# Mini Project Report
# On
# ONLINE STREAMING EVENT DATA ANALYSIS using Python

*Submitted by*
*[All Group Members]*
Name:- 1.RANGIN BERA
2.SHRABANTINI GHOSH
Roll No.:- 1. 2206201
2. 2206213
School of Computer Engineering
KIIT - DU


KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

Deemed to be University U/S 3 of UGC Act, 1956

# Table of Contents

# Chapter 1

# Introduction

With the increasing popularity of online streaming platforms, analyzing viewership trends and engagement metrics has become essential for understanding audience behavior. This project aims to analyze online streaming data to identify trends in viewership, engagement patterns, and technical performance aspects such as buffering rates and bitrates.

## 1.2 Objective of the Project

The main objectives of this project are:

- **Viewership Trend Analysis**: Understand how viewership fluctuates over time
- **Engagement Metrics Analysis**: Assess audience interaction through likes, comments, and shares.
- **Technical Performance Analysis**: Examine the impact of bitrate on buffering rate.
- **Correlation Study**: Analyze relationships between engagement metrics and viewership

## 1.3 Tools & Technologies Used

To accomplish these objectives, we used the following tools and technologies:

- **Programming Language**: Python
- **Data Processing**: Pandas
- **Visualization**: Matplotlib & Seaborn
- **Time Series Analysis**: Datetime module for timestamp conversion

# Chapter 2

# Problem Statement & Objectives

## 2.1 Problem Statement

Online streaming platforms generate large volumes of data on viewership trends, user engagement, and streaming quality. However, analyzing this data to extract meaningful insights is a challenge. This project aims to develop a structured approach to extract, analyze, and visualize streaming data.

## 2.2 Objectives

- Provide meaningful visualizations to interpret key findings.
- Extract and process streaming data to identify viewership trends.
- Analyze engagement metrics (likes, comments, shares) to understand audience interaction.
- Evaluate technical performance metrics such as buffering rate and bitrate

# Methodology

The project followed a systematic methodology to analyze online streaming data using Python. The key steps involved were:

1. Data Collection:
   - A sample dataset containing information on viewership, engagement metrics (likes, comments, shares), and technical performance metrics (buffering rate, bitrate) was used.

2. Data Preprocessing:
   - The data was cleaned using the Pandas library, handling any missing or inconsistent values.
   - The 'Time' column was converted into a datetime format for time series analysis using pd.to_datetime().

3. Data Analysis and Visualization:
   - Viewership Trend Analysis: A line plot was generated using Seaborn to visualize changes in viewership over time.
   - Engagement Metrics Analysis: Histograms were plotted to observe the distribution of likes, comments, and shares.
   - Correlation Analysis: A heatmap was created to study correlations between engagement metrics and viewership.
   - Technical Performance Analysis: A scatter plot was used to explore the relationship between bitrate and buffering rate.

4. Insights and Interpretation:
   - Observations were drawn based on patterns in viewership, engagement levels, and performance metrics. Key insights were recorded to assist in understanding user behavior and stream performance.

# Dataset Description

The dataset consists of the following columns:

1. Time: Timestamp representing the time of data capture.
2. Viewers: Number of viewers at the given time.
3. Likes: Number of likes received.
4. Comments: Number of comments submitted by viewers.
5. Shares: Number of shares made by users.
6. Buffering Rate: The rate of buffering experienced during the stream.
7. Bitrate: The streaming bitrate at the time, measured in kbps.

## Example Live Streaming Data

| Time | Viewers | Likes | Comments | Shares | Buffering Rate | Bitrate |
|---|---|---|---|---|---|---|
| 24-03-2024 18:00 | 500 | 120 | 30 | 10 | 1.2 | 3000 |
| 24-03-2024 18:05 | 700 | 180 | 45 | 15 | 1 | 3500 |
| 24-03-2024 18:10 | 850 | 250 | 60 | 20 | 0.8 | 4000 |
| 24-03-2024 18:15 | 920 | 280 | 75 | 25 | 0.5 | 4200 |
| 24-03-2024 18:20 | 1100 | 320 | 90 | 30 | 0.7 | 4300 |
| 24-03-2024 18:25 | 1050 | 310 | 85 | 28 | 0.6 | 4100 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 24-03-2024 18:30 | 980 | 290 | 80 | 27 | 0.9 | 3900 |
| 24-03-2024 18:35 | 1200 | 350 | 95 | 35 | 0.4 | 4500 |
| 24-03-2024 18:40 | 1300 | 400 | 110 | 40 | 0.3 | 4600 |
| 24-03-2024 18:45 | 1250 | 390 | 105 | 38 | 0.5 | 4550 |

# Chapter 3

# Python Packages Used Details

| Package | Function Used | Explanation |
|---|---|---|
| Pandas | read_csv(), to_datetime() | Reads and processes data |
| Matplotlib | plt.plot(), plt.hist() | Creates visualizations |
| Seaborn | sns.lineplot(), sns.heatmap() | Enhances visualization aesthetics |

| | | |
|---|---|---|
| Datetime | Pd.to_datetime | Converts timestamps for time series analysis |

# Chapter 4

## Source Code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load dataset
file_path = "/content/Online_Streaming_Data
(2).csv"
# Read the data from the CSV file into a
pandas DataFrame called 'data'
Data=pd.read_csv("/content/Online_Streaming_D
ata (2).csv")
```

## #-----Convert timestamp column to datetime----

```
data['Time'] = pd.to_datetime(data['Time'])
```

## # ---------- Viewership Trend Analysis ----------

```
plt.figure(figsize=(12, 6))
sns.lineplot(x='Time', y='Viewers', data=data,
color='blue')
plt.title('Viewership Trend Over Time')
plt.xlabel('Time')
plt.ylabel('Number of Viewers')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

## # ---------- Engagement Metrics Analysis ----------

```
engagement_metrics = ['Likes', 'Comments',
'Shares']
data[engagement_metrics].hist(figsize=(12, 6),
bins=30)
plt.suptitle('Engagement Metrics Distribution')
plt.show()
```

```
# Correlation heatmap
plt.figure(figsize=(8, 5))
sns.heatmap(data[engagement_metrics +
['Viewers']].corr(), annot=True, cmap='coolwarm',
fmt='.2f')
plt.title('Correlation Between Engagement Metrics')
plt.show()
```

## # ---------- Technical Performance Analysis ---------

```
if 'Buffering Rate' in data.columns and 'Bitrate'
in data.columns:
    plt.figure(figsize=(8, 5))
    sns.scatterplot(x='Bitrate', y='Buffering
Rate', data=data)
```

```
    plt.title('Bitrate vs Buffering Rate')
    plt.xlabel('Bitrate')
    plt.ylabel('Buffering Rate')
    plt.grid(True)
    plt.show()
```

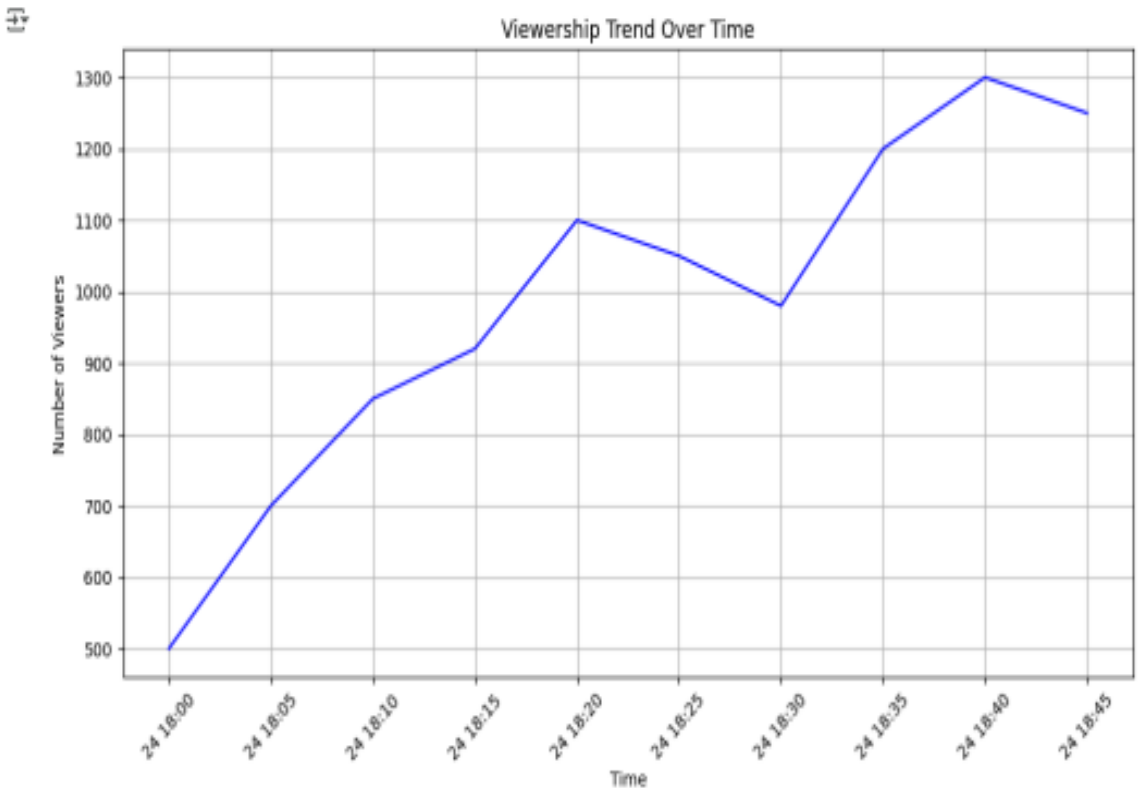
```
# Summary
print("\nKey Observations:")
print("- Peak viewership trends identified.")
print("- Engagement metrics analyzed for audience
interaction patterns.")
print("- Technical performance metrics assessed for
stream quality insights.")
```

# Chapter 5

# Implementation Results

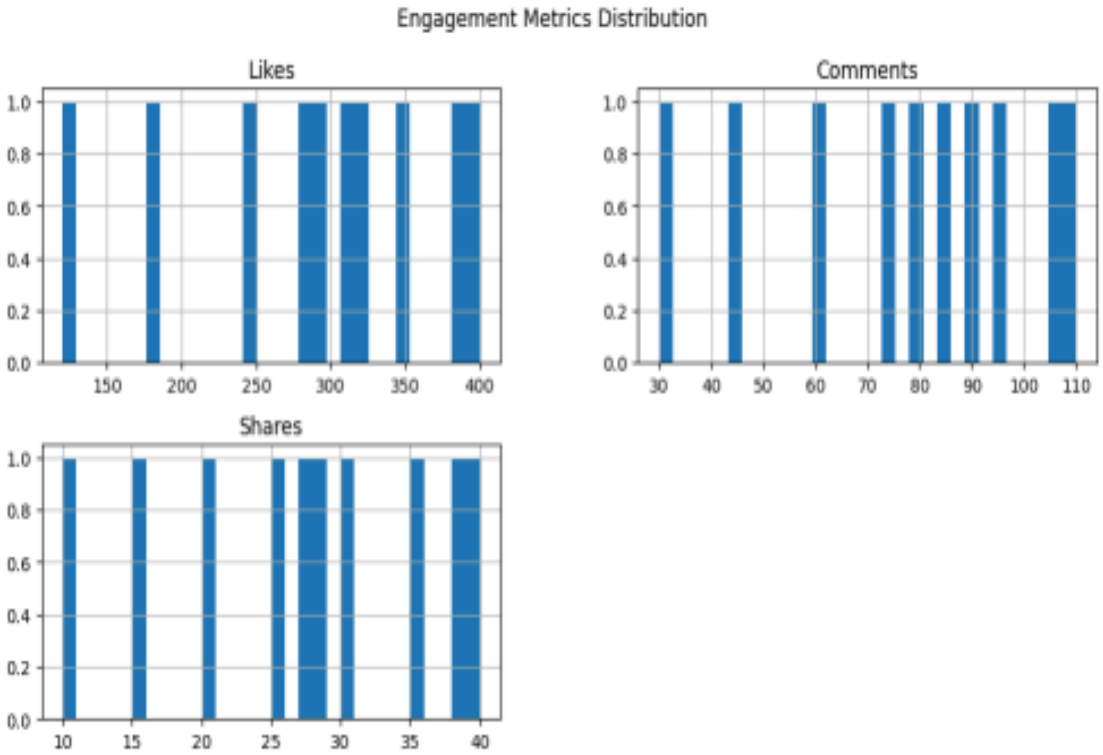
## 5.1 Viewership Trend Analysis

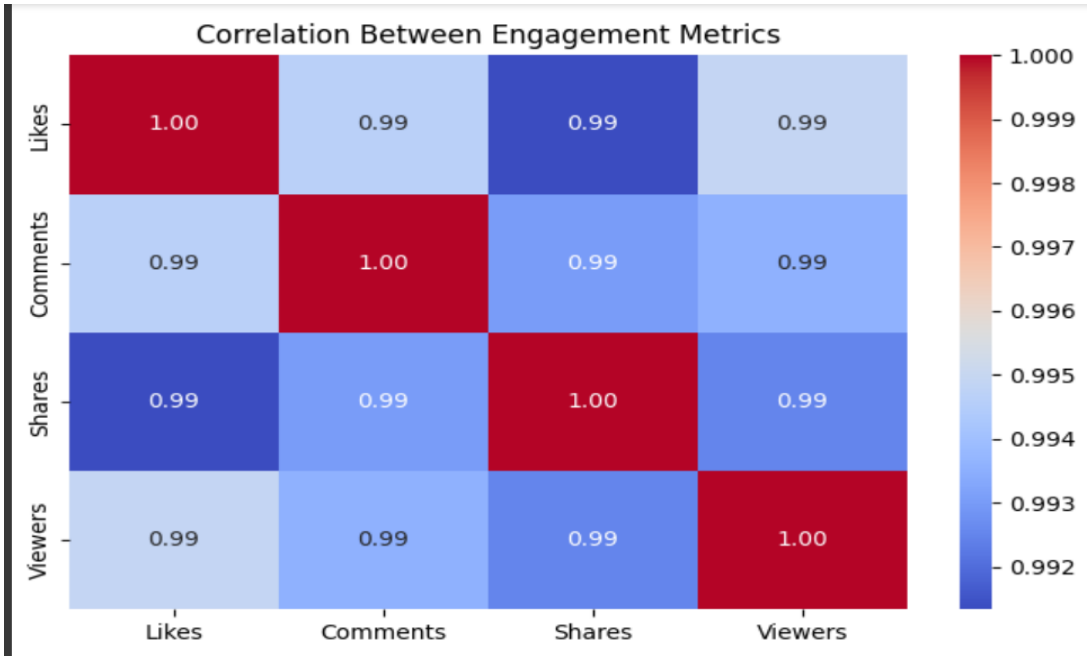
- A line graph was generated showing the trend of viewership over time.
- Peaks in the graph indicate high activity periods, suggesting increased audience engagement during specific time frames.



# 5.2 Engagement Metrics Analysis

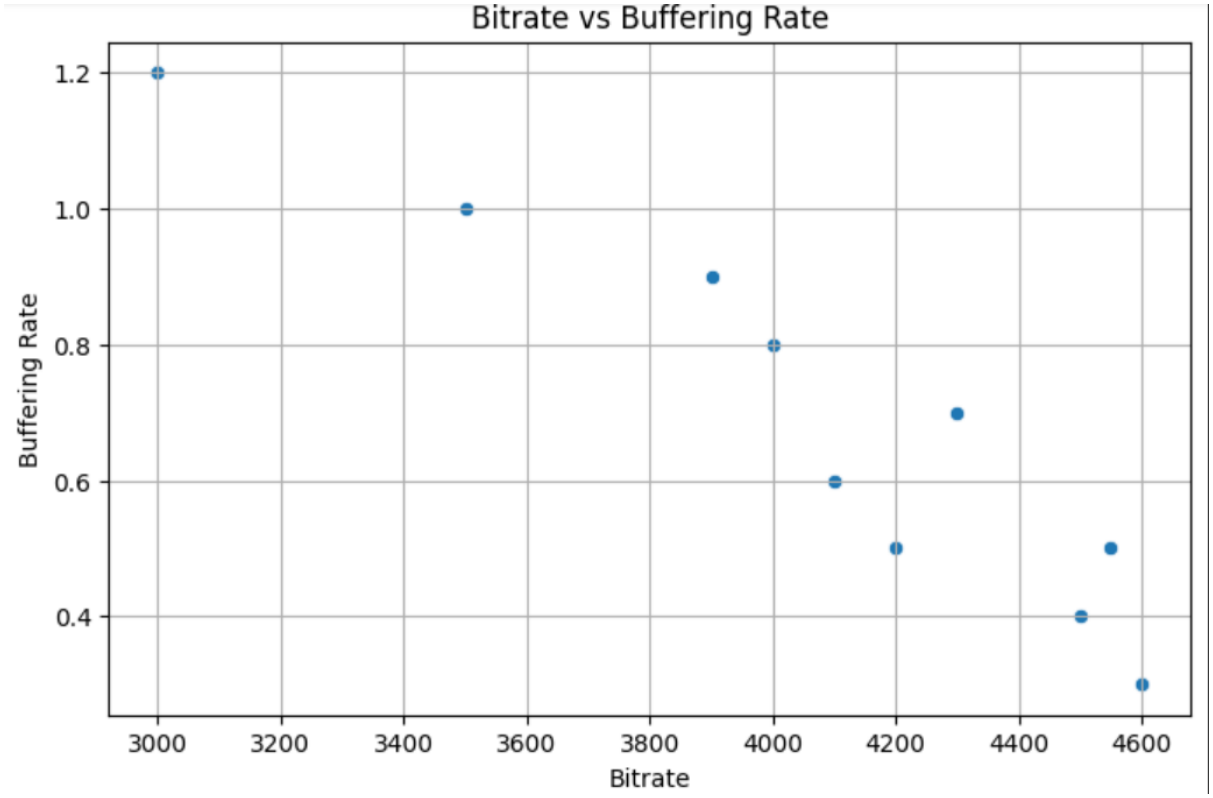
- Histogram visualizations were created to understand the distribution of likes, comments, and shares**.**
- The correlation heatmap indicated the relationship between engagement metrics and viewership trends.






# 5.3 Technical Performance Analysis

- A scatter plot between **bitrate and buffering rate** showed how stream quality is affected.
- Insights were drawn on the ideal bitrate required to minimize buffering



## 5.4 Key Insights

```
Key Observations:
- Peak viewership trends identified.
- Engagement metrics analyzed for audience interaction patterns.
- Technical performance metrics assessed for stream quality insights.
```

# Chapter 6

# Conclusion

This project successfully analyzed online streaming data using Python to derive meaningful insights into viewership trends, engagement metrics, and technical performance.

- *Viewership Insights*: The analysis identified peak streaming hours with maximum audience engagement.
- *Engagement Insights*: A strong correlation was found between likes, comments, and shares, indicating active audience participation during certain periods.
- *Technical Performance Insights*: Lower buffering rates were observed at optimized bitrates, suggesting opportunities for improving stream quality.

The results from this study can help streaming platforms enhance the user experience, improve content delivery strategies, and make data-driven decisions for better audience engagement. Further analysis with

larger datasets and additional metrics could provide deeper insights for future studies.

# References

1. Pandas Documentation: https://pandas.pydata.org/

2. Matplotlib Documentation: https://matplotlib.org/
3. Seaborn Documentation: https://seaborn.pydata.org/
4. Online streaming data analysis techniques and research articles.