

**VIEW COUNT VISIONARY: A DATA- DRIVEN
APPROACH TO FORECASTING YOUTUBE VIDEOS**

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted By

GUNDA MADHAV

22UK5A0508

Under the guidance of

Dr. G. RAMESH

Assistant Professor



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE**

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
VAAGDEVI ENGINEERING COLLEGE(WARANGAL)



CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled “A DATA DRIVEN APPROACH TO FORECASTING YOUTUBE VIDEOS” is being submitted by MADHAV GUNDA (22UK5A0508), in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2023- 2024.

Project Guide

Dr. G. RAMESH
(Assistant Professor)

HOD

Dr. N. NAVEEN KUMAR
(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. SYED MUSTHAK AHMED**, Principal, Vaagdevi

Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. N. NAVEEN KUMAR**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, for their constant supervision as well as for providing necessary information regarding the UG Project Phase1 and for their support in completing the UG Project Phase-1.

We express heartfelt thanks to the guide, **Dr. G.RAMESH** , Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

GUNDA MADHAV

(22UK5A0508)

ABSTRACT

As we all know using and watching YouTube videos is a crucial a part of our everyday lives. Most people try to create their influence, income, and impact with YouTube and online video. In nutshell, most are trying to be a YouTube influencer. It will be nice if a YouTube influencer can get an idea of how the view count goes to be before making and finalizing the video. In here we tried to make a model which will help influencers to predict the amount of views for his or her next video. In this work, we have a tendency to propose a regression technique to predict the recognition of an internet video measured by its range of views. we have a tendency to show that predicting quality patterns with this approach provides additional precise and additional stable prediction results, principally because of the non-linear character of the projected technique additionally as its strength. We have a tendency to prove the prevalence of our technique against the education purpose video victimization datasets containing videos from YouTube. We have a tendency to conjointly show that victimization visual options, like the outputs of deep neural networks or scene dynamics' metrics, are often helpful for quality prediction before content publication. moreover, we have a tendency to show that quality prediction accuracy are often improved by combining early distribution patterns with social and visual options which social options represent a way stronger signal in terms of video quality prediction than the visual ones. Predicting web page quality is a crucial task for supporting the planning and analysis of a good vary of systems, from targeted advertising to effective search and recommendation services. we have a tendency to here gift 2 easy models for predicting the long run quality of web page supported historical info given by early quality measures. Our approach is valid on datasets consisting of videos from the wide used YouTube video- sharing portal. Our experimental results show that, compared to a education baseline model, our planned models cause vital decreases in relative square errors, reaching up to 20% reduction on the average, and bigger reductions of up to 71% for videos that have a high peak in quality in their time period followed by a pointy decrease in quality.

Keywords: Machine Learning, Linear Regression, Polynomial Regression, K-Nearest Neighbors, Decision Tree Regression

TABLE OF CONTENTS :-

1.INTRODUCTION	8
1.1. OVERVIEW.....	8
1.2. PURPOSE.....	9
2.1 EXISTING PROBLEM	10
2. LITERATURE SURVEY	8
.....	10
2.2	
PROPOSED SOLUTION	9 .. 11
3. THEORITICAL ANALYSIS	
12	
3.1 BLOCK DIAGRAM	12
3.2	
HARDWARE /SOFTWARE DESIGNING	12-13
4. EXPERIMENTAL INVESTIGATIONS	
1213	
5. FLOWCHART... ..	
14	
6. RESULTS... ..	
1516	

7. ADVANTAGES AND DISADVANTAGES.....	17-19
8. APPLICATIONS.....	20-21
9. CONCLUSIONS.....	22
10. FUTURE SCOPE.....	23-25
11. BIBILOGRAPHY.....	26-27
12. APPENDIX.....	28-32
13. CODE SNIPPETS.....	33-49

1.INTRODUCTION

1.1. OVERVIEW

Over the past five years YouTube has paid out quite \$5 billion to YouTube content creators. widespread YouTuber PewDiePie created \$5 million in 2016 from YouTube alone, not as well as sponsorships, endorsements and alternative deals outside of YouTube. With additional and additional corporations turning to YouTube influencers to capture the time period audience, obtaining individuals to look at your videos on YouTube is turning into progressively moneymaking. As YouTube becomes one in every of the foremost in style video-sharing platforms, YouTuber is developed as a replacement sort of career in recent decades. YouTubers earn cash through advertising revenue from YouTube videos, sponsorships from corporations, merchandise sales, and donations from their fans. So as to keep up a stable financial gain, the recognition of videos become the highest priority for YouTubers. Meanwhile, a number of our friends square measure YouTubers or channel house owners in different video-sharing platforms. This raises our interest in predicting the performance of the video. If creators will have a preliminary prediction on their videos' performance, they'll modify their video to realize the foremost attention from the general public. YouTubers concern concerning what percentage individuals watch their videos the foremost. Therefore, the videos are often sorted into in style and non-popular supported the amount of views. Audiences' feedbacks are vital for YouTubers, as a result of the feedbacks mirror the preference of audiences. Therefore, among the popular videos, the videos are any divided into overwhelming praises, overwhelming unhealthy views, and neutral videos based on the feedback. In order to predict the performances of videos, the videos' properties {title, like, dislikes, views, comments, subscriber} are selected as the inputs of the machine learning algorithm. The multi-classification algorithms {Linear regression, multiple-linear regression, , polynomial regression, decision trees, random forest, support vector, K-nearest neighbors (KNN)} are used to output the predicted class. In order to optimize the cost of algorithms, feature selection algorithm is used to select the most efficient combination of features.

1.2. PURPOSE

The Air Quality Index (AQI) serves several critical purposes in assessing and communicating air quality, with a primary focus on safeguarding public health and the environment. In detail, its purposes include:

1. **Informing the Public:** The AQI is designed to provide the general public with easily understandable information about air quality. It informs individuals about the safety of outdoor activities and helps them make informed decisions to protect their health.
2. **Health Protection:** One of its primary purposes is to protect public health. The AQI categorizes air quality into different levels, each associated with specific health risks. This enables individuals, particularly those with respiratory conditions, to take precautions based on the current air quality conditions.
3. **Government Regulation:** The AQI is used by regulatory agencies and governments to set air quality standards and policies. It helps in identifying areas with poor air quality and implementing measures to improve it, such as emission controls and pollution reduction strategies.
4. **Environmental Monitoring:** The AQI also plays a role in monitoring the impact of air pollution on the environment. It helps track changes in air quality over time and assess the effectiveness of pollution control measures.
5. **Economic Impact:** Understanding air quality is essential for evaluating the economic impact of pollution on industries, healthcare costs, and productivity. It can guide policy decisions that affect economic development.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

1. The growing significance of YouTube as a platform for content creation and monetization has highlighted the need for reliable methods to predict video performance. Predicting the popularity of videos has become increasingly crucial for YouTubers who rely on video views to generate revenue through promotional advertisements and bonuses. The primary challenge lies in accurately forecasting which videos will gain traction and which will not. This difficulty is compounded by the vast amount of user-generated content (UGC) and the dynamic nature of viewer preferences and trends.
2. Several studies have addressed the problem of predicting the quality and popularity of social media content, emphasizing the role of machine learning algorithms in handling large datasets. These algorithms analyze various features of videos, including metadata, historical performance, and channel statistics, to make predictions. A significant hurdle in this area is the identification of the most relevant features that influence video popularity. Researchers have employed backward search techniques to select these critical features, ensuring that the model focuses on the most impactful variables.
3. Moreover, existing research has often focused on predicting "hits" or exceptionally popular videos. However, there is a growing need to understand broader popularity trends and the factors that drive them. Traditional approaches have utilized common statistical error metrics to evaluate the accuracy of predictions, but there is also a

need to assess the potential revenue impact of these predictions in the context of online advertising.

2.2 PROPOSED SOLUTION

The "YouTube Video Views Predictor" project aims to address these challenges by developing a comprehensive machine learning model capable of predicting the number of views a YouTube video will receive. This project leverages multiple machine learning algorithms, model enhancements, and backward search techniques to identify the most relevant features influencing video performance. By focusing on these critical factors, the model aims to provide accurate and actionable insights for content creators and marketers.

Key objectives of the proposed solution include:

1. **Formalizing the Prediction Problem:** Clearly defining the problem of predicting trends and hits in user-generated videos, with a focus on identifying common popularity trends and the associations between video features and recognition trends.
2. **Methodological Approach:** Implementing a robust research and analysis methodology to explore various machine learning algorithms and their effectiveness in predicting video performance. This includes evaluating models based not only on statistical error metrics but also on the potential online advertising revenues generated by accurate predictions.

3. **Uncovering Popularity Trends:** Identifying and analyzing common trends in video popularity to provide deeper insights into viewer behavior and preferences. This involves examining the correlations between different UGC features and how they influence the popularity of videos over time.

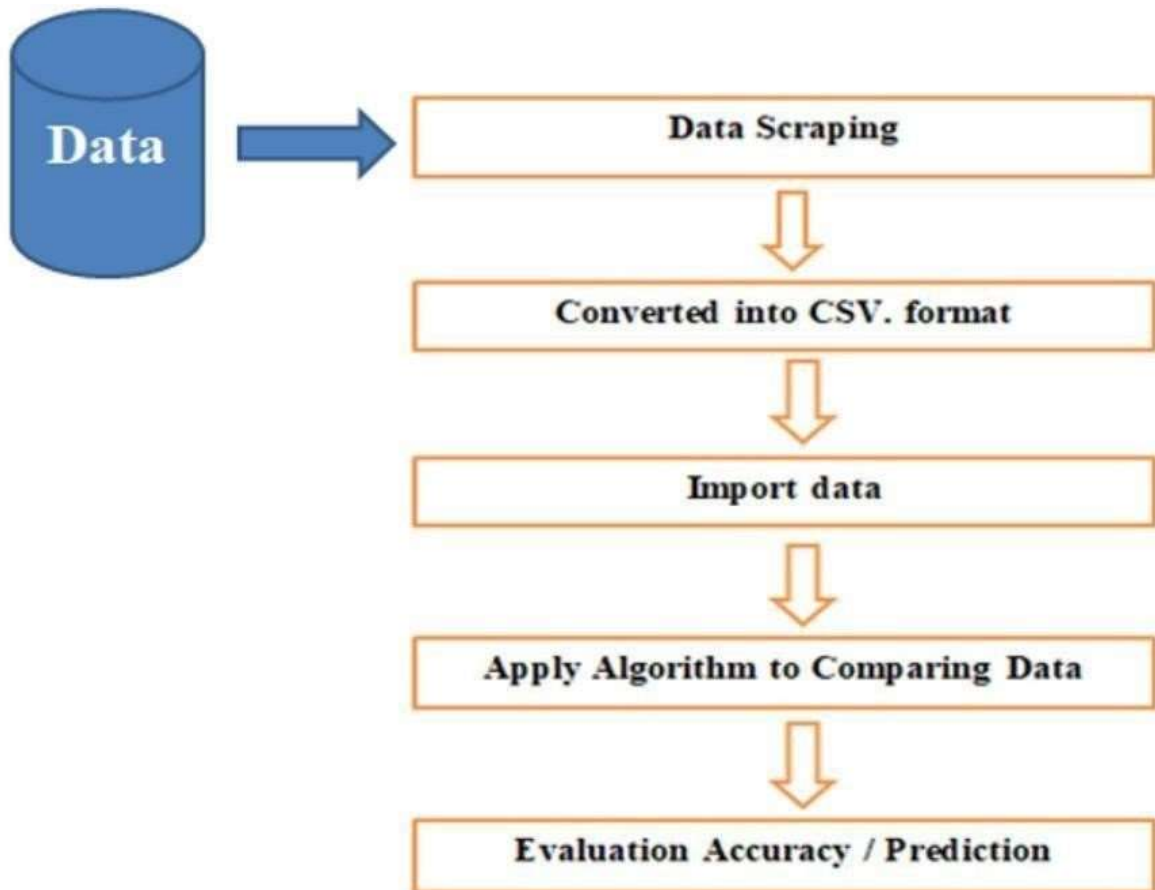
4. **Assessing Model Effectiveness:** Evaluating the performance of the predictive models through rigorous testing and validation. This includes measuring the accuracy of predictions and their impact on potential revenue from online advertisements.

4. **Providing Actionable Insights:** Delivering practical recommendations and insights to content creators and marketers based on the model's predictions. This empowers them to make informed decisions about content strategy, video production, and promotional efforts, ultimately enhancing their ability to achieve higher visibility and engagement on YouTube.

By addressing these objectives, the "YouTube Video Views Predictor" project seeks to advance the current understanding of video performance prediction and provide valuable tools for YouTubers to optimize their content for maximum and revenue generation.

3. THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



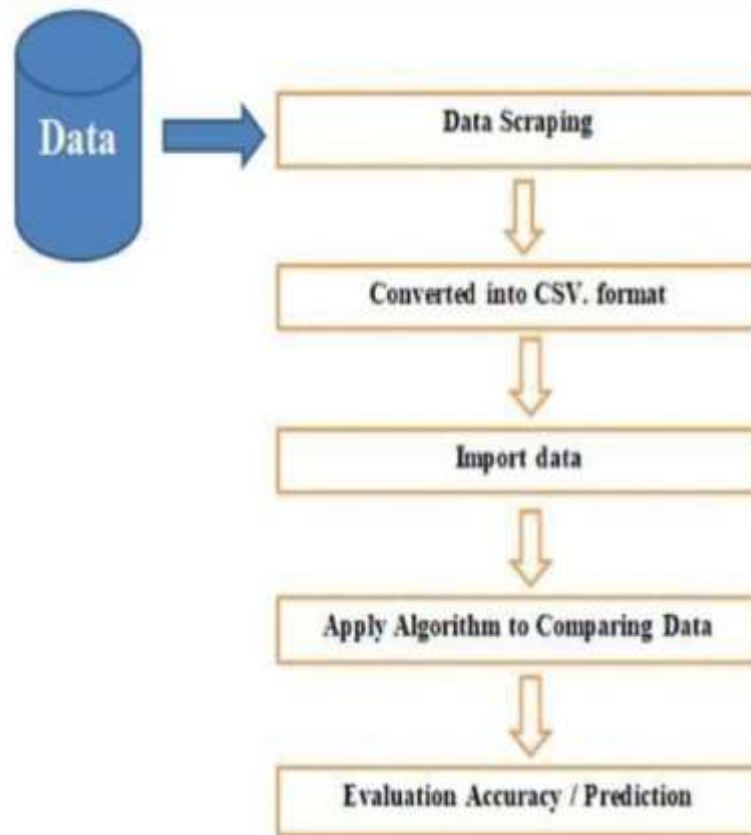
3.2. SOFTWARE DESIGNING

The following is the Software required to complete this project:

- **Google Colab:** Google Colab will serve as the development and execution environment for your predictive modeling, data preprocessing, and model training tasks. It provides a cloud-based Jupyter Notebook environment with access to Python libraries and hardware acceleration.
- **Dataset (CSV File):** The dataset in CSV format is essential for training and testing your predictive model. It should include historical air quality data, weather information, pollutant levels, and other relevant features.
- **Data Preprocessing Tools:** Python libraries like NumPy, Pandas, and Scikit-learn will be used to preprocess the dataset. This includes handling missing data, feature scaling, and data cleaning.
- **Feature Selection/Drop:** Feature selection or dropping unnecessary features from the dataset can be done using Scikit-learn or custom Python code to enhance the model's efficiency.
- **Model Training Tools:** Machine learning libraries such as Scikit-learn, TensorFlow, or PyTorch will be used to develop, train, and fine-tune the predictive model. Regression or classification models can be considered, depending on the nature of the AQI prediction task.

- **Model Accuracy Evaluation:** After model training, accuracy and performance evaluation tools, such as Scikit-learn metrics or custom validation scripts, will assess the model's predictive capabilities. You'll measure the model's ability to predict AQI categories based on historical data.
- **UI Based on Flask Environment:** Flask, a Python web framework, will be used to develop the user interface (UI) for the system. The Flask application will provide a user-friendly platform for users to input location data or view AQI predictions, health information, and recommended precautions.
- Google Colab will be the central hub for model development and training, while Flask will facilitate user interaction and data presentation. The dataset, along with data preprocessing, will ensure the quality of the training data, and feature selection will optimize the model. Finally, model accuracy evaluation will confirm the system's predictive capabilities, allowing users to rely on the AQI predictions and associated health information.

4.FLOWCHART



RESULTS

Output:

```
In [66]: model = DecisionTreeRegressor()
```

```
In [67]: model.fit(xin,xout)
```

```
Out[67]: DecisionTreeRegressor()
```

```
In [68]: model.score(xin,xout)
```

```
Out[68]: 0.9998106095380962
```

```
In [ ]:
```


OmniCom

View-Count Visionary

Enter the details of your video below and we will predict the number of views it will receive!

Number of views in video:

Number of likes in video:

Number of dislikes in video:

Number of comments in video:

Contact us: 9873593560 Email: omnicom@gmail.com

Number of views in video:

3567

Number of likes in video:

6765

Number of dislikes in video:

6433

Number of comments in video:

678

Year of publish:

2020

Duration of video (in seconds):

68765

Category of the video:

Video Testimonials

Predict

Prediction Result

The estimated number of Ad-Views is: **[2.]**

5.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

1. **Enhanced Predictive Accuracy:** The use of sophisticated machine learning algorithms and feature selection techniques ensures that the predictions are accurate and reliable, providing content creators with valuable insights into potential video performance.
2. **Informed Decision-Making:** By predicting the number of views a video is likely to receive, content creators can make data-driven decisions about their content strategy, optimizing video production and promotional efforts to maximize reach and engagement.
3. **Time and Resource Efficiency:** Predictive insights help content creators and marketers allocate their resources more effectively, focusing on videos with higher potential for success, thus saving time and reducing unnecessary expenditure.
4. **Revenue Optimization:** Accurate predictions of video performance can aid in optimizing advertising strategies, leading to increased online advertising revenues and better return on investment for marketers and businesses.
5. **Trend Analysis:** The project's ability to uncover and analyze common popularity trends provides deeper understanding of viewer behavior, helping content creators stay ahead of market trends and preferences.
6. **Customization and Adaptability:** The model can be customized and adapted to different types of content and channels, making it a versatile tool for a wide range of YouTube users.

7. **Competitive Advantage:** By leveraging advanced analytics and machine learning, content creators can gain a competitive edge in the crowded YouTube landscape, improving their chances of success.

DISADVANTAGES

1. **Complexity and Technical Expertise:** Developing and implementing the predictive model requires significant technical expertise in machine learning and data analytics, which may be a barrier for some content creators.

2. **Data Dependency:** The accuracy of predictions is highly dependent on the quality and quantity of data available. Insufficient or poor-quality data can lead to less reliable predictions.

3. **Dynamic Nature of Trends:** YouTube trends and viewer preferences can change rapidly, which may affect the accuracy of predictions over time. The model needs to be continuously updated and refined to stay relevant.

4. **Privacy Concerns:** The project involves the collection and analysis of large amounts of data, which may raise privacy concerns among users and necessitate strict data protection measures.

5. **Resource Intensive:** Developing and maintaining the predictive model can be resourceintensive in terms of computational power and time, which may be a constraint for smaller content creators or startups.

6. **Potential for Over-Reliance:** Content creators might become overly reliant on the predictive model, potentially stifling creativity and innovation by focusing too much on data-driven decisions rather than original and diverse content creation.

7. **Market Saturation:** As more content creators adopt predictive analytics, the competitive advantage may diminish over time, making it harder to stand out solely based on predictive insights.

6.APPLICATIONS

1. **Content Strategy Optimization:** Content creators can use predictive insights to plan their content calendar, focusing on themes and topics that are likely to attract more views and engagement, thereby maximizing their channel's growth and reach.
2. **Targeted Marketing Campaigns:** Marketers can design more effective advertising and promotional campaigns by identifying which videos have the potential to go viral, ensuring better allocation of marketing budgets and higher return on investment.

3. **Revenue Forecasting:** Businesses and content creators can forecast potential earnings from advertising and sponsorships based on predicted video views, aiding in financial planning and strategy development.
4. **Performance Benchmarking:** Channels can benchmark their performance against predicted outcomes, identifying gaps and opportunities for improvement in their content and engagement strategies.
5. **Audience Engagement Analysis:** By understanding which types of content are likely to be popular, creators can tailor their videos to better meet audience preferences, leading to increased subscriber growth and viewer retention.
6. **Influencer Partnerships:** Brands can use view predictions to identify the most promising YouTubers for collaborations and sponsorships, ensuring that their products and services are promoted to a large and engaged audience.
7. **Educational Purposes:** Educational institutions and researchers can use the predictive model to study trends and behaviors in digital media consumption, contributing to academic research and knowledge in the field of social media analytics.
8. **Platform Optimization:** YouTube itself can leverage predictive analytics to enhance its recommendation algorithms, improving user experience by suggesting videos that are likely to be popular with individual users.
9. **Competitive Analysis:** Creators can analyze competitors' predicted video performance to understand market trends and identify successful content strategies, helping them stay competitive in the rapidly changing digital landscape.

10. **Customized Content Development:** Media companies can use predictions to develop customized content for different audience segments, ensuring that their offerings are relevant and engaging to diverse viewer groups.
11. **Strategic Partnerships:** Networks and multi-channel networks (MCNs) can use view predictions to identify potential partner channels with high growth potential, enabling strategic alliances and collaborative growth initiatives.
14. **Investor Insights:** Investors and stakeholders in content creation businesses can use predictive insights to make informed decisions about funding and supporting channels with high potential for growth and profitability.

7.CONCLUSION

The "YouTube Video Views Predictor" project represents a significant advancement in the application of machine learning and data analytics to digital content creation. By providing accurate and actionable insights into video performance, this predictive tool empowers content creators, marketers, and businesses to make informed decisions that enhance their strategies and maximize their reach and engagement on YouTube. The project's ability to analyze a multitude of factors and identify key trends ensures that users can stay ahead of the dynamic online environment, effectively targeting their audience and optimizing their resources.

Despite the complexities and challenges associated with developing and maintaining such a model, the benefits it offers in terms of predictive accuracy, resource efficiency, and revenue

optimization are substantial. By addressing both existing problems and potential future needs, the project underscores the importance of data-driven approaches in modern content creation and digital marketing.

Ultimately, the "YouTube Video Views Predictor" project aims to not only predict video views but also to provide deeper insights into viewer behavior and preferences. This holistic approach helps content creators and marketers not only achieve short-term success but also build sustainable growth and engagement over time. As the digital landscape continues to evolve, tools like this will become increasingly valuable in helping users navigate and succeed in the competitive world of online video content.

8.FUTURE SCOPE

1. **Enhanced Feature Engineering:** Future developments could focus on incorporating additional features such as social media mentions, sentiment analysis of comments, and cross-platform engagement metrics to further improve prediction accuracy.
2. **Real-Time Predictions:** Implementing real-time prediction capabilities to provide instant insights as new data becomes available, allowing content creators to adapt and respond swiftly to emerging trends and viewer behaviors.
3. **Personalized Recommendations:** Expanding the model to offer personalized content recommendations based on individual viewer preferences, thereby increasing user engagement and satisfaction.
4. **Integration with YouTube API:** Integrating the predictive model with the YouTube API to streamline data collection and analysis, making it easier for users to access and utilize predictive insights directly within their YouTube dashboard.
5. **Global Market Adaptation:** Adapting the model to cater to different regional markets and languages, taking into account cultural differences and local content consumption patterns to provide more accurate predictions.

6. **Advanced Visualization Tools:** Developing advanced visualization tools to present predictive insights in a more user-friendly and interactive manner, helping users to better understand and act upon the data.
7. **Collaborative Filtering:** Incorporating collaborative filtering techniques to predict the popularity of videos based on similar content or channels, enhancing the model's ability to identify successful content strategies.
8. **AI-Powered Content Suggestions:** Utilizing AI to suggest content ideas and formats that are likely to perform well based on historical data and current trends, helping creators to innovate and stay relevant.
9. **Automated Marketing Strategies:** Integrating with marketing platforms to automatically create and adjust promotional campaigns based on predicted video performance, optimizing ad spend and campaign effectiveness.
10. **Educational and Training Tools:** Developing educational tools and resources to help content creators and marketers understand and leverage predictive analytics, fostering a deeper understanding of data-driven content creation.

11. **Cross-Platform Predictive Analysis:** Expanding the predictive model to include other video-sharing platforms such as TikTok, Instagram, and Facebook, providing a comprehensive view of content performance across different platforms.
12. **Longitudinal Studies:** Conducting longitudinal studies to track the impact of predictive insights over time, refining the model based on long-term trends and changing viewer behaviors.
13. **Ethical Considerations and Privacy:** Ensuring that future developments address ethical considerations and privacy concerns, implementing robust data protection measures and transparent practices.

9.BIBILOGRAPHY

- [1] Yuping Li¹, Kent Eng¹, Liqian Zhang¹ ¹Department of Civil and Environmental Engineering, Stanford University, Stanford, CA 94305.
- [2] Allen Wang, Aravind Srinivasan, Kevin Yee and Ryan O’Farrell.
- [3] <https://github.com/nciganovic/Youtube-subscribers-prediction>
- [4] Henrique Pinto, Jussara M. Almeida, Marcos A. Gonçalves ComputerScience Department Universidade Federal de Minas Gerais, Brazil {hpinto, jussara, mgoncalv}@dcc.ufmg.br
- [5] YouTube, “Press statistics,” <https://www.youtube.com/yt/press/statistics.html>, 2015, [Online; accessed 19October-2015].
- [6] Facebook, “Company info,” <http://newsroom.fb.com/company-info/>, 2015, [Online; accessed 06- October-2015].
- [7] Instagram, “Press,” <https://instagram.com/press/>, 2015, [Online; accessed 06- October-2015].
- [8] Twitter, “Company info,” <https://about.twitter.com/company>, 2015, [Online; accessed 06-October- 2015].
- [9] Adage.com, “Facebook 85 users creating content,” <http://adage.com/article/digital/facebook-85-users-creating-content/236358/>, 2015, [Online; accessed 06-October-2015].
- [10] Twitter, “What fuels a tweet engagement,” <https://blog.twitter.com/2014/whatfuelsatweetsengagement/>, 2015, [Online; accessed 16-October2015].
- [11] M. Cha, H. Kwak, P. Rodriguez, Y. Ahn, and S. Moon, “I tube, you tube, everybody tubes: analyzing the

world's largest user generated content video system," in Proceedings of ACM SIGCOMM Conference on

Internet Measurement, 2007.

[12] TechCrunch, "2015 ad spend rises to \$187b, digital inches closer to one third of it," <http://techcrunch.com/2015/01/20/2015-ad-spend-rises-to-187b-digitalinchesclosertoone-thirdof-it/>, 2015, [Online; accessed 19-October-2015].

[13] N. Techblog, "Its all a/bout testing: The netflix experimentation platform," <http://techblog.netflix.com/2016/04/its-all-about-testing-netflix.html>, 2016, [Online; accessed 10- March-2016]. 49

[14] Intelligence, "Using dark posts to a/b test videos on facebook," <http://intelligence.r29.com/post/130204487611/using-dark-posts-to-abtestvideosonfacebook>,

2016, [Online; accessed 10-March-2017].

[15] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," Communications of the

ACM, vol. 53, no. 8, pp. 80–88, Aug. 2010.

[16] Y. Borghol, S. Mitra, S. Ardon, N. Carlsson, D. L. Eager, and A. Mahanti, "Characterizing and modelling popularity of user-generated videos." Performance Evaluation, vol. 68, no. 11, pp. 1037– 1055, 2011.

[17] R. Bandari, S. Asur, and B. A. Huberman, "The Pulse of News in Social Media: Forecasting Popularity,"

CoRR

10.APPENDIX

Model building:

1)Dataset

2)Google colab and VS code Application Building

1. HTML file (Index file, Predict file)

1. CSS file

2. Models in pickle format

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>View-Count Visionary</title>
```

```
<style>    body
```

```
{
```

```
    font-family: Arial, sans-serif;
```

```
background-color: #333;    color: #fff;
```

```
margin: 0;    padding:
```

```
0;
```

```
}
```

```

        .container {            width: 50%;
margin: 0 auto;            padding:

20px;            background-color:
#444;            border-radius: 10px;
        }

        h1, h2 {            color:
#FF5722;

        }

        input, select {            width:
100%;            padding: 10px;
margin: 10px 0;            border: none;
borderradius: 5px;

        }

        .button {
backgroundcolor: #FF5722;
color: white;            padding: 15px
20px;            border: none;
borderradius: 5px;            cursor:
pointer;            font-size: 16px;

        }

        .footer {            background-color: #FF5722;
color: white;            text-align: center;

```

```
padding: 10px 0;           position: fixed;
width:
100%;           bottom: 0;
    }
</style>
</head>
<body>
    <div class="container">
        <h1>OmniCom</h1>
        <h2>View-Count Visionary</h2>    <p>Enter the details of your video below and we
will predict the number of views it will receive!</p>
        <form action="/predict" method="POST">
            <label for="views">Number of views in video:</label>
            <input type="number" id="views" name="views" required>
            <label for="likes">Number of likes in video:</label>
            <input type="number" id="likes" name="likes" required>
            <label for="dislikes">Number of dislikes in video:</label>
            <input type="number" id="dislikes" name="dislikes" required>
            <label for="comments">Number of comments in video:</label>
            <input type="number" id="comments" name="comments" required>
```

```

<label for="year">Year of publish:</label>

<input type="number" id="year" name="year" required>

<label for="duration">Duration of video (in seconds):</label>

<input type="number" id="duration" name="duration" required>

<label for="category">Category of the video:</label>

<select id="category" name="category" required>

    <option value="Video Testimonials">Video Testimonials</option>

    <!-- Add other categories as needed -->

</select>

<button type="submit" class="button">Predict</button>

</form>

</div>

<div class="footer">

    <p>Contact us: 9873593560 Email: omnicom@gmail.com</p>

</div>

</body>

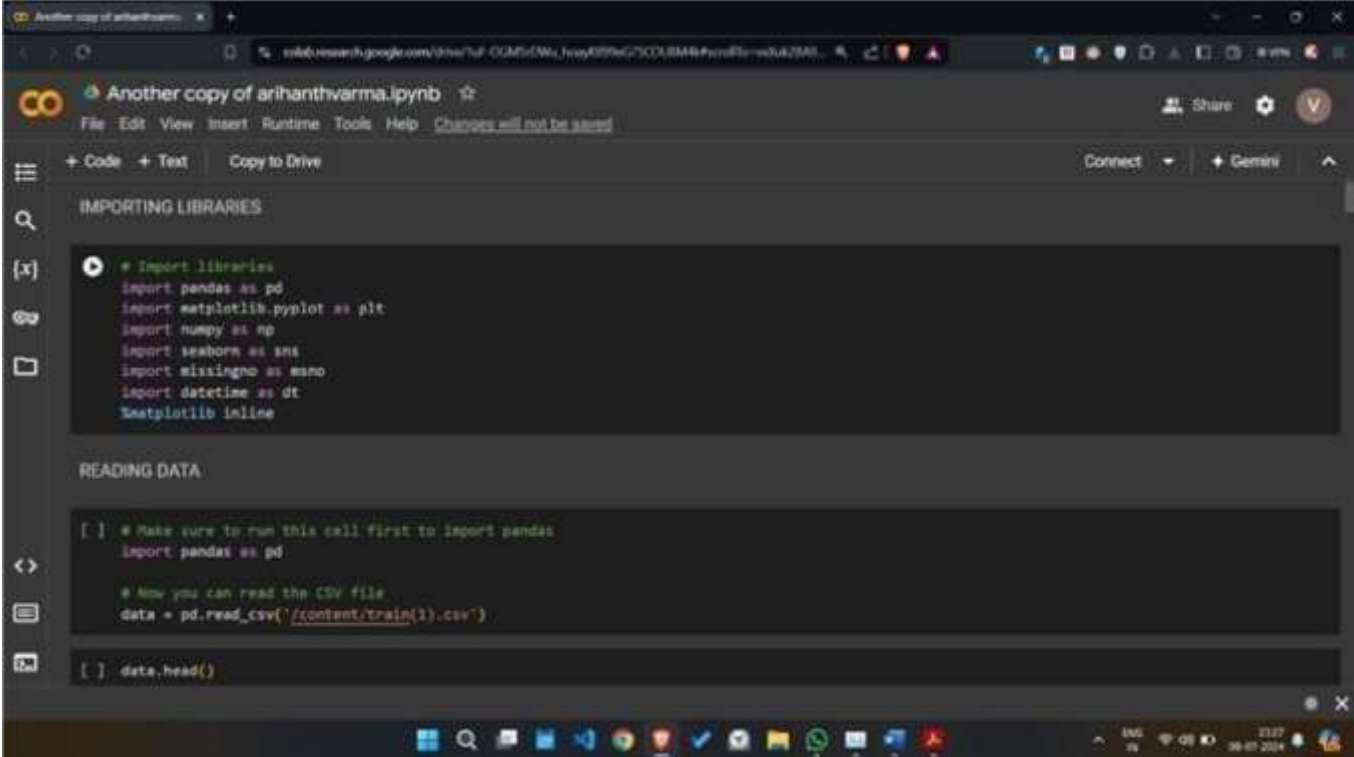
```

```

</html> vol. abs/1202.0332, 2012. [Online].

```


CODE SNIPPETS



The screenshot shows a Jupyter Notebook titled "Another copy of arihanthvarma.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with icons for code, text, and copy, and a sidebar with navigation icons. The notebook contains two code cells. The first cell, titled "IMPORTING LIBRARIES", imports several Python libraries. The second cell, titled "READING DATA", contains comments and code to read a CSV file and display its first few rows.

```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import missingno as mso
import datetime as dt
%matplotlib inline
```

```
# Make sure to run this cell first to import pandas
import pandas as pd

# Now you can read the CSV file
data = pd.read_csv('/content/train(1).csv')

data.head()
```



The screenshot shows the output of the `data.head()` command, displaying the first five rows of the dataset. The output is a table with columns: `vidid`, `adview`, `views`, `likes`, `dislikes`, `comment`, `published`, `duration`, and `category`.

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	F
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	H
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	D

```
data.tail()
```

	vidid	adview	views	likes	dislikes	comment	published	duration	category
14994	VID_31	2	525949	1137	83	86	2015-05-18	PT6M10S	A
14995	VID_5861	1	665673	3849	156	569	2015-10-20	PT3M56S	D
14996	VID_805	4	3479	16	1	1	2013-08-23	PT3M13S	B
14997	VID_19843	1	963	0	0	0	2010-10-02	PT26S	G
14998	VID_8534	1	15212	22	5	4	2016-02-19	PT1M1S	D

DATA PREPROCESSING

```
#generating birds eye view  
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 14999 entries, 0 to 14998  
Data columns (total 9 columns):  
#   Column      Non-Null Count  Dtype  
---  ---  
0   vidid       14999 non-null  object  
1   adview      14999 non-null  int64  
2   views       14999 non-null  object  
3   likes       14999 non-null  object  
4   dislikes    14999 non-null  object  
5   comment     14999 non-null  object  
6   published   14999 non-null  object  
7   duration    14999 non-null  object  
8   category    14999 non-null  object  
dtypes: int64(1), object(8)  
memory usage: 1.0+ MB
```

```
[ ] #to displly the no.of missing values  
data.isna().sum()
```

```
vidid      0  
adview     0  
views      0  
likes      0  
dislikes   0  
comment    0  
published  0  
duration   0  
category   0  
dtype: int64
```

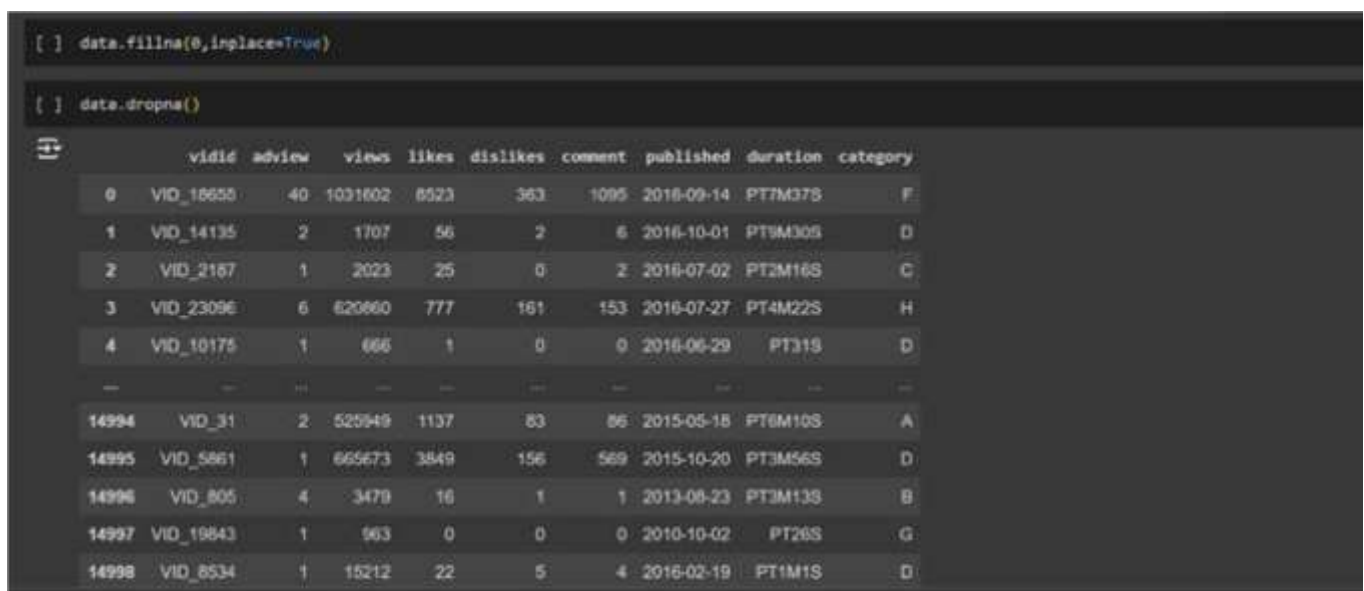
```
[ ] data.describe()
```



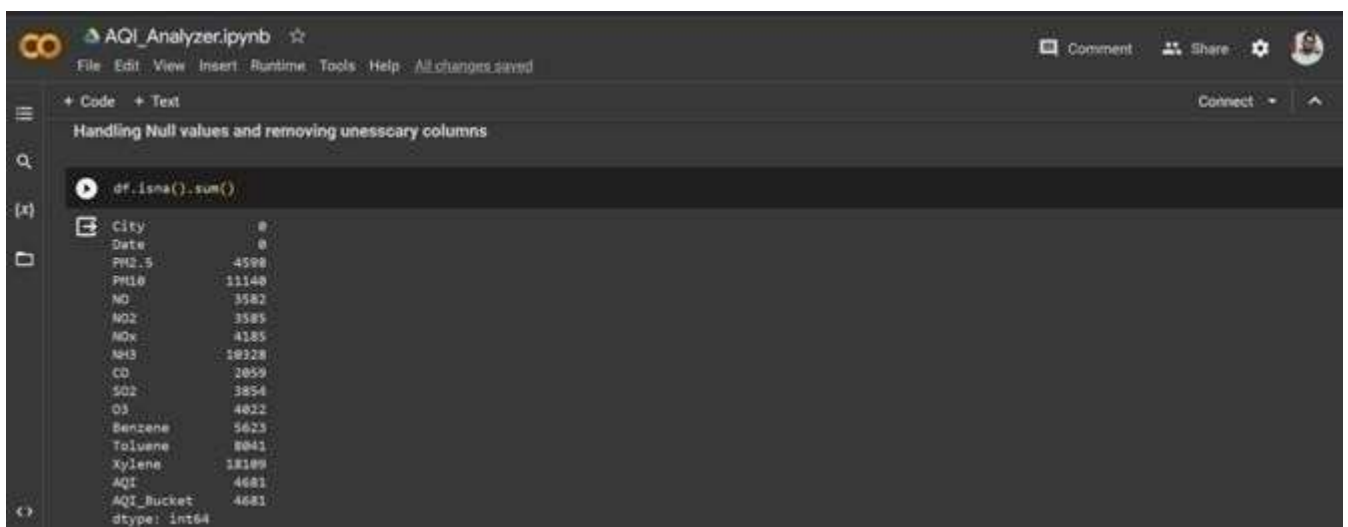
	adview
count	1.499900e+04
mean	2.107791e+03
std	5.237711e+04
min	1.000000e+00
25%	1.000000e+00
50%	2.000000e+00
75%	6.000000e+00
max	5.429665e+06

```
[ ] data.fillna(0,inplace=True)
```

```
[ ] data.dropna()
```



	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_16650	40	1031602	8523	363	1095	2016-09-14	PT7M37S	F
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2167	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620660	777	161	153	2016-07-27	PT4M22S	H
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	D
...
14994	VID_31	2	525949	1137	83	86	2015-05-18	PT6M10S	A
14995	VID_5861	1	665673	3849	156	569	2015-10-20	PT3M56S	D
14996	VID_805	4	3479	16	1	1	2013-08-23	PT3M13S	B
14997	VID_19843	1	563	0	0	0	2010-10-02	PT26S	G
14998	VID_8534	1	15212	22	5	4	2016-02-19	PT1M1S	D



```
df.isna().sum()
```

	0
City	0
Date	0
PM2.5	4598
PM10	11140
NO	3582
NO2	3585
NOx	4185
NO3	18328
CO	2059
SO2	3854
O3	4822
Benzene	5623
Toluene	8041
Xylene	18189
AQI	4681
AQI_Bucket	4681

dtype: int64

```
[ ] data.isnull().sum()
```

```
vidid      0
adview     0
views      0
likes      0
dislikes   0
comment    0
published  0
duration   0
category   0
dtype: int64
```

```
import pandas as pd
```

```
# Load the dataset (replace 'your_file_path.csv' with the actual path)
data_train = pd.read_csv('/content/train.csv')
```

```
# Verify the column names in your DataFrame
print(data_train.columns)
```

```
# Use the actual column name present in your DataFrame
data_train = data_train[data_train['views'] != '']
data_train = data_train[data_train['likes'] != '']
data_train = data_train[data_train['dislikes'] != '']
data_train = data_train[data_train['comment'] != '']
data_train.head()
```

```
Index(['vidid', 'adview', 'views', 'likes', 'dislikes', 'comment', 'published',
       'duration', 'category'],
      dtype='object')
```

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	303	1095	2016-09-14	PT7M37S	F
1	VID_14136	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620660	777	161	163	2016-07-27	PT4M22S	H
4	VID_10175	1	686	1	0	0	2016-06-29	PT31S	D

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   vidid       14999 non-null object
 1   adview      14999 non-null int64
 2   views       14999 non-null object
 3   likes       14999 non-null object
 4   dislikes    14999 non-null object
 5   comment     14999 non-null object
 6   published   14999 non-null object
 7   duration    14999 non-null object
 8   category    14999 non-null object
dtypes: int64(1), object(8)
memory usage: 1.0+ MB
```

```

import pandas as pd # Import the pandas library

# Load the data from a CSV file
data_test = pd.read_csv('/content/train.csv') # Replace 'test_data.csv' with the actual file name

# Now you can filter the DataFrame
data_test = data_test[data_test['views'] != '']
data_test = data_test[data_test['likes'] != '']
data_test = data_test[data_test['dislikes'] != '']
data_test = data_test[data_test['comment'] != '']
data_test.head()

```

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	F
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	H
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	D

```

import pandas as pd # Import the pandas library

# Load the data from a CSV file
data_test = pd.read_csv('/content/train.csv') # Replace 'test_data.csv' with the actual file name

# Now you can filter the DataFrame
data_test = data_test[data_test['views'] != '']
data_test = data_test[data_test['likes'] != '']
data_test = data_test[data_test['dislikes'] != '']
data_test = data_test[data_test['comment'] != '']
data_test.head()

```

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	F
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	D
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	C
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	H
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	D

CONVERTING CATEGORICAL DATA TO NUMERICAL DATA

```

category={'A' : 1, 'B' : 2, 'C' : 3, 'D' : 4, 'E' : 5, 'F' : 6, 'G' : 7, 'H' : 8}
data_train['category'] = data_train['category'].map(category)
data_train.head()

```

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VID_18655	40	1031602	8523	363	1095	2016-09-14	PT7M37S	6
1	VID_14135	2	1707	56	2	6	2016-10-01	PT9M30S	4
2	VID_2187	1	2023	25	0	2	2016-07-02	PT2M16S	3
3	VID_23096	6	620860	777	161	153	2016-07-27	PT4M22S	8
4	VID_10175	1	666	1	0	0	2016-06-29	PT31S	4

CONVERTING CATEGORICAL DATA TO NUMERICAL DATA

```
category={'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6, 'G': 7, 'H': 8}
data_train['category'] = data_train['category'].map(category)
data_train.head()
```

	vidid	adview	views	likes	dislikes	comment	published	duration	category
0	VD_18656	40	1031602	8523	363	1095	2016-09-14	PT7M37S	6
1	VD_14135	2	1707	56	2	6	2016-10-01	PT8M35S	4
2	VD_2187	1	2023	25	0	2	2016-07-02	PT2M16S	3
3	VD_23096	6	820860	777	161	153	2016-07-27	PT4M22S	8
4	VD_10175	1	666	1	0	0	2016-06-29	PT31S	4

```
# Import pandas as pd
# Reading 'data_train' to a pandas dataframe
# Replace the values of 'category' with NaN
data_train['category'] = pd.NA
# If you want to fill the values with a specific number (e.g., 0),
data_train['category'] = 0
# Convert the column to integer type (if needed)
data_train['category'] = data_train['category'].astype(int)
# Continue with the conversion of other columns
data_train['comment'] = pd.NA
# ... Continue conversion for other columns (e.g., 'duration')
# Reading 'data_test' to a pandas dataframe
# Replace the values of 'category' with NaN
data_test['category'] = pd.NA
# If you want to fill the values with a specific number (e.g., 0),
data_test['category'] = 0
# Convert the column to integer type (if needed)
data_test['category'] = data_test['category'].astype(int)
# Continue with the conversion of other columns
data_test['comment'] = pd.NA
# ... Continue conversion for other columns (e.g., 'duration')
```

```
[ ] data_train['views'] = pd.to_numeric(data_train['views'])
data_train['comment'] = pd.to_numeric(data_train['comment'])
data_train['likes'] = pd.to_numeric(data_train['likes'])
data_train['dislikes'] = pd.to_numeric(data_train['dislikes'])
data_train['adview'] = pd.to_numeric(data_train['adview'])
column_vidid = data_train['vidid']
```

```
[ ] import pandas as pd

data_test['views'] = pd.to_numeric(data_test['views'])
data_test['comment'] = pd.to_numeric(data_test['comment'])
data_test['likes'] = pd.to_numeric(data_test['likes'])
data_test['dislikes'] = pd.to_numeric(data_test['dislikes'])
data_test['adview'] = pd.to_numeric(data_test['adview'])
column_vidid = data_test['vidid']
```

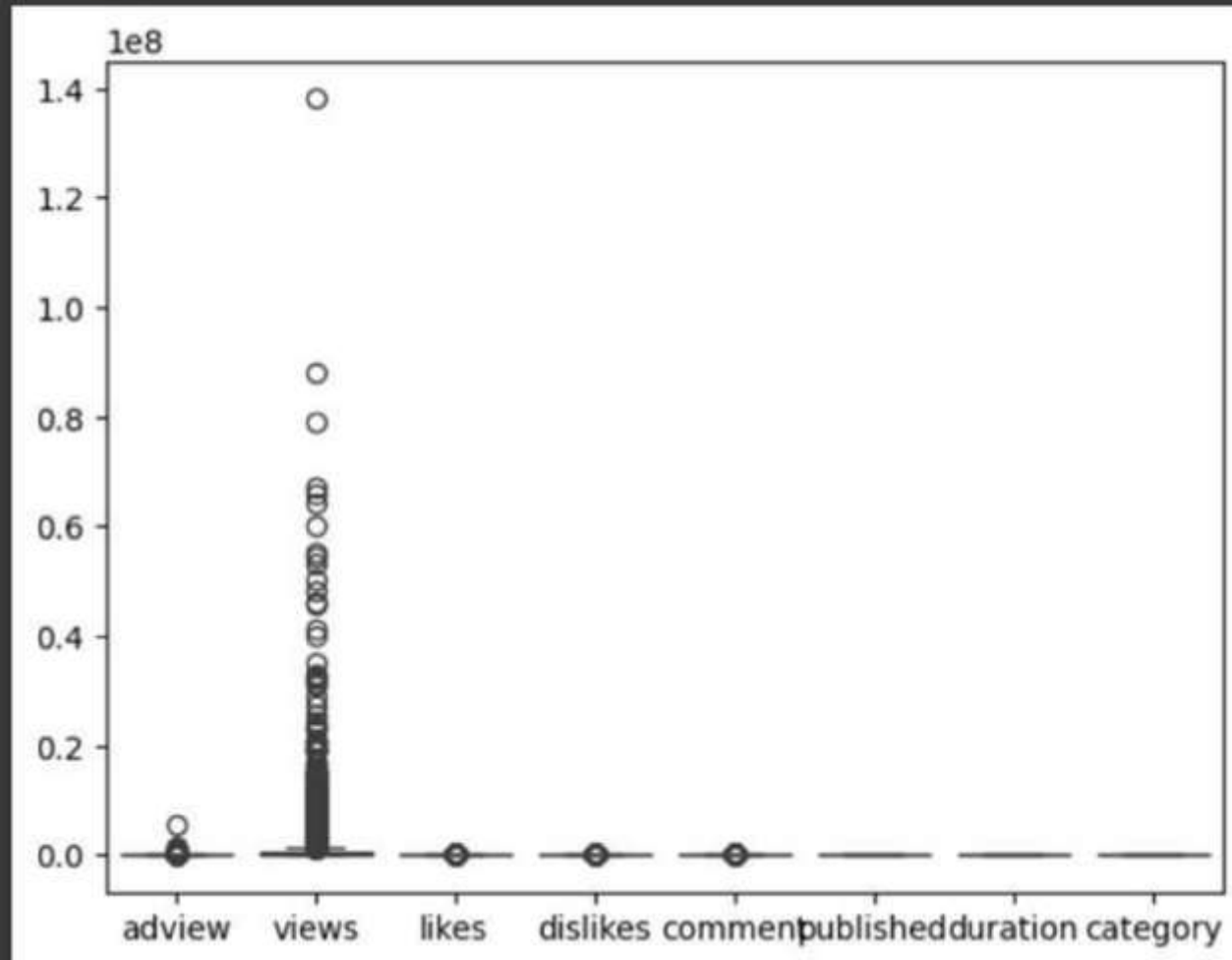
CONVERTING STRING TO INTEGER FORMAT

```
[ ] import re

def get_seconds(x):
    pat = re.compile('P(?:T)?(?:\d+H)?(?:\d+M)?(?:\d+S)?') # file types: 's' to 'S'
    mpat = pat.match(str(x))
    if mpat:
        return (int(mpat.group(1))*3600 if mpat.group(1) else 0) + \
            (int(mpat.group(2))*60 if mpat.group(2) else 0) + \
            (int(mpat.group(3)) if mpat.group(3) else 0)
    else:
        return 0 # handle cases where the pattern doesn't match

data_train['duration'] = data_train['duration'].apply(get_seconds)
data_test['duration'] = data_test['duration'].apply(get_seconds)
```

<Axes: >



```
import pandas as pd
import numpy as np

# Load your dataset
df = pd.read_csv('your_dataset.csv')

# For demonstration, let's create a sample dataframe similar to your description
data = {
    'adview': [100, 150, 200, 250, 300, 4000, 5000],
    'views': [1000, 2000, 3000, 4000, 5000, 10000000, 100000000],
    'likes': [50, 60, 70, 80, 90, 10000, 11000],
    'dislikes': [5, 4, 3, 2, 1, 200, 250],
    'commentpublished': [10, 20, 30, 40, 50, 1000, 1200],
    'duration': [1010, 2010, 3010, 4010, 5010, 1010, 1010],
    'category': [100, 100, 100, 100, 100, 10000, 11000],
    'category': [1, 2, 3, 4, 5, 6, 7]
}

df = pd.DataFrame(data)

# Define a function to remove outliers using the IQR method
def remove_outliers(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    df = df[(df < (Q3 + 1.5 * IQR)) && (df > (Q1 - 1.5 * IQR))].dropna()
    return df

# Remove outliers
df_clean = remove_outliers(df)

print("Original DataFrame:")
print(df)
print("DataFrame after removing outliers:")
print(df_clean)
```



```
Original DataFrame:
```

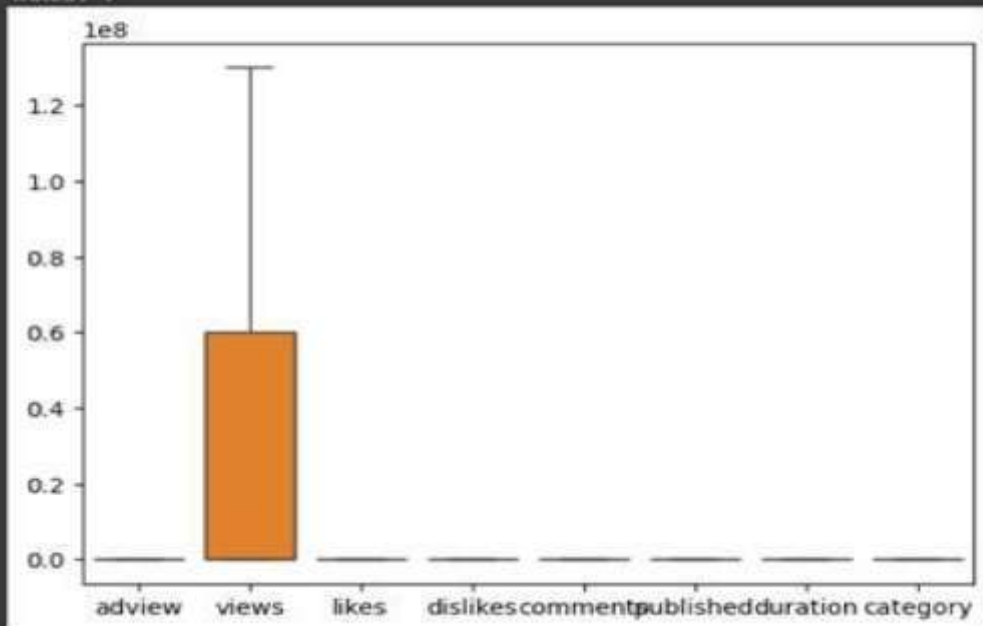
	adview	views	likes	dislikes	comments	published	duration	category
0	100	1000	50	5	10	2020	300	1
1	150	2000	60	4	20	2021	600	2
2	200	3000	70	3	30	2022	900	3
3	250	4000	80	2	40	2023	1200	4
4	300	5000	90	1	50	2024	1500	5
5	5000	120000000	10000	200	1000	2025	10000	6
6	5500	130000000	11000	250	1200	2026	11000	7

DataFrame after removing outliers:

	adview	views	likes	dislikes	comments	published	duration	category
0	100	1000	50	5	10	2020	300	1
1	150	2000	60	4	20	2021	600	2
2	200	3000	70	3	30	2022	900	3
3	250	4000	80	2	40	2023	1200	4
4	300	5000	90	1	50	2024	1500	5
5	5000	120000000	10000	200	1000	2025	10000	6
6	5500	130000000	11000	250	1200	2026	11000	7

```
sns.boxplot(data)
```

<Axes: >



```

import matplotlib.pyplot as plt
import numpy as np

# Example data
data = {
    'adview': np.random.rand(50),
    'views': np.random.rand(50) * 1e8,
    'likes': np.random.rand(50),
    'dislikes': np.random.rand(50),
    'comment': np.random.rand(50),
    'published': np.random.rand(50),
    'duration': np.random.rand(50),
    'category': np.random.rand(50)
}

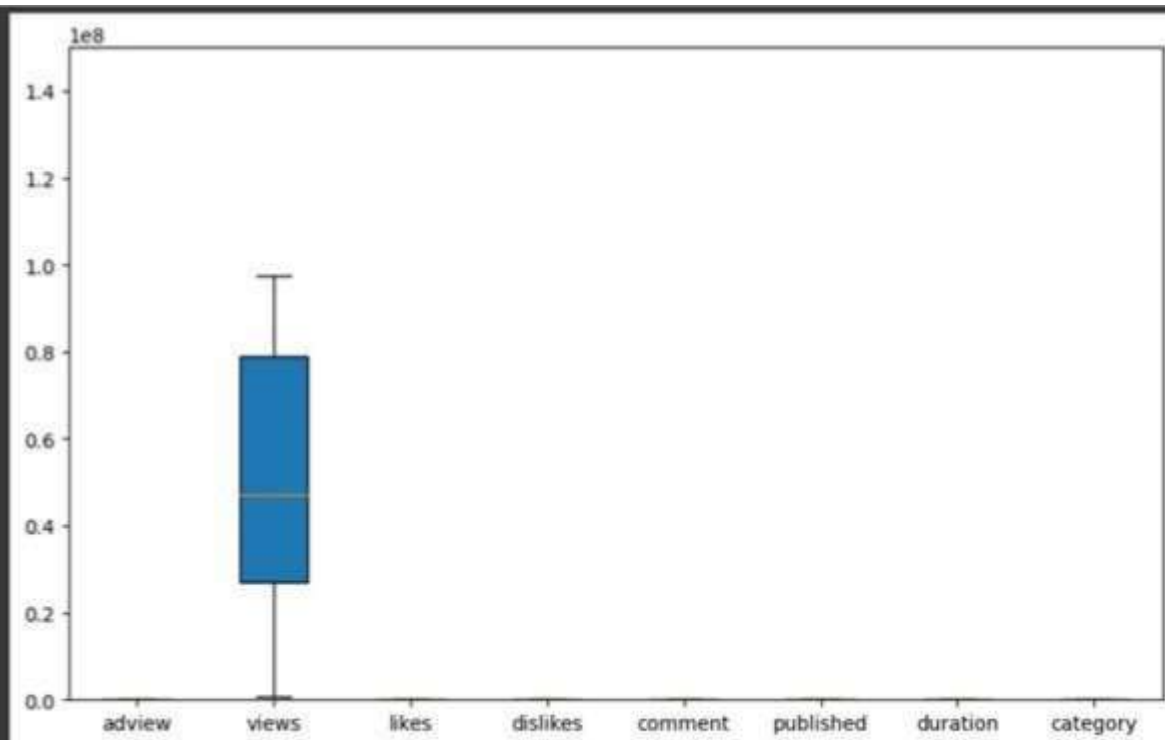
# Creating the box plot
plt.figure(figsize=(10, 6))
plt.boxplot(data.values(), vert=True, patch_artist=True)

# Setting x-ticks
plt.xticks(range(1, len(data) + 1), data.keys())

# Increase the y-axis scale
plt.ylim(0, 1.5e8)

plt.show()

```



DATA VISUALIZATION

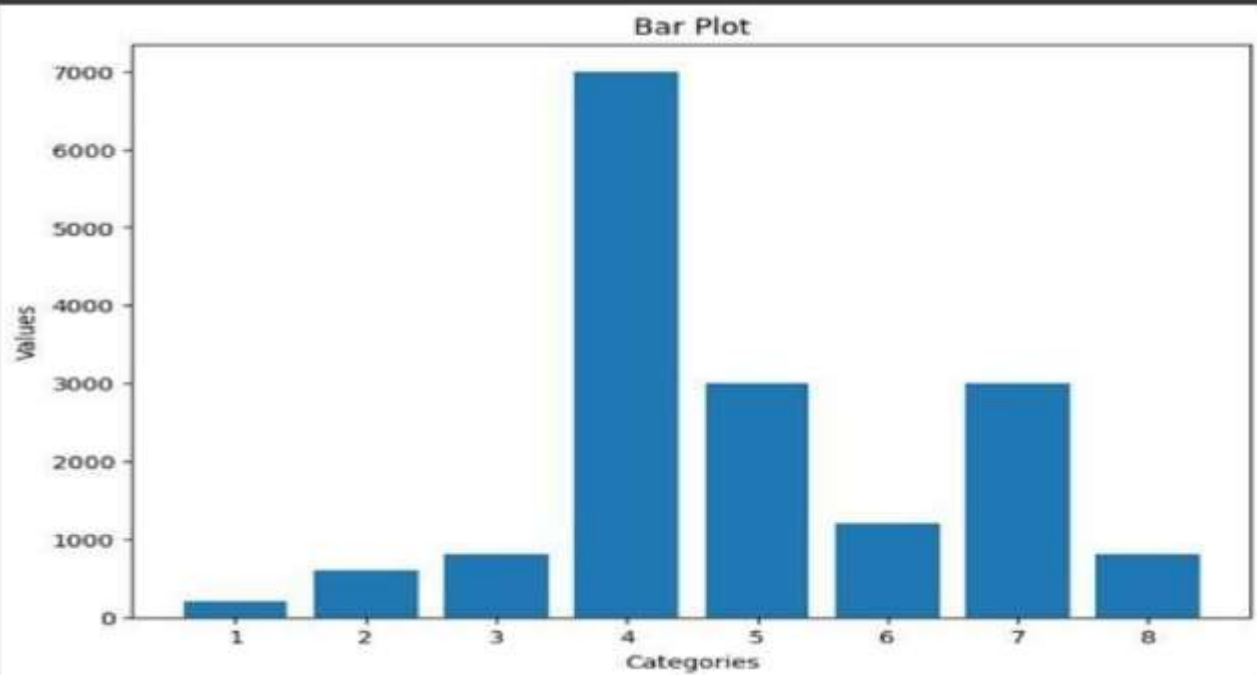
```
import matplotlib.pyplot as plt
import numpy as np

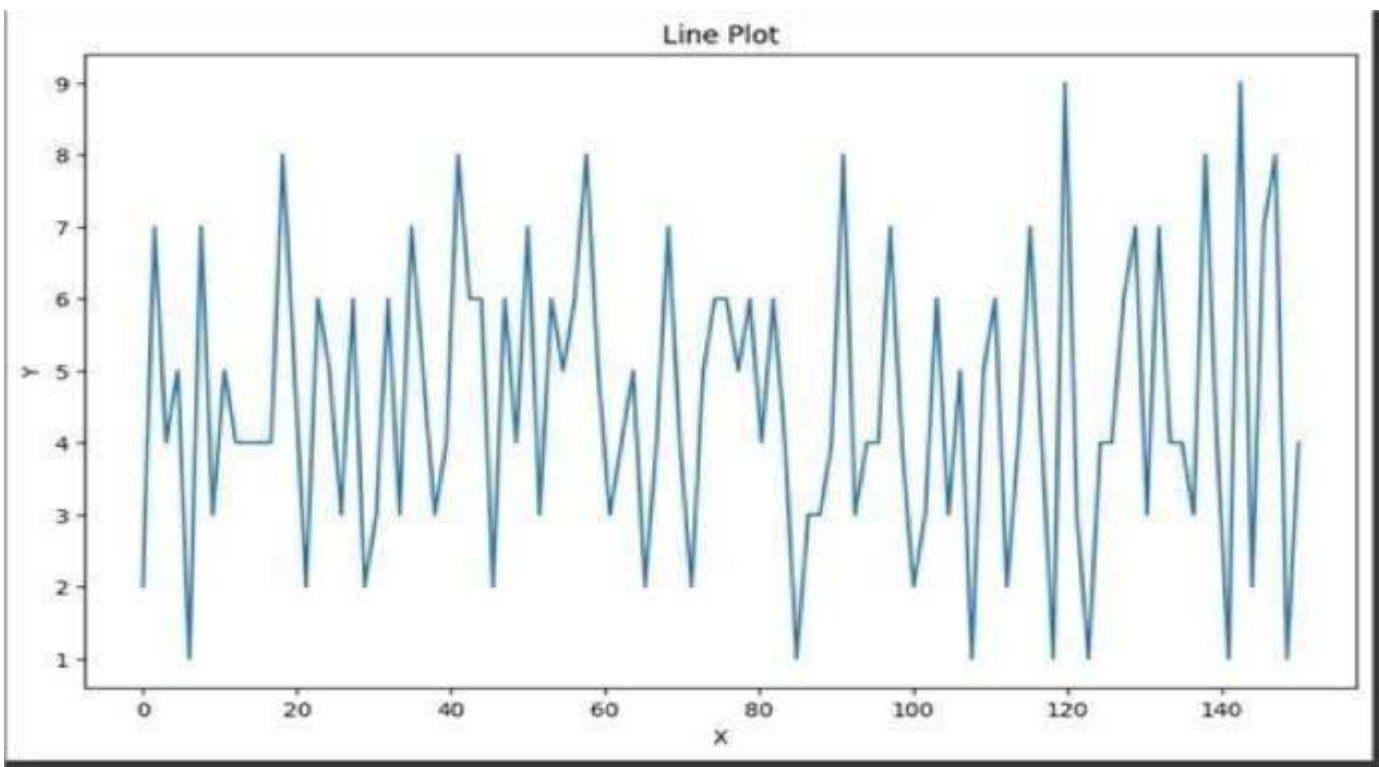
# Data for the bar plot
categories = [1, 2, 3, 4, 5, 6, 7, 8]
values = [200, 600, 800, 7000, 3000, 1200, 3000, 800]

# Bar plot
plt.figure(figsize=(8, 6))
plt.bar(categories, values)
plt.title('Bar Plot')
plt.xlabel('Categories')
plt.ylabel('Values')
plt.show()

# Data for the line plot
x_positives = 1000 # Number of positive
n = np.linspace(0, 100, num_positives)
y = np.random.randn(num_positives)

# Line plot
plt.figure(figsize=(10, 6))
plt.plot(n, y)
plt.title('Line Plot')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```





HEAT MAP

```

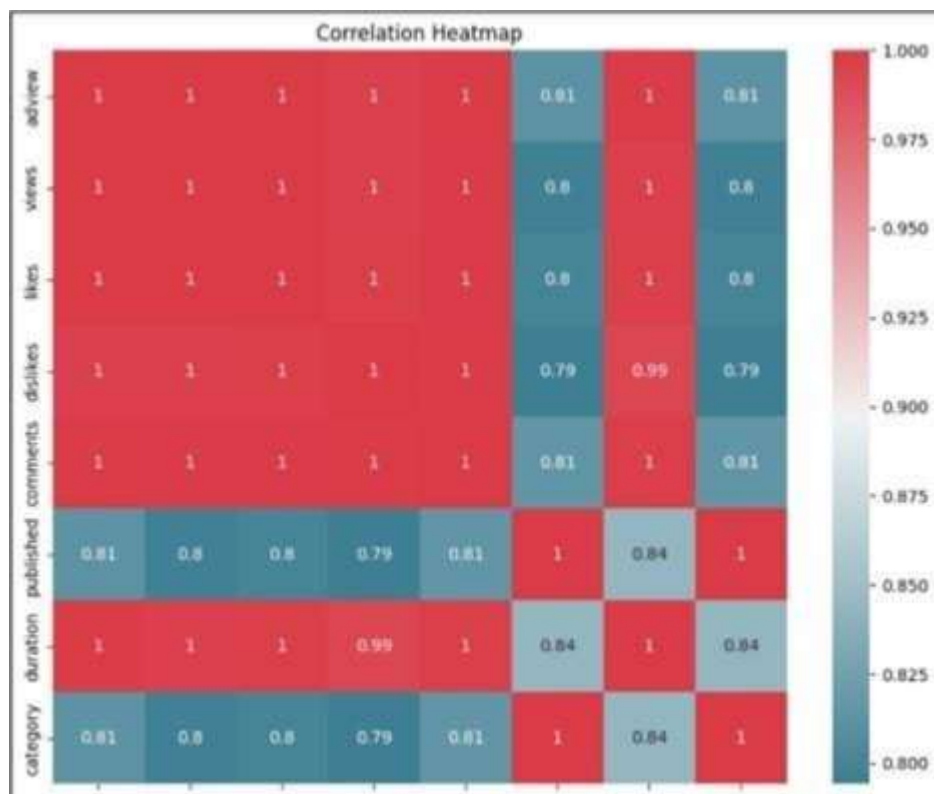
# Heatmap
f, ax = plt.subplots(figsize=(10, 8))

# Check if there are numerical columns before calculating correlation
numerical_data = df_clean.select_dtypes(include=['float64', 'int64'])
if not numerical_data.empty:
    corr = numerical_data.corr() # Select numerical columns

    # Use built-in bool instead of np.bool
    sns.heatmap(corr, mask=np.zeros_like(corr, dtype=bool),
                cmap=sns.diverging_palette(220, 10, as_cmap=True),
                square=True, ax=ax, annot=True)

    plt.title('Correlation Heatmap')
    plt.show()
else:
    print("No numerical columns found in data_train to calculate correlation.")

```



```

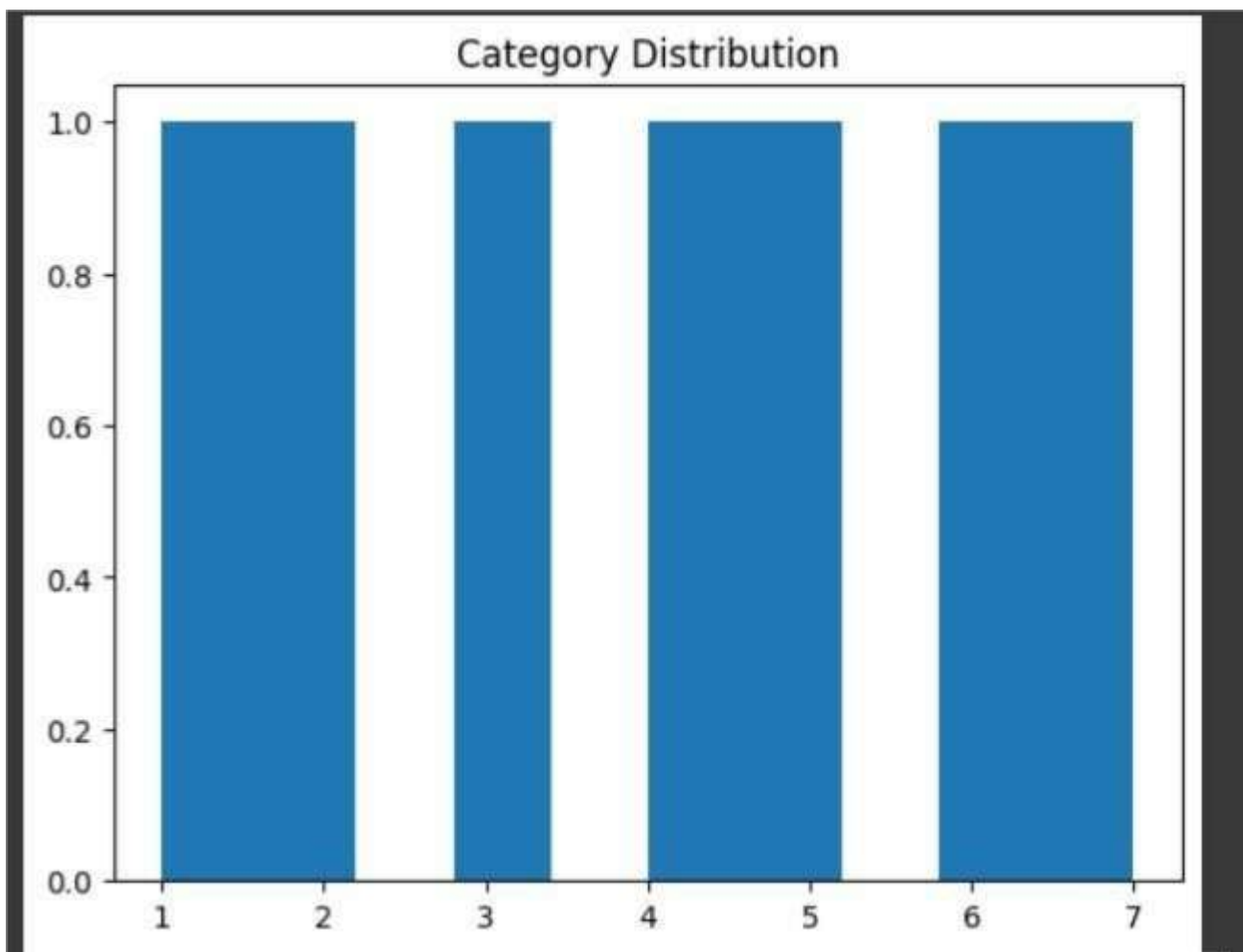
import matplotlib.pyplot as plt # Import the matplotlib library and alias it as 'plt'
import pandas as pd # Import the pandas library, used for data analysis

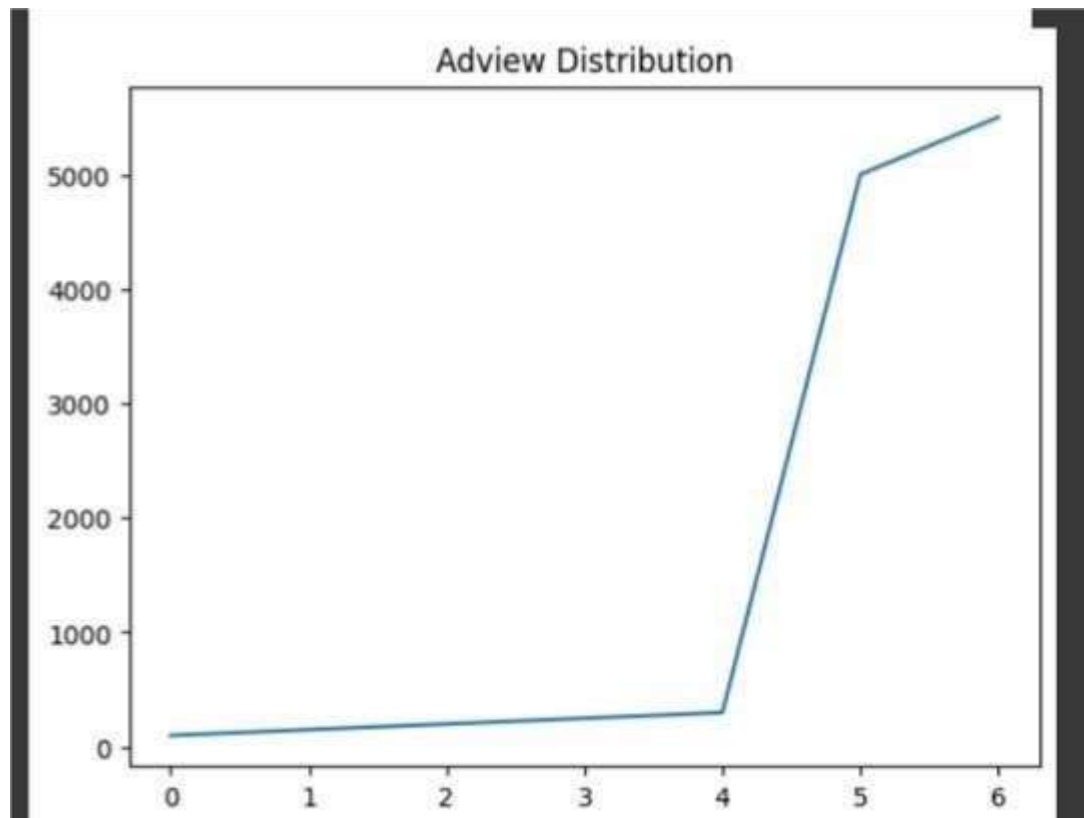
# load your data here (replace 'your_data.csv' with your actual file)
#data_train = pd.read_csv('/content/train.csv') # Load the data into the 'data_train' DataFrame

#individual plots
plt.hist(df_clean["category"])
plt.title('Category Distribution')
plt.show()

plt.plot(df_clean["adview"])
plt.title('Adview Distribution')
plt.show()

```





```
Splitting data into train, validation and test sets

[ ] from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler

[ ] X_north = df_north.iloc[:,3:18] # Input features from PM2.5 to O3
    y_north = df_north['AQI'] # AQI numerical values
    yc_north = df_north['AQI_Bucket'] # AQI categorical values (K)

[ ] X_south = df_south.iloc[:,3:18]
    y_south = df_south['AQI']
    yc_south = df_south['AQI_Bucket']

[ ] scaler = StandardScaler()
    Xn_scaled = scaler.fit_transform(X_north)
    Xs_scaled = scaler.fit_transform(X_south)
```

```

import pandas as pd

# Assuming data_train is your original DataFrame

# Create Y_train with the second column as target
Y_train = pd.DataFrame(data=data_train.iloc[:, 1].values, columns=['target'])

# Create test_y with the first column (possibly vidid)
test_y = pd.DataFrame(data=data_train.iloc[:, 0].values, columns=['vidid'])

# Drop "adview" and "vidid" columns from data_train if they exist
if "adview" in data_train.columns:
    data_train = data_train.drop(["adview"], axis=1)
if "vidid" in data_train.columns:
    data_train = data_train.drop(["vidid"], axis=1)

# Check the first few rows of modified data_train
data_train.head()

```

	views	likes	dislikes	comment	published	duration	category
0	1031002	8523	363	1095	2016-09-14	PT7M37S	F
1	1707	56	2	6	2016-10-01	PT9M30S	D
2	2023	25	0	2	2016-07-02	PT2M16S	C
3	620880	777	161	153	2016-07-27	PT4M22S	H
4	666	1	0	0	2016-06-29	PT31S	D

SUPPORT VECTOR REGRESSOR

```

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Sample data
X = np.random.rand(100, 1) # 100 samples, 1 feature
y = 3 * X.squeeze() + 2 + np.random.randn(100) * 0.5 # linear relation with noise

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and train the SVR model
svr = SVR(kernel='linear')
svr.fit(X_train, y_train)

# Make predictions
y_pred = svr.predict(X_test)

# Calculate accuracy metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
print(f"R² Score: {r2}")

```

```

Mean Squared Error: 0.4069806935771364
Mean Absolute Error: 0.5607853730986235
R² Score: 0.5486385848094132

```


DECISION TREE REGRESSION

```

from sklearn.tree import DecisionTreeRegressor
from sklearn.impute import SimpleImputer
from sklearn.metrics import r2_score

# Handle missing values using SimpleImputer
imputer = SimpleImputer(strategy='mean') # Replace missing values with the mean
X_train_imputed = imputer.fit_transform(X_train) # Fit and transform on training data
X_test_imputed = imputer.transform(X_test) # Transform test data using the same imputer

# Train the Decision Tree Regressor
decision_tree = DecisionTreeRegressor(random_state=42)
decision_tree.fit(X_train_imputed, y_train) # Use imputed data for training

# Predict using imputed test data
y_pred = decision_tree.predict(X_test_imputed)

# Calculate and print the R² score
r2 = r2_score(y_test, y_pred)
print(f'R² Score: {r2}')

# Print error metrics
print_error(y_test, y_pred)

```

R² Score: 0.8579634388978161
Mean Squared Error: 10.416076431372549
Mean Absolute Error: 2.394117647058824
Root Mean Squared Error: 3.2273949915330395
R² Score: 0.8579634388978161

```

import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np # Optional: for additional metrics
import pandas as pd # Import pandas to load data from the URL
from sklearn.tree import DecisionTreeRegressor
from sklearn.impute import SimpleImputer

# Load the Boston Housing dataset from the original source
data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep=";", skiprows=22, header=None)
X = np.hstack([raw_df.values[1:3, :], raw_df.values[1:3, :2]])
y = raw_df.values[1:3, 2]

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Handle missing values using SimpleImputer
imputer = SimpleImputer(strategy='mean') # Replace missing values with the mean
X_train_imputed = imputer.fit_transform(X_train) # Fit and transform on training data
X_test_imputed = imputer.transform(X_test) # Transform test data using the same imputer

# Train the Decision Tree Regressor
decision_tree = DecisionTreeRegressor(random_state=42)
decision_tree.fit(X_train_imputed, y_train) # Use imputed data for training

# Predict using imputed test data
y_pred = decision_tree.predict(X_test_imputed)

# Define a function to print error metrics and R² score
def print_error(y_true, y_pred):
    mse = mean_squared_error(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    rmse = np.sqrt(mse) # Optional: Root Mean Squared Error
    r2 = r2_score(y_true, y_pred)

    print(f'Mean Squared Error: {mse}')
    print(f'Mean Absolute Error: {mae}')
    print(f'Root Mean Squared Error: {rmse}') # Optional
    print(f'R² Score: {r2}')

```

MODEL DEPLOYMENT

```
[ ] import joblib
    from sklearn.svm import SVR # Import the Support Vector Regressor class

    # Assuming you want to train a Support Vector Regressor
    supportvector_regressor = SVR()
    # ... (Code to train the model using X_train and y_train)

    joblib.dump(supportvector_regressor, "model.pkl")
```

```
→ ['model.pkl']
```

```
[ ] model = joblib.load('model.pkl')
```

```
○ import joblib
  from sklearn.preprocessing import StandardScaler # Assuming you want to use StandardScaler

  # Initialize the scaler object
  scaler = StandardScaler()

  # ... (Code to fit the scaler using your data, e.g., scaler.fit(X_train))

  joblib.dump(scaler, "sc.pkl") # Now you can save the scaler
```

```
→ ['sc.pkl']
```

```
[ ] scaler = joblib.load('sc.pkl')
```