Trending Lies: How TikTok Followers and Engagement Drive Misinformation

Data Curation by Mayar Abu latifeh

Contents

1. Data Sourcing

Dataset Name	Description	Number of Columns	Number of Rows	Size	Source
tiktok_dataset.csv (Primary dataset)	Captures engagement metrics and characteristics of flagged claim-based videos.	11	19,383	2,908 KB	Kaggle- Find here
Top_Influencers.csv	Details about top 50 tiktokers all over the world.	7	50	4KB	Kaggle- Find here

2. Data Profiling

I conducted a detailed profiling process on the TikTok dataset using Python. My goal was to understand its structure, identify missing or inconsistent data, and uncover relationships between variables to prepare it for analysis.

1. Structure Discovery

I started by examining the structure of the dataset.

- **Dataset Size**: The dataset contains 19,382 rows and 11 columns, confirmed through shape inspection.
- **Column Review**: I listed all column names to understand the attributes, such as claim_status, video_id, and video_view_count.
- **Preview**: Viewing the first few rows gave me an idea of the data types and values.
- I dropped the transcription column early on, because I wont be using it in my analysis.
- **Summary Information**: I used Info() to check data types, non-null counts, and memory usage.

```
tiktok_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19382 entries, 0 to 19381
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   claim_status              19084 non-null  object 
 1   video_id                  19382 non-null  int64  
 2   video_duration_sec        19382 non-null  int64  
 3   verified_status           19382 non-null  object 
 4   author_ban_status         19382 non-null  object 
 5   video_view_count          19084 non-null  float64
 6   video_like_count          19084 non-null  float64
 7   video_share_count         19084 non-null  float64
 8   video_download_count      19084 non-null  float64
 9   video_comment_count       19084 non-null  float64
dtypes: float64(5), int64(2), object(3)
memory usage: 1.5+ MB
```

- **Descriptive Statistics**: I calculated metrics like mean, min, max, and percentiles for numerical columns to understand their distributions.

```
[8]: categorical=tiktok_data.select_dtypes(include=['object'])
numerical=tiktok_data.select_dtypes(include=['float', 'int'])

# Descriptive statistics
numerical.describe()
```

[8]:

	video_id	video_duration_sec	video_view_count	video_like_count	video_share_count	video_download_count	video_comment_count
count	1.938200e+04	19382.000000	19084.000000	19084.000000	19084.000000	19084.000000	19084.000000
mean	5.627454e+09	32.421732	254708.558688	84304.636030	16735.248323	1049.429627	349.312146
std	2.536440e+09	16.229967	322893.280814	133420.546814	32036.174350	2004.299894	799.638865
min	1.234959e+09	5.000000	20.000000	0.000000	0.000000	0.000000	0.000000
25%	3.430417e+09	18.000000	4942.500000	810.750000	115.000000	7.000000	1.000000
50%	5.618664e+09	32.000000	9954.500000	3403.500000	717.000000	46.000000	9.000000
75%	7.843960e+09	47.000000	504327.000000	125020.000000	18222.000000	1156.250000	292.000000
max	9.999873e+09	60.000000	999817.000000	657830.000000	256130.000000	14994.000000	9599.000000

- **Unique Values:** I identified video_id as the unique identifier for the dataset, ensuring each row corresponds to a distinct video.

```
9]: unique = [col for col in tiktok_data.columns if tiktok_data[col].nunique() == len(tiktok_data)]
unique

9]: ['video_id']
```

2. Content Discovery:

I looked deeper into the content to identify issues:

- **Missing Values**: I found missing data in 298 rows, mainly in claim_status and video_view_count. Most of these null values were located in the last rows of my dataset.

```
1]: nulls=tiktok_data.isna().sum()
    print("Number of Nulls in Tiktok dataset:\n",nulls)

    Number of Nulls in Tiktok dataset:
     claim_status             298
    video_id                   0
    video_duration_sec         0
    verified_status            0
    author_ban_status          0
    video_view_count         298
    video_like_count         298
    video_share_count        298
    video_download_count     298
    video_comment_count      298
    dtype: int64

2]: missing_values = tiktok_data[tiktok_data.isna().any(axis=1)]
    print("Shape: ", missing_values.shape)

    Shape:  (298, 10)
```

- **Zero Values**: Columns like video_download_count had zero values. However, due to the nature of my data, this is normal, as it indicates that some videos simply didn't get views. These values are still relevant to my analysis.

```
4]: zero_values = (tiktok_data == 0).sum(axis=0)
    print("Zero values in each column:")
    print(zero_values)

    Zero values in each column:
    claim_status               0
    video_id                   0
    video_duration_sec         0
    verified_status            0
    author_ban_status          0
    video_view_count           0
    video_like_count           4
    video_share_count         99
    video_download_count     977
    video_comment_count     3434
    dtype: int64
```

- **Duplicate Detection**: I confirmed there were no duplicate rows in the dataset.

3. Relationship Discovery:

- **Encoding Categorical Features**: I encoded categorical features like claim_status, verified_status, and author_ban_status to convert them into numerical formats. This step was essential for ensuring these features could be included in correlation analysis and other numerical computations.

```
for col in categorical:
    print(col+ " values: ",categorical[col].unique())
```

```
claim_status values:  ['claim' 'opinion' nan]
verified_status values:  ['not verified' 'verified']
author_ban_status values:  ['under review' 'active' 'banned']
```

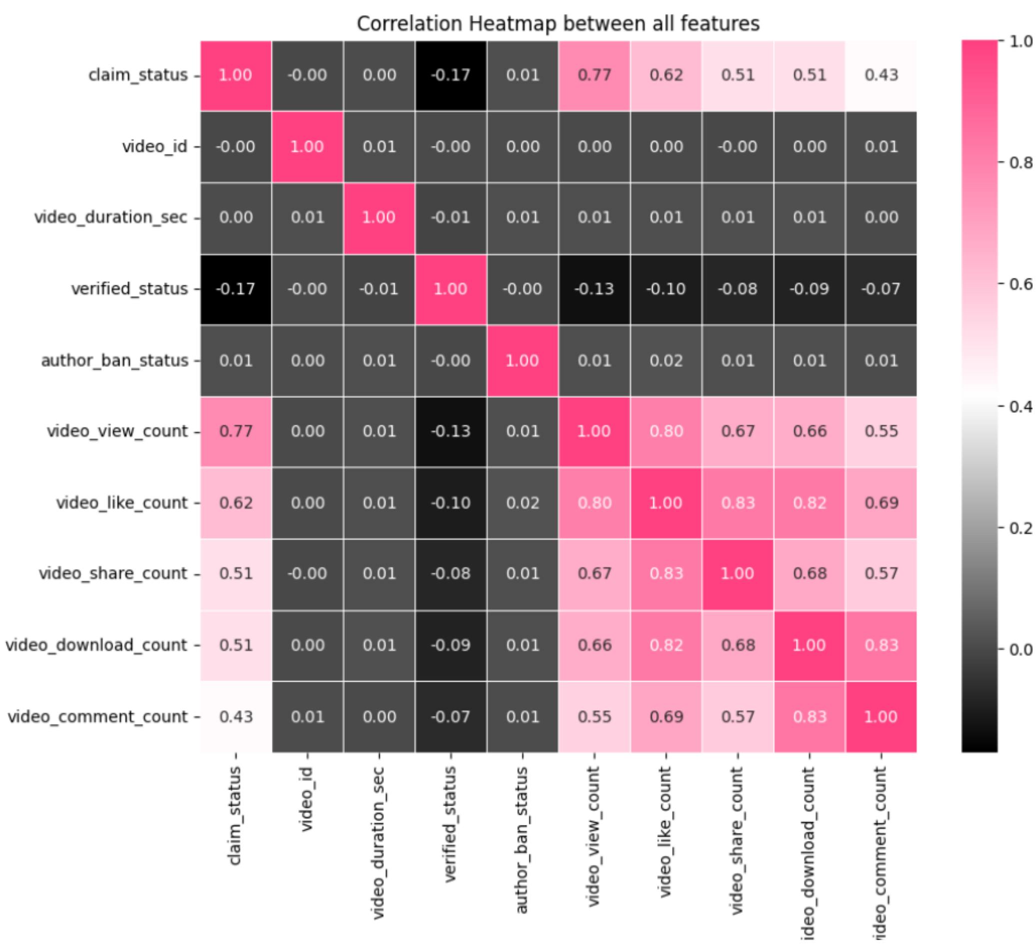
```
encoding_dict = {
    "claim_status": {"claim": 1, "opinion": 0, np.nan: -1},
    "verified_status": {"not verified": 0, "verified": 1},
    "author_ban_status": {"under review": 0, "active": 1, "banned": 2}
}
# Replacing original columns with encoded values
for col, mapping in encoding_dict.items():
    data2[col] = data2[col].map(mapping)

data2.head()
```

	claim_status	video_id	video_duration_sec	verified_status	author_ban_status
0	1	7017666017	59	0	0
1	1	4014381136	32	0	1
2	1	9859838091	31	0	1
3	1	1866847991	25	0	1
4	1	7105231098	19	0	1

- **Correlation Matrix**: I calculated the correlation coefficients between alll columns using the .corr() function. This helped quantify the strength and direction of the relationships between variables. For example:

- video_view_count and video_like_count showed a strong positive correlation (0.80), indicating that videos with more views tend to have more likes.
- video_download_count and video_share_count also had a strong correlation (0.68), suggesting that shared videos are more likely to be downloaded.

- **Heatmap Visualization**: I used a heatmap to visually represent these correlations. The heatmap highlighted key relationships, such as the clustering of interactions between video_view_count, video_like_count, and video_share_count. Additionally, the relationship between claim_status and video_view_count was evident, reinforcing its relevance to my main idea. This visualization made it easier to identify patterns that might influence my future analyses.



3. Data Wrangling

In this phase, I focused on cleaning and transforming the TikTok dataset to prepare it for effective analysis. The wrangling process addressed missing values, outliers, and feature engineering to create a dataset ready for deriving insights.

1. Handling Missing Values

As part of cleaning the dataset, I identified missing values using the .isna().sum() method. It turned out that 298 rows had missing values in critical columns like claim_status and engagement metrics (video_view_count, video_like_count, etc.).

I decided to drop these rows for a few important reasons:

- **Missing Classification Data**: The claim_status column, which indicates whether a video is a claim or an opinion, was missing for these rows. Since this classification is central to my analysis, keeping these rows would add ambiguity without contributing meaningful insights.
- **Irrelevant Engagement Metrics**: These rows also lacked values in key engagement metrics like video_view_count and video_like_count. Without this data, these rows didn't provide value for understanding video performance or engagement trends.

I realized that keeping these rows would only introduce noise to the dataset and reduce the reliability of the analysis. By dropping them, I ensured that the remaining dataset was clean and focused on relevant and complete data.

After dropping the rows, I verified that there were no missing values left in the dataset, allowing me to move forward confidently.

```
tiktok_data=tiktok_data.dropna()
tiktok_data.isna().sum()
```

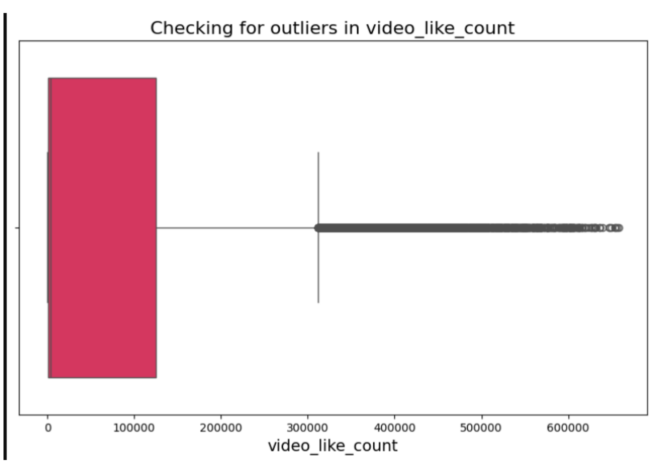
```
claim_status            0
video_id                0
video_duration_sec      0
verified_status         0
author_ban_status       0
video_view_count        0
video_like_count        0
video_share_count       0
video_download_count    0
video_comment_count     0
dtype: int64
```

2. Outlier Detection and Analysis

I used boxplots for numerical columns (video_duration_sec, video_view_count, video_like_count, etc.) to visually identify outliers.

Steps:

- I created a reusable function to generate boxplots for each numerical column.
- Observed significant outliers, especially in video_like_count and video_view_count, where a few videos had exceptionally high engagement metrics.




```
numerical=tiktok_data.select_dtypes(include=['float', 'int'])
```

```
def boxplot(column):
    plt.figure(figsize=(10, 6))
    ax = sns.boxplot(data= tiktok_data, x=numerical[f"{column}"], color="#EE1D52")
    plt.title(f"Checking for outliers in {column}", fontsize=16)
    plt.xlabel(column, fontsize=14)
    plt.show()
```

```
boxplot("video_duration_sec")
print("\n"+"-"*140+"\n")
boxplot("video_view_count")
print("\n"+"-"*140+"\n")
boxplot("video_like_count")
print("\n"+"-"*140+"\n")
boxplot("video_share_count")
print("\n"+"-"*140+"\n")
boxplot("video_comment_count")
print("\n"+"-"*140+"\n")
boxplot("video_download_count")
```

Median vs Mean Analysis to see the influence of outliers:

- I calculated the mean and median for each numerical column to evaluate the influence of outliers.
- I noticed a significant difference between the mean and median in metrics like video_view_count and video_like_count, confirming the impact of outliers.

```
print("Median VS Mean to inspect the affect of outliers\n")
summary_df = numerical.agg(['mean', 'median']).transpose()
summary_df.columns = ['Mean', 'Median']

summary_df
```

Median VS Mean to inspect the affect of outliers

	Mean	Median
video_id	5.624840e+09	5.609500e+09
video_duration_sec	3.242381e+01	3.200000e+01
video_view_count	2.547086e+05	9.954500e+03
video_like_count	8.430464e+04	3.403500e+03
video_share_count	1.673525e+04	7.170000e+02
video_download_count	1.049430e+03	4.600000e+01
video_comment_count	3.493121e+02	9.000000e+00

Rather than dropping these outliers, I decided to retain them because they represent viral videos, which are critical for understanding engagement trends. Removing these outliers would have excluded valuable insights into what makes certain videos extraordinarily popular. To manage this effectively, I created a Video Classification feature, categorizing videos into levels of engagement (Unpopular, Typical, Popular, Viral). This allowed me to highlight and analyze outliers without compromising the overall structure of the dataset.

3. Handling Outliers and Creating New Features

To handle the outliers in my dataset, I created a new feature called Total Interactions, which aggregates key engagement metrics: video_view_count, video_like_count, video_comment_count, and video_share_count.

This feature also provides a view of overall engagement, offering a clearer understanding of trends while reducing the impact of extreme values.

```
tiktok_data.loc[:,'Total Interactions'] = (
    tiktok_data['video_view_count'] +
    tiktok_data['video_like_count'] +
    tiktok_data['video_comment_count'] +
    tiktok_data['video_share_count']
)
```

After creating the feature, I classified the videos into four distinct categories—Unpopular, Typical, Popular, and Viral—using percentile thresholds (25th, 75th, and 95th percentiles of Total Interactions). These thresholds helped me categorize videos based on their engagement levels.

```
# Calculating percentiles for thresholds
p25 = tiktok_data['Total Interactions'].quantile(0.25)  # 25th percentile
p75 = tiktok_data['Total Interactions'].quantile(0.75)  # 75th percentile
p95 = tiktok_data['Total Interactions'].quantile(0.95)  # 95th percentile


def classify_video(total_interactions):
    if total_interactions >= p95:
        return 'Viral'
    elif total_interactions >= p75:
        return 'Popular'
    elif total_interactions >= p25:
        return 'Typical'
    else:
        return 'Unpopular'

tiktok_data['Video Classification'] =tiktok_data['Total Interactions'].apply(classify_video)
```

This classification approach allowed me to include outliers in a meaningful way. Instead of discarding videos with exceptionally high engagement (e.g., viral videos), I treated them as valuable insights for understanding trends in user interaction and engagement behavior.

```
classification_counts=tiktok_data['Video Classification'].value_counts()
print("Counts of each classification:")
print(classification_counts)
```

```
Counts of each classification:
Video Classification
Typical      9542
Unpopular    4771
Popular      3816
Viral         955
Name: count, dtype: int64
```

By integrating outliers into my analysis through classification, I ensured that my dataset reflected real-world patterns and behaviors while minimizing the distortion caused by extreme values. This step enhanced the interpretability and relevance of my findings, making the data more robust and insightful for further analysis.

4. Datable Schema:

Column Name	Data Type	Description
claim_status	String	Indicates whether the video is a claim or opinion.
video_id	Integer	Unique identifier for each video.
video_duration_sec	Integer	Duration of the video in seconds.
verified_status	String	Indicates whether the author of the video is verified (verified/not verified).
author_ban_status	String	Status of the author account (e.g., active, under review, or banned).
video_view_count	Integer	Number of views the video received.
video_like_count	Integer	Number of likes the video received.
video_share_count	Integer	Number of times the video was shared.
video_download_count	Integer	Number of times the video was downloaded.
video_comment_count	Integer	Number of comments on the video.
Total Interactions	Integer	Sum of all interactions: views, likes, comments, and shares, reflecting overall engagement.

5. Influencers Dataset:

- **Structure and Content Discovery**: I explored the structure and identified key attributes like rank, followers, likes, and country, while noting any missing values.

```
]: influencers_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50 entries, 0 to 49
Data columns (total 7 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Rank                50 non-null     int64
 1   Username            50 non-null     object
 2   Owner               50 non-null     object
 3   Followers(millions) 50 non-null     float64
 4   Likes(billions)     50 non-null     float64
 5   Description         50 non-null     object
 6   Country             37 non-null     object
dtypes: float64(2), int64(1), object(4)
memory usage: 2.9+ KB
```

```
influencers_data.describe()
```

	Rank	Followers(millions)	Likes(billions)
count	50.00000	50.000000	50.000000
mean	25.50000	58.184000	1.762400
std	14.57738	25.190048	1.820001
min	1.00000	38.300000	0.260000
25%	13.25000	43.000000	0.750000
50%	25.50000	50.550000	1.300000
75%	37.75000	62.250000	2.100000
max	50.00000	162.200000	11.500000

- **Handling Missing Values**: I identified missing country values for several influencers. Using researched mappings based on usernames, I filled in the missing data to ensure consistency without assumptions.

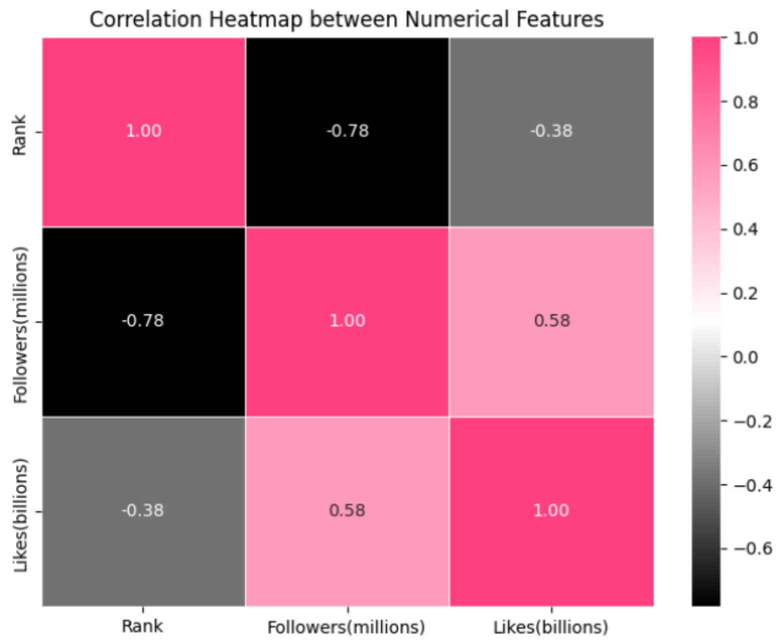
```
nan_countries = influencers_data[influencers_data['Country'].isna()]
nan_countries['Owner']
```

```
4             Addison Rae
10               The Rock
14           Jason Derulo
15         Dixie D'Amelio
18           Kylie Jenner
19             Loren Gray
21           Younes Zarou
33            Joe Albanese
34              Junya Gou
35    Pongámoslo a Prueba
36             Avani Gregg
37           Rod Contreras
38              Ria Ricis
Name: Owner, dtype: object
```

```
#mapping useres to countries based on reserach
username_to_country = {
    '@addisonre': 'United States',
    '@therock': 'United States',
    '@jasonderulo': 'United States',
    '@dixiedamelio': 'United States',
    '@kyliejenner': 'United States',
    '@lorengray': 'United States',
    '@youneszarou': 'Germany',
    '@joealbanese': 'United States',
    '@junya1gou': 'Japan',
    '@pongamoslo_a_prueba': 'Mexico',
    '@avani': 'United States',
    '@elrodcontreras': 'Mexico',
    '@riaricis': 'Indonesia'
}


influencers_data['Country'] = influencers_data.apply(
    lambda row: username_to_country[row['Username']] if pd.isna(row['Country']) else row['Country'],
    axis=1
)
```

- **Outlier Detection**: I created boxplots for numerical fields like followers and likes to detect significant outliers. Instead of removing them, I retained the outliers, as they represent valuable insights about highly successful influencers.

- **Correlation Analysis**: By generating a heatmap, I examined relationships between numerical features, observing notable correlations like followers to likes, which validated trends in engagement.



- **Data Cleaning and Transformation**: I ensured no duplicates or unnecessary zero values existed, maintaining the dataset's integrity for further analysis.
- **Data Table Schema**

Column Name	Data Type	Description
Rank	Integer	Rank of the influencer based on popularity metrics.
Username	String	Social media username of the influencer.
Owner	String	Real name of the influencer.
Followers(millions)	Float	Number of followers the influencer has in millions.
Likes(billions)	Float	Total number of likes received by the influencer’s content in billions.
Description	String	A brief description of the influencer's niche or role.
Country	String	Country associated with the influencer.

Final Steps

I then exported both datasets after cleaning

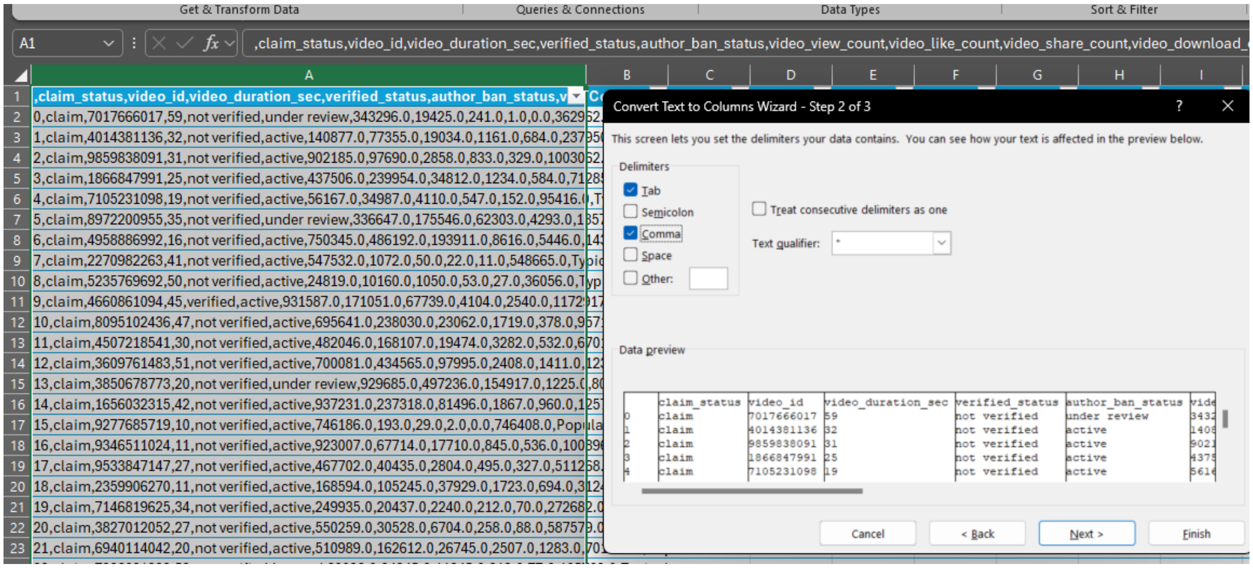
```
: influencers_data.to_csv("clean_influencers_data.csv")
```

And fixed their formats in Excel

A1 ,claim_status,video_id,video_duration_sec,verified_status,author_ban_status

	A	B	C	D	E	F	G	H	I	J	K
1	,claim_status,video_id,video_duration_sec,verified_status,author_ban_status,video_view_count,video_like_cou										
2	0,claim,7017666017,59,not verified,under review,343296.0,19425.0,241.0,1.0,0.0,362962.0,Typical										
3	1,claim,4014381136,32,not verified,active,140877.0,77355.0,19034.0,1161.0,684.0,237950.0,Typical										
4	2,claim,9859838091,31,not verified,active,902185.0,97690.0,2858.0,833.0,329.0,1003062.0,Popular										
5	3,claim,1866847991,25,not verified,active,437506.0,239954.0,34812.0,1234.0,584.0,712856.0,Popular										
6	4,claim,7105231098,19,not verified,active,56167.0,34987.0,4110.0,547.0,152.0,95416.0,Typical										
7	5,claim,8972200955,35,not verified,under review,336647.0,175546.0,62303.0,4293.0,1857.0,576353.0,Typical										
8	6,claim,4958886992,16,not verified,active,750345.0,486192.0,193911.0,8616.0,5446.0,1435894.0,Viral										
9	7,claim,2270982263,41,not verified,active,547532.0,1072.0,50.0,22.0,11.0,548665.0,Typical										
10	8,claim,5235769692,50,not verified,active,24819.0,10160.0,1050.0,53.0,27.0,36056.0,Typical										
11	9,claim,4660861094,45,verified,active,931587.0,171051.0,67739.0,4104.0,2540.0,1172917.0,Popular										
12	10,claim,8095102436,47,not verified,active,695641.0,238030.0,23062.0,1719.0,378.0,957111.0,Popular										
13	11,claim,4507218541,30,not verified,active,482046.0,168107.0,19474.0,3282.0,532.0,670159.0,Typical										
14	12,claim,3609761483,51,not verified,active,700081.0,434565.0,97995.0,2408.0,1411.0,1234052.0,Popular										
15	13,claim,3850678773,20,not verified,under review,929685.0,497236.0,154917.0,1225.0,805.0,1582643.0,Viral										
16	14,claim,1656032315,42,not verified,active,937231.0,237318.0,81496.0,1867.0,960.0,1257005.0,Popular										
17	15,claim,9277685719,10,not verified,active,746186.0,193.0,29.0,2.0,0.0,746408.0,Popular										
18	16,claim,9346511024,11,not verified,active,923007.0,67714.0,17710.0,845.0,536.0,1008967.0,Popular										
19	17,claim,9533847147,27,not verified,active,467702.0,40435.0,2804.0,495.0,327.0,511268.0,Typical										
20	18,claim,2359906270,11,not verified,active,168594.0,105245.0,37929.0,1723.0,694.0,312462.0,Typical										
21	19,claim,7146819625,34,not verified,active,249935.0,20437.0,2240.0,212.0,70.0,272682.0,Typical										
22	20,claim,3827012052,27,not verified,active,550259.0,30528.0,6704.0,258.0,88.0,587579.0,Typical										
23	21,claim,6940114042,20,not verified,active,510989.0,162612.0,26745.0,2507.0,1283.0,701629.0,Popular										
24	22,claim,7986031283,52,not verified,banned,60096.0,34245.0,11345.0,219.0,77.0,105763.0,Typical										
25	23,claim,5733649579,54,not verified,active,750942.0,136252.0,27119.0,2637.0,1206.0,915519.0,Popular										
26	24,claim,2736327047,39,not verified,active,353722.0,39844.0,10399.0,181.0,54.0,404019.0,Typical										
27	25,claim,6792082620,33,not verified,active,506061.0,288150.0,39351.0,2405.0,1134.0,834696.0,Popular										
28	26,claim,2385469584,10,not verified,under review,50892.0,7976.0,2903.0,125.0,78.0,61849.0,Typical										
29	27,claim,6569363811,22,not verified,under review,812056.0,329068.0,3515.0,5200.0,1108.0,1145747.0,Popular										
30	28,claim,6301836558,21,not verified,active,677855.0,332569.0,97961.0,5531.0,2386.0,1110771.0,Popular										
31	29,claim,6854087993,48,not verified,active,74323.0,20940.0,3678.0,274.0,63.0,99004.0,Typical										
32	30,claim,7568304116,55,not verified,active,573518.0,324003.0,125931.0,4737.0,2918.0,1026370.0,Popular										
33	31,claim,7730527836,21,not verified,banned,376637.0,226700.0,69329.0,2060.0,278.0,672944.0,Typical										
34	32,claim,8304906458,23,not verified,active,781955.0,407367.0,65029.0,4274.0,2240.0,1256591.0,Popular										
35	33,claim,7723647550,33,not verified,active,822795.0,390001.0,125607.0,4012.0,2055.0,1340458.0,Viral										
36	34,claim,1260034177,10,not verified,active,418750.0,133645.0,27821.0,2387.0,206.0,580422.0,Typical										
37	35,claim,5407291277,12,not verified,active,535825.0,73157.0,1642.0,1573.0,97.0,610721.0,Typical										
38	36,claim,6844920875,29,not verified,active,924262.0,465230.0,37874.0,8124.0,635.0,1428001.0,Viral										
39	37,claim,9171709060,30,not verified,active,489267.0,110476.0,17911.0,2585.0,90.0,617744.0,Typical										

clean_tiktok_dataset



claim_sta	video_id	video_du	verified_	author_b	video_vie	video_lik	video_sh	video_do	video_con	Total Interactio	Video Classific
claim	7017666017	59	not verified	under reviev	343296	19425	241	1	0	362962	Typical
claim	4014381136	32	not verified	active	140877	77355	19034	1161	684	237950	Typical
claim	9859838091	31	not verified	active	902185	97690	2858	833	329	1003062	Popular
claim	1866847991	25	not verified	active	437506	239954	34812	1234	584	712856	Popular
claim	7105231098	19	not verified	active	56167	34987	4110	547	152	95416	Typical
claim	8972200955	35	not verified	under reviev	336647	175546	62303	4293	1857	576353	Typical
claim	4958886992	16	not verified	active	750345	486192	193911	8616	5446	1435894	Viral
claim	2270982263	41	not verified	active	547532	1072	50	22	11	548665	Typical
claim	5235769692	50	not verified	active	24819	10160	1050	53	27	36056	Typical
claim	4660861094	45	verified	active	931587	171051	67739	4104	2540	1172917	Popular
claim	8095102436	47	not verified	active	695641	238030	23062	1719	378	957111	Popular
claim	4507218541	30	not verified	active	482046	168107	19474	3282	532	670159	Typical
claim	3609761483	51	not verified	active	700081	434565	97995	2408	1411	1234052	Popular
claim	3850678773	20	not verified	under reviev	929685	497236	154917	1225	805	1582643	Viral
claim	1656032315	42	not verified	active	937231	237318	81496	1867	960	1257005	Popular
claim	9277685719	10	not verified	active	746186	193	29	2	0	746408	Popular
claim	9346511024	11	not verified	active	923007	67714	17710	845	536	1008967	Popular
claim	9533847147	27	not verified	active	467702	40435	2804	495	327	511268	Typical
claim	2359906270	11	not verified	active	168594	105245	37929	1723	694	312462	Typical
claim	7146819625	34	not verified	active	249935	20437	2240	212	70	272682	Typical
claim	3827012052	27	not verified	active	550259	30528	6704	258	88	587579	Typical
claim	6940114042	20	not verified	active	510989	162612	26745	2507	1283	701629	Popular
claim	7986031283	52	not verified	banned	60096	34245	11345	219	77	105763	Typical
claim	5733649579	54	not verified	active	750942	136252	27119	2637	1206	915519	Popular
claim	2736327047	39	not verified	active	353722	39844	10399	181	54	404019	Typical
claim	6792082620	33	not verified	active	506061	288150	39351	2405	1134	834696	Popular
claim	2385469584	10	not verified	under reviev	50892	7976	2903	125	78	61849	Typical
claim	6569363811	22	not verified	under reviev	812056	329068	3515	5200	1108	1145747	Popular
claim	6301836558	21	not verified	active	677855	332569	97961	5531	2386	1110771	Popular
claim	6854087993	48	not verified	active	74323	20940	3678	274	63	99004	Typical
claim	7568304116	55	not verified	active	573518	324003	125931	4737	2918	1026370	Popular
claim	7730527836	21	not verified	banned	376637	226700	69329	2060	278	672944	Typical
claim	8304906458	23	not verified	active	781955	407367	65029	4274	2240	1256591	Popular
claim	7723647550	33	not verified	active	822795	390001	125607	4012	2055	1340458	Viral
claim	1260034177	10	not verified	active	418750	133645	27821	2387	206	580422	Typical
claim	5407291277	12	not verified	active	535825	73157	1642	1573	97	610721	Typical
claim	6844920875	29	not verified	active	924262	465230	37874	8124	635	1428001	Viral
claim	9171709060	30	not verified	active	489267	110476	17911	2585	90	617744	Typical