numeric columns
     corr_matrix = df[['views', 'likes', 'dislikes', 'comment_count']].corr()

     # Generate heatmap
     plt.figure(figsize=(8,6))
     sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
     plt.title("Correlation Between Engagement Metrics")
     plt.show()
```



Distribution of Views

A histogram was used to visualize the distribution of views across videos. Given the wide range of view counts, a logarithmic scale was applied to better handle extreme values. The histogram revealed that most videos had relatively few views, with a few highly popular videos skewing the distribution.

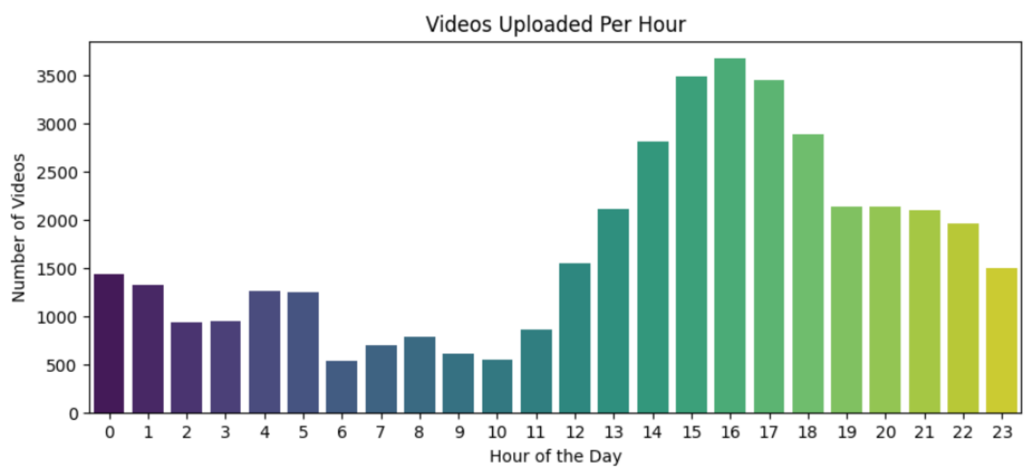
```
#distribution of views using a histogram on a logarithmic scale

plt.figure(figsize=(10,5))
plt.hist(df["views"], bins=50, color="skyblue", edgecolor="black")
plt.xlabel("Views")
plt.ylabel("Frequency")
plt.title("Distribution of Views")
plt.xscale("log")  # Since views can vary greatly
plt.show()
```



Analysis of Upload Time

To explore video upload patterns, the time of video publication was extracted and analyzed. By plotting the number of videos uploaded per hour of the day, we identified trends in upload patterns. The analysis suggested that certain times of day, particularly late afternoon and evening, saw higher video uploads, which might correlate with peak viewership periods.

```
""" Analyze video upload patterns using time series by extracting and visualizing upload hours. """
df["upload_hour"] = pd.to_datetime(df["publish_time"]).dt.hour

plt.figure(figsize=(10,4))
sns.countplot(x=df["upload_hour"], palette="viridis")
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Videos")
plt.title("Videos Uploaded Per Hour")
plt.show()
```

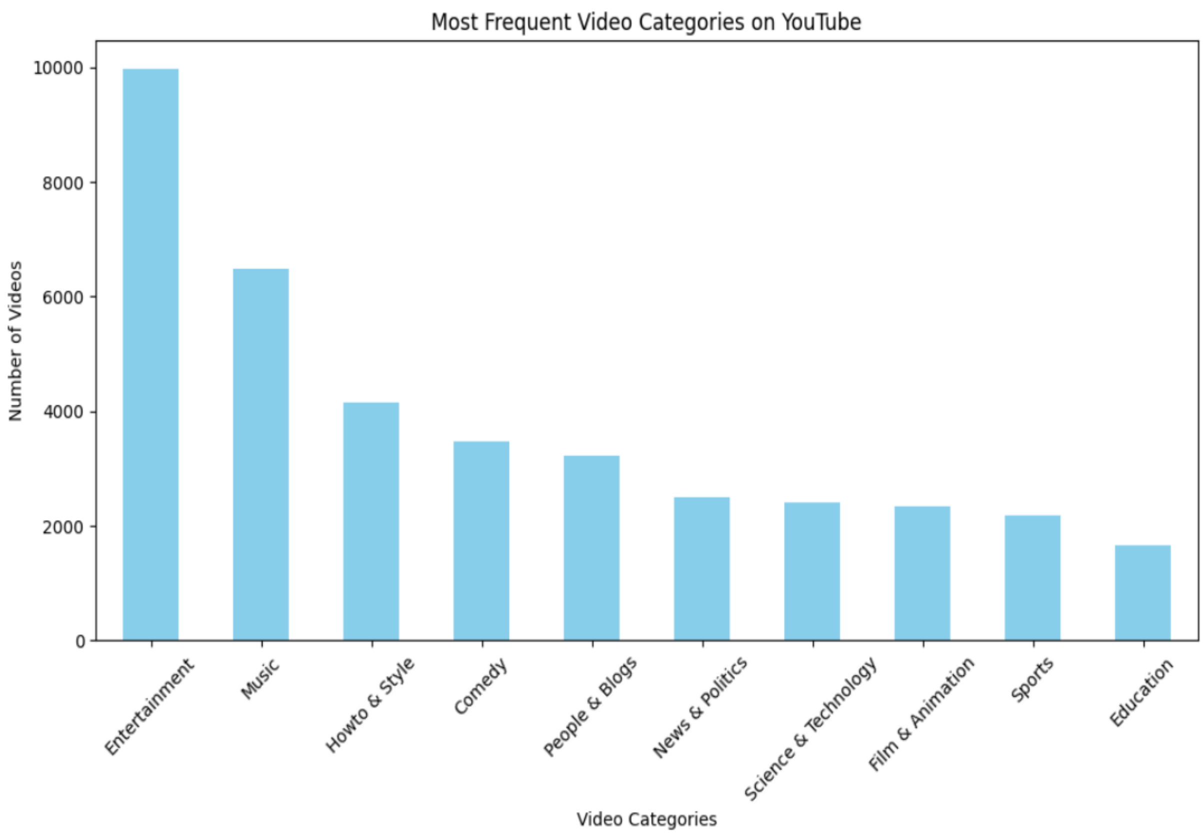


Analysis of YouTube Video Categories

The analysis of YouTube video categories involves mapping numerical category IDs to human-readable names using data from a JSON file. The dataset's category_id column is mapped to corresponding category names, allowing for better interpretability. The frequency of each category is then computed to determine the most common content types. To visualize this distribution, a bar chart is generated using matplotlib.pyplot, highlighting the top ten most frequent categories. This analysis provides insights into content trends on YouTube, helping researchers and content creators understand audience preferences.

```python
""" Map category IDs to names, count occurrences, and visualize the most frequent video categories. """
import json

with open(f"{path}/US_category_id.json", 'r') as f:
    categories = json.load(f)

# Create a mapping of category_id to category_name
category_mapping = {int(cat['id']): cat['snippet']['title'] for cat in categories['items']}

# Map the category_id to category names
df['category_name'] = df['category_id'].map(category_mapping)

# Count occurrences of each category
category_counts = df['category_name'].value_counts()

# Display the most frequent video categories
print(category_counts)

# Visualizing the top categories
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
category_counts.head(10).plot(kind='bar', color='skyblue')
plt.xlabel("Video Categories")
plt.ylabel("Number of Videos")
plt.title("Most Frequent Video Categories on YouTube")
plt.xticks(rotation=45)
plt.show()
```

```
category_name
Entertainment             9964
Music                     6472
Howto & Style             4146
Comedy                    3457
People & Blogs            3210
News & Politics           2487
Science & Technology      2401
Film & Animation          2345
Sports                    2174
Education                 1656
Pets & Animals             920
Gaming                     817
Travel & Events            402
Autos & Vehicles           384
Nonprofits & Activism       57
Shows                       57
Name: count, dtype: int64
```

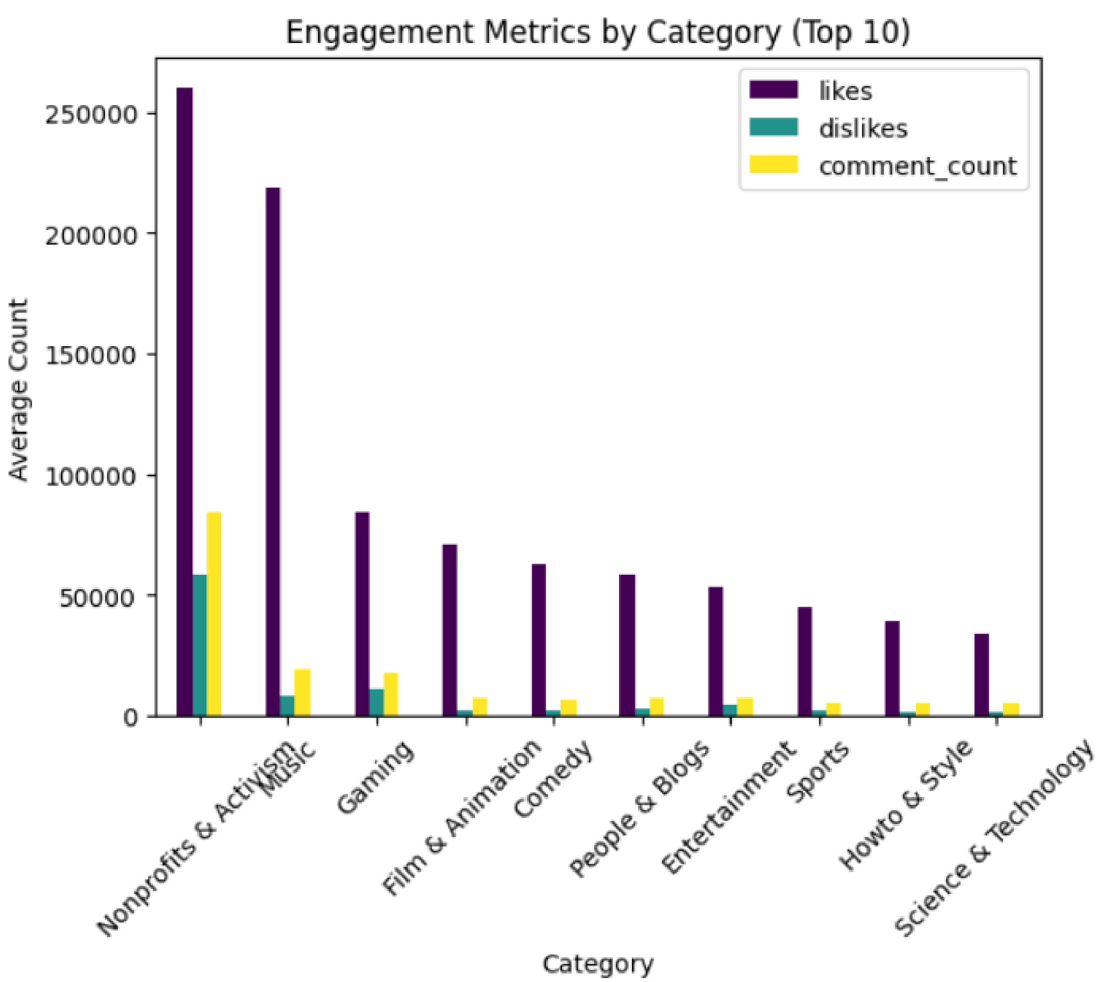


Visualization of Engagement Metrics by Category

This section analyzes the average engagement metrics, including likes, dislikes, and comments, across different YouTube video categories. The category IDs are mapped to their corresponding names using data from a JSON file. The dataset is then grouped by category, and the mean values of engagement metrics are calculated to determine which categories receive the most audience interaction. The data is sorted based on average likes to highlight the most popular categories. Finally, a bar chart is generated to visualize the top ten categories with the highest engagement, offering insights into content performance and viewer preferences.

```
[33] """ Compute and visualize average engagement metrics (likes, dislikes, comments) for each category. """
     # Load category data and create category ID to name mapping
     category_file = f"{path}/US_category_id.json"  # Adjust path if necessary
     with open(category_file, "r") as f:
         category_data = json.load(f)

     category_mapping = {int(item["id"]): item["snippet"]["title"] for item in category_data["items"]}

     # Map category_id to category name
     df["category"] = df["category_id"].map(category_mapping)

     # Compute average engagement metrics per category
     engagement_metrics = df.groupby("category").agg({
         "likes": "mean",
         "dislikes": "mean",
         "comment_count": "mean"
     }).sort_values(by="likes", ascending=False)  # Sort by most liked categories

     # Display engagement statistics
     print(engagement_metrics)

     # Visualize engagement metrics
     plt.figure(figsize=(12, 6))
     engagement_metrics[["likes", "dislikes", "comment_count"]].head(10).plot(kind="bar", colormap="viridis")
     plt.xlabel("Category")
     plt.ylabel("Average Count")
     plt.title("Engagement Metrics by Category (Top 10)")
     plt.xticks(rotation=45)
     plt.show()
```

```
                            likes      dislikes  comment_count
category
Nonprofits & Activism  259923.614035  58076.859649   84364.859649
Music                  218918.199011   7907.757726   19359.764524
Gaming                  84502.183599  11241.696450   18042.488372
Film & Animation        70787.836247   2590.681450    7627.744136
Comedy                  62582.223315   2091.521840    6521.718831
People & Blogs          58135.825234   3173.800935    7719.013084
Entertainment           53243.325070   4314.297772    7383.229426
Sports                  45363.942502   2361.339006    5148.185373
Howto & Style           39286.076942   1320.284370    5583.586589
Science & Technology    34374.276551   1894.378176    4993.721783
Education               29745.031401    816.408213    3286.378019
Pets & Animals          21055.110870    573.238043    2892.070652
Shows                   18993.666667    429.964912    1668.719298
Travel & Events         12030.462687    846.833333    2267.440299
Autos & Vehicles        11056.395833    632.838542    2042.830729
News & Politics          7298.364696   1680.759550    2428.400885
<Figure size 1200x600 with 0 Axes>
```



Sentiment Analysis of Video Titles

This section performs sentiment analysis on YouTube video titles to classify them as Positive, Neutral, or Negative. The TextBlob library is used to compute the sentiment polarity of each title, where values greater than zero indicate positive sentiment, values less than zero indicate negative sentiment, and values equal to zero are categorized as neutral. This classification provides insights into how the wording of video titles might influence viewer engagement and perception. The results are then summarized by counting the occurrences of each sentiment category, offering a statistical overview of sentiment distribution across the dataset.

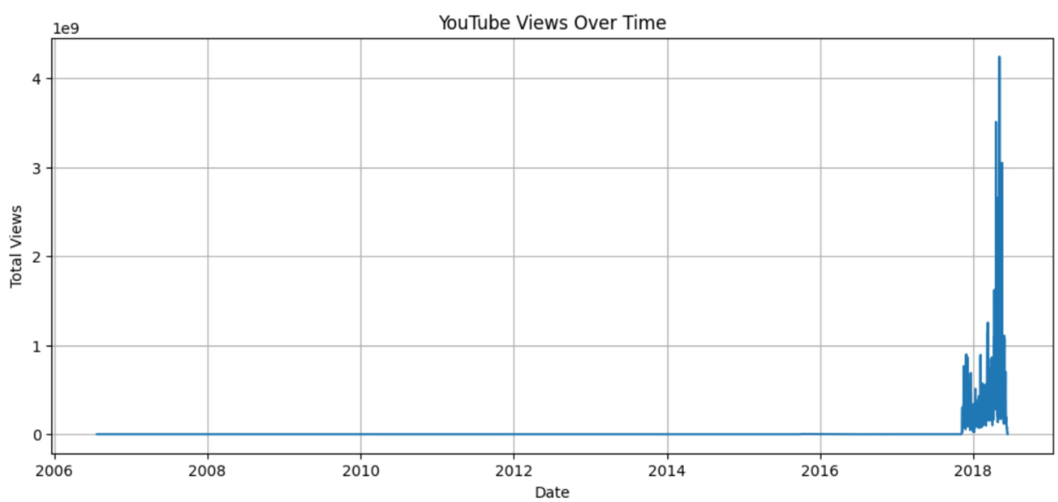
```python
""" Perform sentiment analysis on video titles and categorize as Positive, Neutral, or Negative. """
from textblob import TextBlob

# Function to compute sentiment polarity
def get_sentiment(text):
    return TextBlob(str(text)).sentiment.polarity

# Apply sentiment analysis to video titles
df['title_sentiment'] = df['title'].apply(get_sentiment)

# Categorize sentiment
df['title_sentiment_category'] = df['title_sentiment'].apply(
    lambda x: 'Positive' if x > 0 else ('Negative' if x < 0 else 'Neutral')
)

# Display sentiment distribution
print(df['title_sentiment_category'].value_counts())
```

```
title_sentiment_category
Neutral      23637
Positive     11492
Negative      5820
Name: count, dtype: int64
```

Time-Series Analysis of YouTube Views

This section examines the temporal trends in YouTube video views based on their publication dates. The publish_time column is first converted into a proper datetime format, allowing the extraction of relevant date components such as the publication date and hour. By aggregating the total number of views per day, a time-series dataset is created, which helps in identifying patterns such as spikes in viewership on specific dates. A line graph is then plotted to visualize fluctuations in daily view counts, providing insights into trends like seasonal variations, viral content surges, and the overall engagement patterns of YouTube audiences over time.

```python
""" Analyze time-series trends of YouTube views based on video publish dates. """
# Convert publish_time to datetime
df['publish_time'] = pd.to_datetime(df['publish_time'])

# Extract date parts
df['publish_date'] = df['publish_time'].dt.date
df['publish_hour'] = df['publish_time'].dt.hour

# Aggregate views per day
daily_trend = df.groupby('publish_date')['views'].sum()

# Plot time-series trend
import matplotlib.pyplot as plt

plt.figure(figsize=(12,5))
daily_trend.plot()
plt.title("YouTube Views Over Time")
plt.xlabel("Date")
plt.ylabel("Total Views")
plt.grid()
plt.show()
```

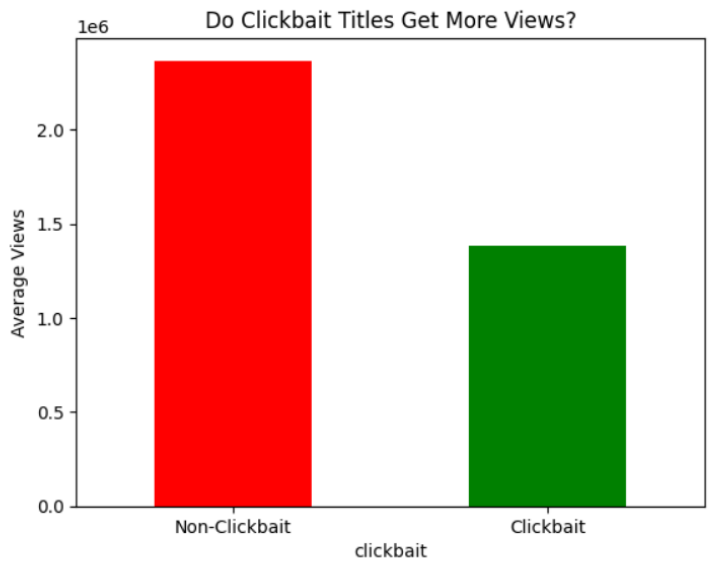


Clickbait Analysis: Impact of Attention-Grabbing Words on Views

This analysis investigates whether the presence of clickbait words in YouTube video titles correlates with higher average viewership. A predefined list of commonly used clickbait phrases such as *"shocking," "amazing,"* and *"you won't believe"* is used to flag videos containing these words. The dataset is then categorized into two groups: videos with and without clickbait titles. The average number of views for both categories is calculated and compared to determine if clickbait significantly influences audience engagement. A bar chart is generated to visualize the difference, helping to assess whether clickbait titles to increasing views.

```
"""Analyzes whether clickbait words in video titles lead to higher average views."""

# Clickbait Analysis: Do Certain Words Attract More Views?
clickbait_words = ['shocking', 'amazing', 'must watch', 'unbelievable', 'you won't believe', 'insane', 'epic']

# Count videos with clickbait words
df['clickbait'] = df['title'].apply(lambda x: any(word in x.lower() for word in clickbait_words))

# Compare average views for clickbait vs non-clickbait videos
clickbait_stats = df.groupby('clickbait')['views'].mean()

print(clickbait_stats)

# Plot comparison
clickbait_stats.plot(kind='bar', color=['red', 'green'])
plt.xticks([0,1], ['Non-Clickbait', 'Clickbait'], rotation=0)
plt.ylabel('Average Views')
plt.title('Do Clickbait Titles Get More Views?')
plt.show()
```

```
clickbait
False    2.368336e+06
True     1.386631e+06
Name: views, dtype: float64
```



5. Results and Discussion

The analysis provided several important insights into YouTube video performance:

1. **Engagement Metrics**: There is a clear positive correlation between the number of views and both likes and comments. This suggests that more popular videos tend to receive higher engagement in the form of likes and comments, indicating that viewers are more likely to interact with content they find appealing.
2. **Optimal Upload Time**: Videos uploaded during peak hours (late afternoon and evening) tended to perform better, potentially because they aligned with higher traffic periods on the platform. This insight can help content creators optimize their upload times to reach a larger audience.
3. **Video Categories**: Certain categories, such as music and gaming, garnered more views and engagement compared to others. Understanding the preferences within specific categories can help content creators tailor their content to meet audience demands.
4. **View Distribution**: A few videos had exceptionally high views, while the majority had significantly lower engagement. This highlights the "long tail" distribution typical of online content, where a small fraction of videos generate the majority of the views.

5. Conclusion

The analysis of YouTube video performance using data mining techniques has provided valuable insights into the factors influencing video success on the platform. By understanding the relationships between views, likes, dislikes, and comment counts, content creators can make more informed decisions regarding video creation and optimization. Additionally, the findings related to optimal upload times and category preferences offer actionable recommendations that can help increase video engagement and visibility.

As YouTube continues to evolve, content creators must leverage data-driven strategies to stay competitive. The application of data mining techniques in this report provides a solid foundation for further research and experimentation, as there are always new ways to enhance video performance through data analysis.

7. References

1. Data Science, Kaggle. "YouTube New Dataset". https://www.kaggle.com/datasnaek/youtube-new
2. Han, J., Kamber, M., & Pei, J. (2012). **Data Mining: Concepts and Techniques** (3rd ed.). Morgan Kaufmann Publishers.
3. Wes McKinney. (2018). **Python for Data Analysis** (2nd ed.). O'Reilly Media.
4. https://seaborn.pydata.org/ for Seaborn Documentation.
5. https://matplotlib.org/ for Matplotlib Documentation.