Stars and Stats: Investigating Space Station YouTube Viewer Behavior

Tasneem Banu
College of Engineering, Design, and Computing
University of Colorado Denver
Denver, Colorado, USA
tasneem.banu@ucdenver.edu

Index

1. Introduction

In today's technologically advanced world, the volume of data is growing exponentially, giving way to an entirely new dimension in scientific advancements. Every day, we generate a vast amount of data, which is getting increasingly complex. In such instances, we can use Big Data Science to extract and handle the data while also allowing us to predict patterns and train machine learning models. While these models perform well at making predictions, they often operate as black boxes, making their decision-making processes non-transparent and difficult to understand. This is where Explainable learning bridges the gap between accuracy and interpretability. In my project, I do not just arrive at the outcome but also unravel the reason behind the algorithms' decisions. One exciting application of such an algorithm is in YouTube playlist analysis. Space exploration has always been an exciting topic for humans. In this project, I will work with explainable learning algorithms using YouTube data. From the results of the trend forecasting and sentiment analysis, I believe the space content creators would be able to refine their content such that it is in the interest of the public and is in line with the educational requirements. It will also help pave way for simpler scientific communications among public which would increase the cultural impact of space related information.

2. Motivation

On July 14th, the Satish Dhawan Space Center in India successfully launched Chandrayaan 3, and it became the most-watched live stream globally. The Indian Space Research Organization (ISRO) has become a common name among all space enthusiasts. This impressive feat ignited my curiosity and crystallized my motivation. In this course, I found the perfect opportunity to blend big data science with space exploration. My project aims to make sense of the complex information that tells us about our space through the lenses of various government and private space channels present on YouTube.

3. Problem Statement and Background

3.1 Description of the Dataset

The dataset is a collection of rich media and viewer interactions from rich repository of video content from the official YouTube channels of major space agencies like ISRO, NASA, ESA, Roscosmos, and key private sector players such as SpaceX and Boeing. The data encompasses video metadata, viewer engagement metrics, and user comments, all sourced through the YouTube Data API. Through Python scripts I will extract and preprocess this data. This API will provide access to a wealth of information, including titles, descriptions, view counts and detailed user comments. The analysis will extend through exploratory data analysis and statistical methods to investigate trend forecasting, and sentiment analysis to decode public perception. The ability to reveal trends and develop a reliable sentiment analysis model to provide actionable insights for content creators and educators is how I would define success in this project.

3.2 Problem Statement

The core issue of this project is to comprehend and measure the worldwide involvement with space exploration information on digital platforms, specifically YouTube. It is not clear what exactly drives interaction and positive emotions in this area. This is the task I want to investigate. I want to analyze this unorganized data by directly acquiring it from YouTube Data, identifying trends in popularity, determining what affects audience interaction, and uncovering the sentiments conveyed in comments. The objective of my study is to address the disparity between the algorithms' decisions and the reasons behind these decisions by utilizing explainable machine-learning techniques that provide insight into the opaque nature of data analytics. The objective here is to provide individuals involved in content creation, education, and those with a keen interest in space with the necessary information to develop engaging tales that effectively capture the attention of the audience.

4. Methodology and Results

4.1.Importing Libraries

In this model, I make use of a few libraries that make it easier for us to work. These dependencies are installed, and then imported in accordance with the project needs. Here I will investigate all the libraries that I have used in our code.

4.1.1 google-api-python-client

This is a library that facilitates access to a range of Google APIs, such as YouTube, Google Drive, Calendar, and others. The utilization of this library is for the purpose of automating data collection and analysis, particularly when considering the extensive amount of data available on the YouTube platform.

4.1.2 Pandas

Pandas are a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool built on top of the Python programming language. It provides data structures and operations for manipulating numerical tables and time series.

4.1.3 vaderSentiment

VADER (Valence Aware Dictionary and Sentiment Reasoner) SentimentIntensityAnalyzer is a tool for sentiment analysis. It is particularly well-suited for analyzing sentiments expressed in social media due to its reliance on a lexicon that is specifically attuned to the types of expressions found in such contexts.

4.1.4 Natural Language Toolkit

It is a cross-platform data visualization and graphical plotting library for Python and its numerical extension NumPy. In most instances, a few lines of code are all that is required to generate a visual data plot using a Python matplotlib script.

4.1.5 sci-kit learn

A popular python machine learning library for getting started with machine learning. It provides efficient tools for machine learning and statistical modeling. The *train_test_split* function creates training and testing data for features and labels and produces an evaluation matrix.

4.1.6 LIME

Local Interpretable Model-agnostic Explanations is a Python library that helps users to interpret machine learning models. It works on the principle that even though the model may be complex and difficult to interpret, it can explain the predictions of the model by approximating it locally with an interpretable model.

4.1.7. SMOTE

Synthetic Minority Over-sampling Technique is a popular technique for addressing class imbalance in machine learning datasets by generating synthetic samples for the minority class. This can improve the performance of machine learning models, especially when dealing with imbalanced datasets.

4.2 Defining API Keys and Channel IDs

4.2.1 Defining API Keys

API keys are used to track and control how the API is being used. For using API key for this project, I first registered by signing up for YouTube data API and generated my own API key.

4.2.2 Defining Channel IDs

Here I have used a channel IDs array which will have a unique channel ID for each YouTube channel.

```
channel_ids = ['UCw5hEVOTfz_AfzsNFWyNlNg', 'UCLA_DiR1FfKNvjuUpBHmylQ', 'UCIBaDdAbGlFDeS33shmlD0A', 'UCtI0Hodo5o5dUb6
               #ISRO, NASA, ESA, SPACEX, BLUEORIGIN, ROCKETLAB, CSA
               ]
```

4.2.3 API Client

Here I have instantiated a client for the YouTube Data API. Using this YouTube service object, I will make requests to the You Tube Data API and use it for further analysis.

```
api_service_name = "youtube"
api_version = "v3"

youtube = build(
    api_service_name, api_version, developerKey=api_key)
```

4.3 Retrieving Statistical Information

4.3.1 Statistics regarding You Tube channels

Here I have defined a function *get_channel_stats* to retrieve statistics for a list of channels using the YouTube Data API. It creates a dictionary data for each channel with this information. It returns a Data Frame *channel_stats* with the statistics for each channel in *channel_ids*. *get_channel_stats* give a quick overview of channel statistics including a total video count as reported by the API.

```
def get_channel_stats(youtube, channel_ids):

    all_data = []

    request = youtube.channels().list(
        part="snippet,contentDetails,statistics",
        id=','.join(channel_ids)
    )
    response = request.execute()

    #loop through items
    for item in response['items']:
        data = {'channelName': item['snippet']['title'],
                'subscribers':item['statistics']['subscriberCount'],
                'views':item['statistics']['viewCount'],
                'totalVideos':item['statistics']['videoCount'],
                'playlistId':item['contentDetails']['relatedPlaylists']['uploads']
       }
        all_data.append(data)

    return(pd.DataFrame(all_data))
```

```
channel_stats
```

	channelName	subscribers	views	totalVideos	playlistId
0	SpaceX	6570000	701372647	519	UUtI0Hodo5o5dUb67FeUjDeA
1	Rocket Lab	164000	9608073	104	UUsWq7LZaizhli-c-Yo_bcpw
2	NASA	11400000	966490254	5711	UULA_DiR1FfKNvjuUpBHmylQ
3	European Space Agency, ESA	996000	227277214	4709	UUIBaDdAbGlFDeS33shmlD0A
4	Blue Origin	258000	30597126	135	UUVxTHEKKLxNjGcvVaZindlg
5	Canadian Space Agency	1120000	288329128	568	UUdNtqpHlU1pCaVy2wlzxHKQ
6	ISRO Official	4530000	159041742	86	UUw5hEVOTfz_AfzsNFWyNlNg

4.3.2 Number of videos in each playlist

Here I will define *get_video_count_per_playlist* to retrieve the number of videos in each playlist, given a list of playlist IDs. Thus, this code will give a dictionary that links each playlist ID to the total number of videos it contains, which is helpful for understanding the amount of content that each YouTube channel produces. It gives us an exact count for each playlist.

```
#statistics for YouTube channels
channel_stats_df = get_channel_stats(youtube, channel_ids)
playlist_ids_list = channel_stats_df['playlistId'].tolist()

#number of videos in each playlist
video_counts_for_playlists = get_video_count_per_playlist(youtube, playlist_ids_list)
print(video_counts_for_playlists)
```

```
{'UUtI0Hodo5o5dUb67FeUjDeA': 519, 'UUVxTHEKKLxNjGcvVaZindlg': 135, 'UUsWq7LZaizhIi-c-Yo_bcpw': 104, 'UUIBaDdAbGlFDe
S33shmlD0A': 4709, 'UUdNtqpHlU1pCaVy2WlzxHKQ': 569, 'UULA_DiR1FfKNvjuUpBHmylQ': 5717, 'UUw5hEVOTfz_AfzsNFWyNlNg': 8
6}
```

4.3.3 Detailed information of each Video

get_video_details function gathers detailed statistics and metadata for a set of videos. The *all_video_info* data frame includes the videoID, channelTitle and all the other parameters as mentioned in *get_video_details* function.

```
def get_video_details(youtube, video_ids):
    all_video_info = []

    for i in range(0, len(video_ids), 50):
        request = youtube.videos().list(
            part="snippet,contentDetails,statistics",
            id =','.join(video_ids[i:i+50])
        )
        response = request.execute()

        for video in response['items']:
            stats_to_keep = {
                'snippet': ['channelTitle', 'title', 'description', 'tags', 'publishedAt'],
                'statistics': ['viewCount', 'likeCount', 'favoriteCount', 'commentCount'],
                'contentDetails': ['duration', 'definition', 'caption']
            }
```

After this the *get_video_ids* section retrieves all video IDs for each playlist.

```
def get_all_video_ids(youtube, playlist_ids):
    all_ids = {}
    for playlist_id in playlist_ids:
        video_ids = get_video_ids(youtube, playlist_id)
        all_ids[playlist_id] = video_ids
    return all_ids
```

Finally using the *get_all_video_details* function will retrieve details about each video. The below code prints out the top 5 rows of *all_video_details data frame* for each channel ID.

```
Top 5 videos for channel: ISRO Official
      video_id   channelTitle  \
0  scVBrnTde5M  ISRO Official
1  To9_pBpLyP0  ISRO Official
2  BMig6ZpqrIs  ISRO Official
3  _IcgGYZTXQw  ISRO Official
4  beK3C5lMZ6A  ISRO Official

                                               title  \
0  Gaganyaan TV-D1 Mission - Test flight from Sat...
1  Gaganyaan TV-D1 Mission - Test flight from Sat...
2  Gaganyaan TV-D1 Mission - Test flight from Sat...
3  Launch of PSLV-C57/Aditya-L1 Mission from Sati...
4  Chandrayaan-3  Addressing by  Honorable Prime ...

                                         description  \
0  Gaganyaan TV-D1 is “In-flight Abort Demonstrat...
1  Gaganyaan TV-D1 is “In-flight Abort Demonstrat...
2  Gaganyaan TV-D1 is “In-flight Abort Demonstrat...
```

Using the keys from *all_video_details* I printed out the details for each channel using the desired channel key. Below are the results for ISRO Official.

```
      video_id   channelTitle  \
0   scVBrnTdeSM  ISRO Official
1   To9_pBpLyP0  ISRO Official
2   BMig6ZpqrIs  ISRO Official
3   _IcgGYZTXQw  ISRO Official
4   beK3C5lMZ6A  ISRO Official
..          ...           ...
84  h87hLynFiaQ  ISRO Official
85  -—8ORixBXQE  ISRO Official
86  b6cYRXWrp_U  ISRO Official
87  _Rf6pN-OIQk  ISRO Official
88  2SRMbDv0kDQ  ISRO Official

                                                title  \
0   Gaganyaan TV-D1 Mission - Test flight from Sat...
1   Gaganyaan TV-D1 Mission - Test flight from Sat...
2   Gaganyaan TV-D1 Mission - Test flight from Sat...
3   Launch of PSLV-C57/Aditya-L1 Mission from Sati...
4   Chandrayaan-3  Addressing by  Honorable Prime ...
..                                                ...
84                                MOON LANDING SITES
85                  ISRO CHANDRAYAAN-2 (3D ANIMATION)
86                      ISRO CHANDRAYAAN-2 (TEASER_2)
87                   ISRO CHANDRAYAAN-2 (TEASER HINDI)
88                  ISRO CHANDRAYAAN-2 (TEASER ENGLISH)

                                          description  \
0   Gaganyaan TV-D1 is “In-flight Abort Demonstrat...
1   Gaganyaan TV-D1 is “In-flight Abort Demonstrat...
2   Gaganyaan TV-D1 is “In-flight Abort Demonstrat...
3   Aditya L1 shall be the first space based India...
4   Chandrayaan-3  Addressing by  Honorable Prime ...
..                                                ...
84
85
86
87
88

                                          tags          publishedAt  \
0                                         None  2023-10-21T05:30:50Z
1                                         None  2023-10-21T05:02:33Z
2                                         None  2023-10-21T03:24:00Z
3                                         None  2023-09-02T07:41:32Z
4   [chandrayaan 3 live, chandrayaan 3 live update...  2023-08-26T03:20:28Z
..                                                ...           ...
84                                        None  2019-07-03T08:30:38Z
85                          [#ISRO #CHANDRAYAAN.2]  2019-06-25T12:01:58Z
86                          [#ISRO #CHANDRAYAAN.2]  2019-06-25T11:57:25Z
87                          [#ISRO #CHANDRAYAAN.2]  2019-06-25T11:52:11Z
88                          [#ISRO #CHANDRAYAAN.2]  2019-06-25T11:49:18Z

   viewCount likeCount commentCount    duration definition caption
0     268788     11563          476     PT7M1S         hd   false
1     332871     20573          683    PT27M6S         hd   false
2     538491     45717         1127   PT35M40S         hd   false
3    7446708    524023         6176  PT1H50M56S         hd   false
4    6284032    248859        12214     PT1H6S         hd   false
..       ...       ...          ...         ...        ...     ...
84    213641     11137          363     PT37S         hd   false
85   4158665    136400         3866    PT3M55S         hd   false
86     48617      1900           49    PT2M40S         hd   false
87     41688      2034           90     PT3M1S         hd   false
88     77272      5034          230    PT2M32S         hd   false

[89 rows x 12 columns]
```

4.4 Trend Forecasting

4.4.1 Data Cleaning

Here I converted the '*viewCount*' column to integers for each data Frame in *all_video_details* dictionary and replaced *'None'* values with zeros.

```python
# Replace 'None' with zeros and convert 'viewCount' to integers
for channel_name, details_df in all_video_details.items():
    details_df['viewCount'] = details_df['viewCount'].replace(to_replace=[None], value=0).astype(int)
    all_video_details[channel_name] = details_df
```

4.4.2 Handling class imbalance using SMOTE

Here I calculated the percentile thresholds to determine *is_trending* status based on the below code logic.

```python
# Calculate the chosen percentile view count threshold for each channel
percentile_thresholds = {}
for channel_name, details_df in all_video_details.items():
    # Consider only videos older than 2 weeks for historical threshold calculation
    historical_videos = details_df[pd.to_datetime(details_df['publishedAt'], utc=True) < (pd.Timestamp.utcnow()
    print(historical_videos['publishedAt'].head())
    threshold = percentile(historical_videos['viewCount'], chosen_percentile)
    percentile_thresholds[channel_name] = threshold
    print(f"Channel: {channel_name}, {chosen_percentile}th Percentile: {threshold}")

# Now, for each channel, determine if a video is trending based on the new percentile
for channel_name, details_df in all_video_details.items():
    # Calculate the 'is_trending' status for each video
    details_df['is_trending'] = details_df.apply(
        lambda x: 1 if (x['viewCount'] >= percentile_thresholds[channel_name] and
                        (pd.Timestamp.utcnow() - pd.to_datetime(x['publishedAt'], utc=True) <= pd.Timedelta(week
        axis=1
    )
```

After that I defined the features array to consist of *viewCount, likeCount, commentCount,* and *days_since_published* for *is_trending* target variable. Finally, I applied SMOTE since my dataset had very unbalanced classes. SMOTE oversamples the minority class to address the class imbalance issue. After following these preprocessing steps, we will have a balanced training set (*X_train_resampled* and *y_train_resampled*) that can be used to train machine learning model to predict whether a video is trending or not. This balanced dataset helps prevent the model from being biased towards the majority class.

```
Channel: European Space Agency, ESA, 60th Percentile: 7389.599999999998
Number of Trending Videos: 56
Channel: Rocket Lab, 60th Percentile: 84944.8
Number of Trending Videos: 1
Channel: ISRO Official, 60th Percentile: 154390.2
Number of Trending Videos: 10
Channel: Blue Origin, 60th Percentile: 83845.4
Number of Trending Videos: 0
Channel: Canadian Space Agency, 60th Percentile: 11216.4
Number of Trending Videos: 0
Channel: SpaceX, 60th Percentile: 762193.6
Number of Trending Videos: 1
Channel: NASA, 60th Percentile: 20847.0
Number of Trending Videos: 92
Distribution before SMOTE:
0    11733
1      160
Name: is_trending, dtype: int64
Distribution after SMOTE:
0    9391
1    9391
Name: is_trending, dtype: int64
```

4.4.3 Training the Classifier

Here I trained my Random Forest Classifier on the balanced training set and evaluated the model on the test set. I then proceeded to predict the target values on the original test set (*X_test_imputed*) and calculate the accuracy of the model's predictions compared to the true labels using the *accuracy_score* function. Finally, we will generate a classification report using the *classification_report* function, which provides additional metrics beyond accuracy, such as precision, recall, F1-score, and support, for each class (in this case, 0 and 1).

```
#for smote
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Initialize the Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)

# Train the classifier on the balanced training set
rf_classifier.fit(X_train_resampled, y_train_resampled)

# Predict on the original test set to evaluate the model
y_test_pred = rf_classifier.predict(X_test_imputed)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_test_pred)
print(f"Test Accuracy: {accuracy}")

# Generate a classification report
class_report = classification_report(y_test, y_test_pred)
print("Classification Report:")
print(class_report)
```

4.4.4 LIME Algorithm

Here I used LIME to explain the trend forecasting for the selected video instance which in our case is Chandrayaan-3 Mission Soft-landing LIVE Telecast and helped us understand why it was predicted as class 1.

```
# Create a LimeTabularExplainer
explainer = LimeTabularExplainer(X_train_resampled, mode="classification", training_labels=y_train_resampled)

# Generate an explanation for the selected instance
explanation = explainer.explain_instance(instance_features, rf_classifier.predict_proba)
```

4.5 Sentiment Analysis

4.5.1 Fetch Comments as a list

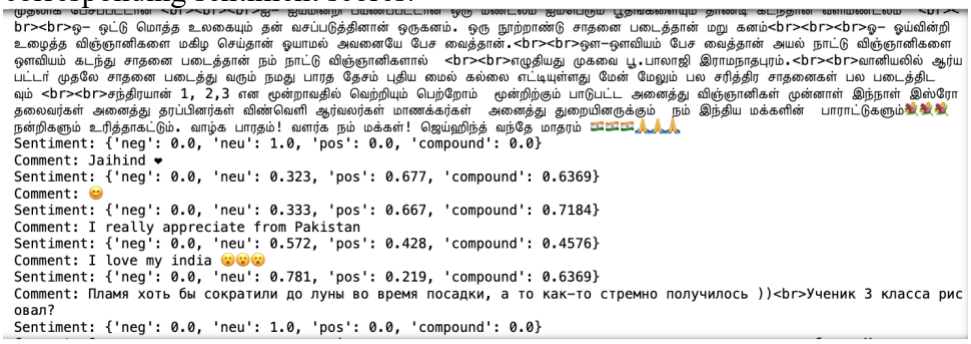
get_comments retrieve all comments from a specified video and returns them as a list. For each item in the response, it extracts the actual text of the comment and appends it to the *comments* list.

```
video_id = 'DLA_64yz8Ss'  # Replace with your desired video ID
comments = get_comments(youtube, video_id)
```

4.5.2 VADER on each comment

This part of the code utilizes SentimentIntensityAnalyzer to compute sentiment scores. The *polarity_scores* method returns a dictionary with four entries: 'neg' for negative, 'neu' for neutral,

'pos' for positive, and 'compound' for a composite score. For each comment, it prints the comment text and its corresponding sentiment scores.



After this I used *vader_predict* function to further use it in LIME XAI . It will take a list of text inputs, analyze each using VADER for sentiment analysis, and then convert the compound sentiment scores into a format suitable for LIME.

4.5.3 Categorization

Here I categorized comments into different sentiment groups based on their VADER sentiment scores.

```
for comment in comments:
    score = analyzer.polarity_scores(comment)['compound']
    if score <= -0.5:
        categories['strongly_negative'].append(comment)
    elif -0.5 < score < 0:
        categories['mildly_negative'].append(comment)
    elif 0 <= score <= 0.5:
        categories['neutral'].append(comment)
    elif 0.5 < score < 0.9:
        categories['mildly_positive'].append(comment)
    else:
        categories['strongly_positive'].append(comment)
```

4.5.4 LIME algorithm for comments from a video

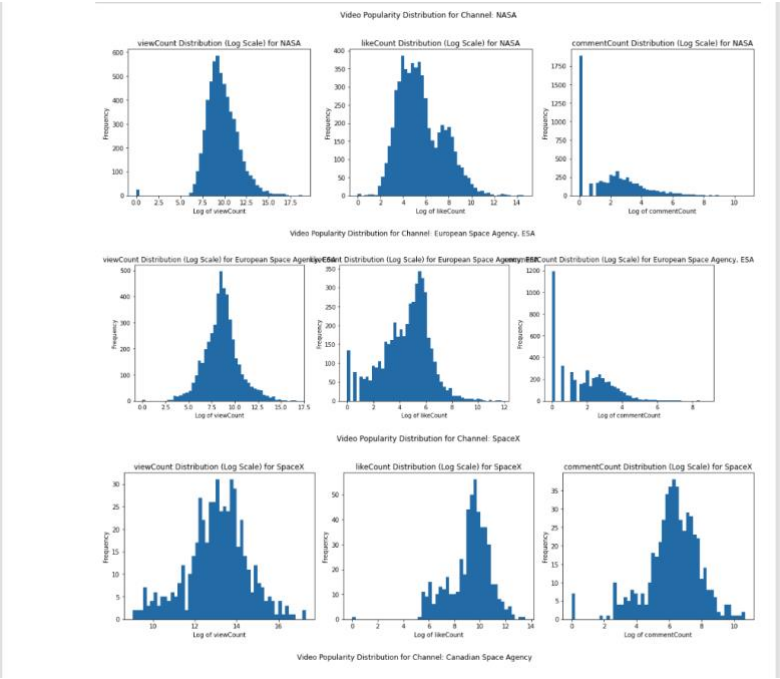
Here I used LIME to fetch comments from the Chandrayaan-3 Mission Soft-landing LIVE Telecast as LIME works in an instance by instance manner. I have coded such that it selects one comment from each category shown above. This way we get to understand the reasoning behind each category.

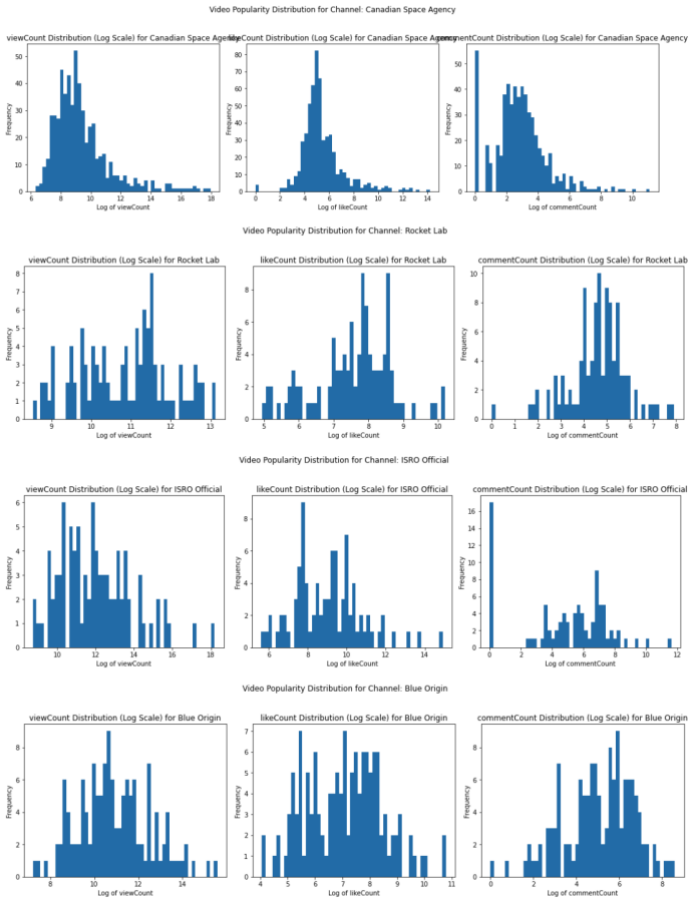
```
for category, comment in selected_comments.items():
    print(f"\nAnalyzing comment from category: {category}")
    print(f"Comment: {comment}")

    try:
        exp = explainer.explain_instance(comment, vader_predict, num_features=10)
        exp.show_in_notebook(text=True)
    except Exception as e:
        print(f"Error in explaining comment from {category}: {e}")
```

5. Lessons Learned

In this section I will discuss about the results achieved in this project.

5.1 Visualization of Channel Statistics using Histograms





From the above graph we can make the below conclusions. These histograms are plotted on a logarithmic scale. Here's a summary based on the description:

6.1.1 View Count Distribution

The data presented by NASA exhibits a prominent peak in the lower logarithmic values, showing a notable occurrence of videos with modest viewership. Additionally, the data reveals a prolonged

tail, suggesting the existence of a minority of videos with considerably higher viewership. Both the European Space Agency (ESA) and the Canadian Space Agency (CSA) demonstrate a comparable trend characterized by peaks in moderate view ranges. This suggests a continuous level of viewing, albeit with a lower number of videos attaining exceptionally high views in comparison to NASA. SpaceX exhibits a diverse range of audience, as evidenced by its histogram extending significantly on the logarithmic scale. This shows a wider distribution of view counts, encompassing videos with exceptionally high popularity. Rocket Lab, ISRO Official, and Blue Origin all have histograms with narrower peaks. This means that the views are more evenly spread out in the moderate range, with fewer instances of very high viewing.

5.1.2 Like Count Distribution

The histograms depicting the distribution of likes across all channels typically exhibit a prominent peak between the lower to middle range of the logarithmic scale. This observation suggests that most videos tend to accumulate a modest amount of likes in relation to their respective view counts. The like count histogram of SpaceX exhibits a greater dispersion, indicating that the videos of this company acquire a more diverse range of likes. This observation may perhaps be associated with the varying view counts of these videos. Additional channels such as the European Space Agency (ESA), the Canadian Space Agency, and the ISRO Official have a somewhat narrower spectrum of likes, indicating a more uniform degree of user interaction in terms of likes received by every video.

5.1.3 Comment Count Distribution

The histograms depicting comment counts across the various channels have a greater degree of right-skewness in comparison to the histograms representing views and likes. This suggests that a significant proportion of videos tend to receive a lower number of comments. The comment distribution observed in NASA's data exhibits a notable concentration of comments at the lower end of the scale. However, it is worth noting that there are also movies that have garnered a substantial number of comments, indicating a broad range of involvement levels. Channels such as the Canadian Space Agency and ISRO Official have a notable prevalence of films with a reduced number of comments. Additionally, the histograms for these channels do not expand significantly towards higher logarithmic values, suggesting a relative scarcity of highly discussed movies in their content repertoire.

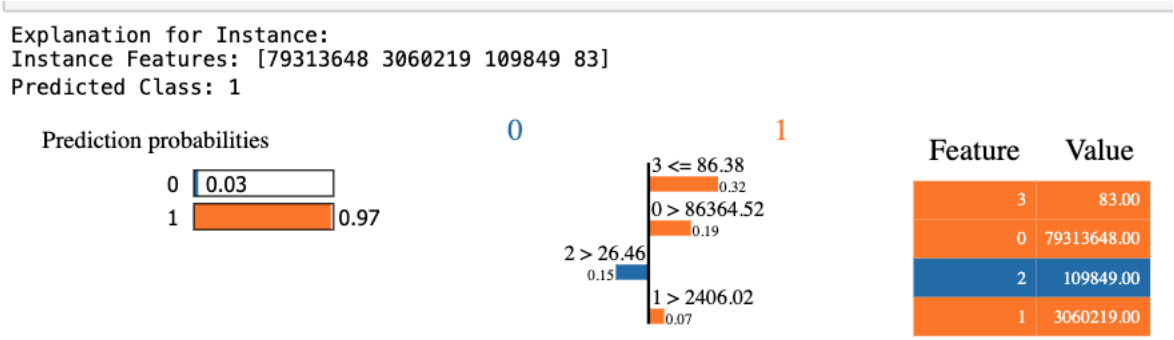
5.2 Conclusion from trend forecasting

My classifier aims to predict whether a video is trending or not based on historical data. Because the dataset collected has unbalanced skewness, so I used SMOTE to balance the dataset where certain classes were underrepresented. I have used a RandomForestClassifier to extract important features to identify the most influential ones in predicting a video’s trending status. And then I proceeded to calculate the accuracy and classification report. In our case we had a high accuracy of 99.83% and a balanced classification report which suggests that the model is highly effective at classifying given data. This could either be because the model is accurate in real life or could be wrong due to the dataset being manipulated due to SMOTE. For the non-trending class (label 0) and trending class (label 1), the model shows perfect precision and recall, indicating that it can almost flawlessly identify both types of videos. The F1-score, which is also very high for both classes, indicates robust model performance. The macro average F1-score and the weighted average F1-score confirms that the model's performance is consistent across the dataset. However, given the imbalance in the dataset , it is important to ensure that the model is not overfitting and that these metrics are not artificially high due to class imbalance.

```
Test Accuracy: 0.9983186212694409
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2342
           1       0.95      0.95      0.95        37

    accuracy                           1.00      2379
   macro avg       0.97      0.97      0.97      2379
weighted avg       1.00      1.00      1.00      2379
```

5.3 LIME Analysis on Trend Forecasting

LIME provides local interpretability. Here we can see which features contribute to a video being classified as trending or not. If LIME reveals that predictions are based on sensible features, we can have more confidence in the model. If LIME shows that a model is using irrelevant features to make predictions, this could be a sign of data leakage, overfitting, or the need for better preprocessing. Feature 0,1,2 and 3 are view count., like count, comment count and the number of days since the video was published respectively. The video under analysis is the globally most-watched live stream video: Chandrayaan-3 Mission Soft-landing LIVE Telecast. The model correctly predicts this video as Class 1 indicating that it is trending with a prediction confidence of 97%. The bar chart indicates the weight of each feature in the model's prediction. It clearly depicts how feature0 i.e., view count is most substantial positive influence. In conclusion, LIME explanation suggests that the view count is the most significant predictor of whether a video is trending, with likes and comments also being important but to a lesser degree. This type of analysis is useful for understanding the decision-making process of machine learning models, especially in scenarios where transparency and interpretability are crucial.



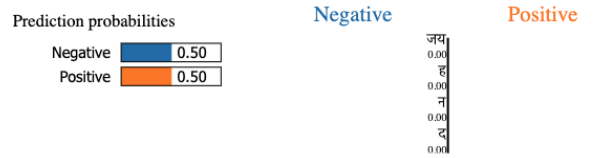
5.4 LIME conclusion from sentiment analysis

The visualizations below depict the LIME text classification explanation for different comments categorized by sentiment. For each comment, we see the predicted probabilities for negative and positive classes, along with the text of the comment with certain words highlighted. These highlights indicate the words that contribute most to the model's prediction. Here's a summary for each comment:

Neutral Comment	Mildly Negative Comment	Mildly Positive Comment	Strongly Negative Comment	Strongly Positive Comment
Uncertain giving equal probabilities (0.50) to both negative and positive classes.	Predicts a negative class with a higher probability (0.66) over the positive class (0.34).	Leans towards a positive prediction (0.80) rather than negative (0.20).	Prediction probability for the negative class is high (0.79), indicating strong confidence in a negative sentiment.	Very confident in a positive prediction (0.99) as opposed to negative (0.01).

In conclusion, these LIME explanations give insights into which words are most influential in the sentiment analysis model's predictions. They help in understanding why a comment is classified as positive or negative, providing transparency into the model's behavior.

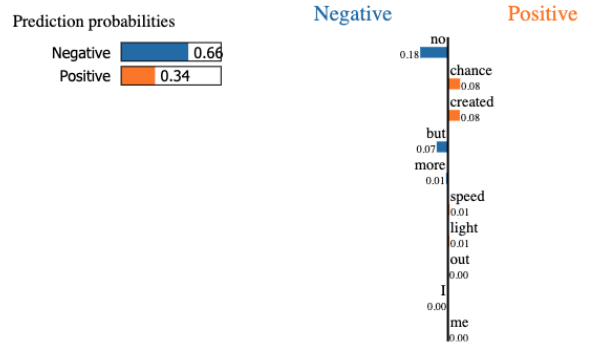
```
Analyzing comment from category: neutral
Comment: जय हिन्द
Debug: Predictions format: (5000, 2) [[0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]
 [0.5 0.5]]
```



Text with highlighted words

जय हिन्द

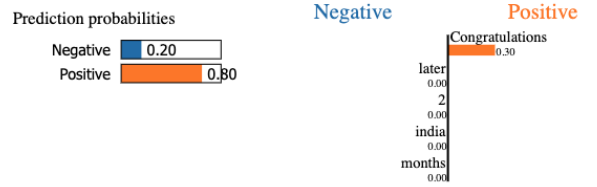
```
Analyzing comment from category: mildly_negative
Comment: Give 1 chance I can created reverse magnetic field and out said travel 10x more light speed 🙂 but no one
believe me
Debug: Predictions format: (5000, 2) [[0.66495 0.33505]
 [0.38765 0.61235]
 [0.67905 0.32095]
 [0.38765 0.61235]
 [0.44275 0.55725]]
```



Text with highlighted words

Give 1 chance I can created reverse magnetic field and out said travel 10x more light speed 🙂 but no one believe me

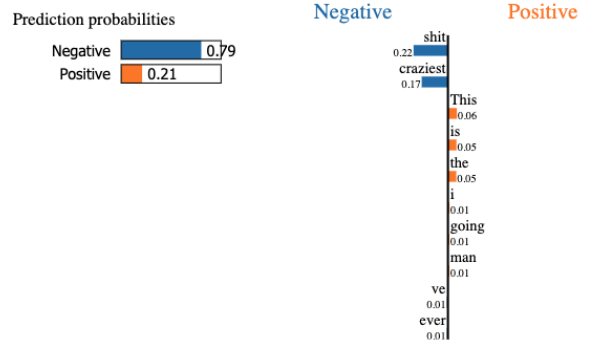
```
Analyzing comment from category: mildly_positive
Comment: Congratulations india, 2 months later .
Debug: Predictions format: (5000, 2) [[0.2003 0.7997]
 [0.5    0.5   ]
 [0.5    0.5   ]
 [0.2003 0.7997]
 [0.5    0.5   ]]
```



Text with highlighted words

Congratulations india, 2 months later .

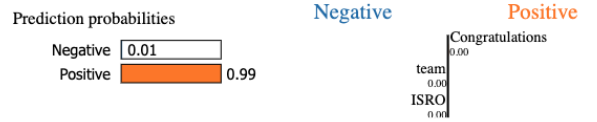
```
Analyzing comment from category: strongly_negative
Comment: This is the craziest shit i&#39;ve ever seen man what is going on
Debug: Predictions format: (5000, 2) [[0.79295 0.20705]
 [0.5     0.5    ]
 [0.7787  0.2213 ]
 [0.1938  0.8062 ]]
```



Text with highlighted words

This is the craziest shit i#39;ve ever seen man what is going on

```
Analyzing comment from category: strongly_positive
Comment: Congratulations ISRO team ❤❤❤❤❤
Debug: Predictions format: (5000, 2) [[0.0102  0.9898 ]
 [0.0102  0.9898 ]
 [0.01405 0.98595]
 [0.0102  0.9898 ]
 [0.01405 0.98595]]
```



Text with highlighted words

Congratulations ISRO team ❤❤❤❤❤