TikTok Claim Classification Project EDA

August 1, 2023

1 TikTok Claim Classification Project EDA

1.0.1 Task 1. Imports, links, and loading

[1]:

```
# Import packages for data manipulation
import pandas as pd
import numpy as np

# Import packages for data visualization
import seaborn as sns
import matplotlib.pyplot as plt
```

[2]:

```
# Load dataset into dataframe
data = pd.read_csv("tiktok_dataset.csv")
```

1.0.2 Task 2: Data exploration and cleaning

[3]:

```
# Display and examine the first few rows of the dataframe
data.head()
```

[3]:

```
   # claim_status    video_id  video_duration_sec  \
0  1        claim  7017666017                  59
1  2        claim  4014381136                  32
2  3        claim  9859838091                  31
3  4        claim  1866847991                  25
4  5        claim  7105231098                  19

                            video_transcription_text verified_status  \
0  someone shared with me that drone deliveries a…    not verified
1  someone shared with me that there are more mic…    not verified
2  someone shared with me that american industria…    not verified
3  someone shared with me that the metro of st. p…    not verified
4  someone shared with me that the number of busi…    not verified

  author_ban_status  video_view_count  video_like_count  video_share_count  \
0      under review           343296.0           19425.0              241.0
1            active           140877.0           77355.0            19034.0
2            active           902185.0           97690.0             2858.0
3            active           437506.0          239954.0            34812.0
4            active            56167.0           34987.0             4110.0

   video_download_count  video_comment_count
0                   1.0                  0.0
1                1161.0                684.0
2                 833.0                329.0
3                1234.0                584.0
4                 547.0                152.0
```

[4]:

```
# Get the size of the data
data.size
```

[4]:

```
232584
```

[5]:

```
# Get the shape of the data
data.shape
```

[5]:

```
(19382, 12)
```

[6]:

```
# Get basic information about the data
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19382 entries, 0 to 19381
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   #                         19382 non-null  int64
 1   claim_status              19084 non-null  object
 2   video_id                  19382 non-null  int64
 3   video_duration_sec        19382 non-null  int64
 4   video_transcription_text  19084 non-null  object
 5   verified_status           19382 non-null  object
 6   author_ban_status         19382 non-null  object
 7   video_view_count          19084 non-null  float64
 8   video_like_count          19084 non-null  float64
 9   video_share_count         19084 non-null  float64
 10  video_download_count      19084 non-null  float64
 11  video_comment_count       19084 non-null  float64
dtypes: float64(5), int64(3), object(4)
memory usage: 1.8+ MB
```

[7]:

```
# Generate a table of descriptive statistics
data.describe()
```

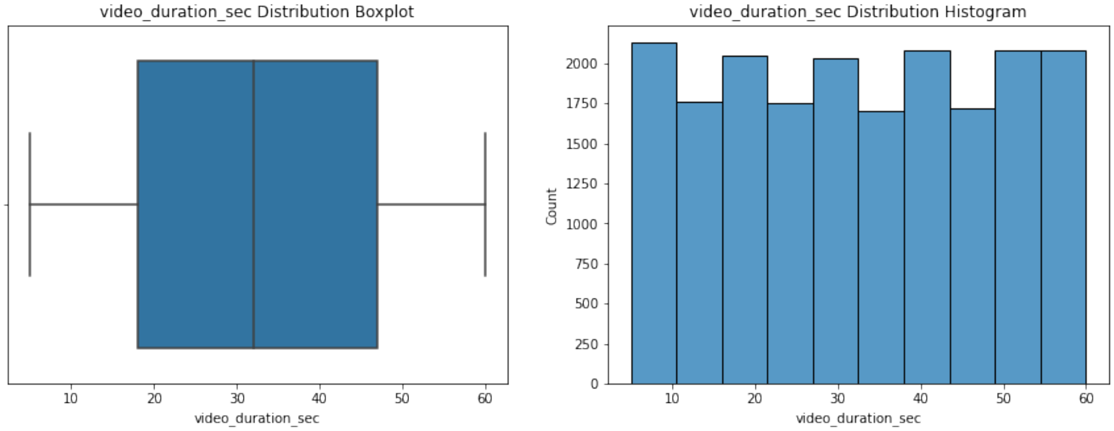
[7]:

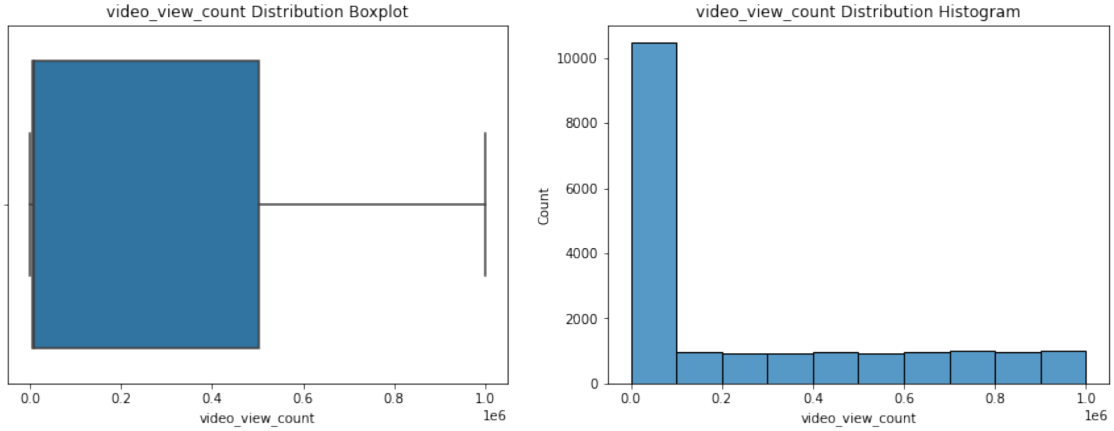
```
                  #      video_id  video_duration_sec  video_view_count  \
count  19382.000000  1.938200e+04        19382.000000      19084.000000
mean    9691.500000  5.627454e+09           32.421732     254708.558688
std     5595.245794  2.536440e+09           16.229967     322893.280814
min        1.000000  1.234959e+09            5.000000         20.000000
25%     4846.250000  3.430417e+09           18.000000       4942.500000
50%     9691.500000  5.618664e+09           32.000000       9954.500000
75%    14536.750000  7.843960e+09           47.000000     504327.000000
max    19382.000000  9.999873e+09           60.000000     999817.000000

       video_like_count  video_share_count  video_download_count  \
count      19084.000000       19084.000000          19084.000000
mean       84304.636030       16735.248323           1049.429627
std       133420.546814       32036.174350           2004.299894
min            0.000000           0.000000              0.000000
25%          810.750000         115.000000              7.000000
50%         3403.500000         717.000000             46.000000
75%       125020.000000       18222.000000           1156.250000
max       657830.000000      256130.000000          14994.000000

       video_comment_count
count         19084.000000
mean            349.312146
std             799.638865
min               0.000000
25%               1.000000
50%               9.000000
75%             292.000000
max            9599.000000
```

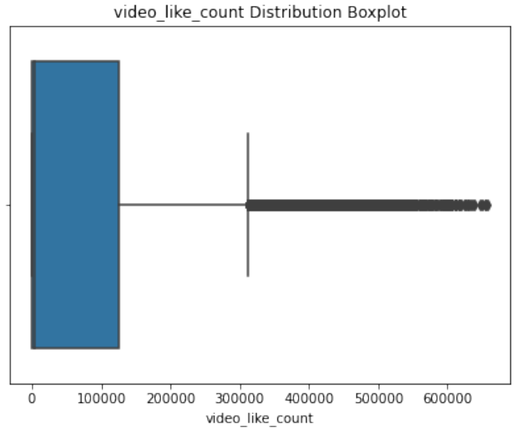
1.0.3 Task 3. Build visualizations

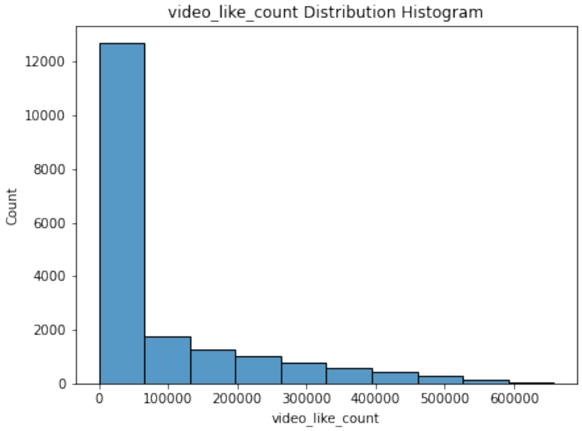
[12]:

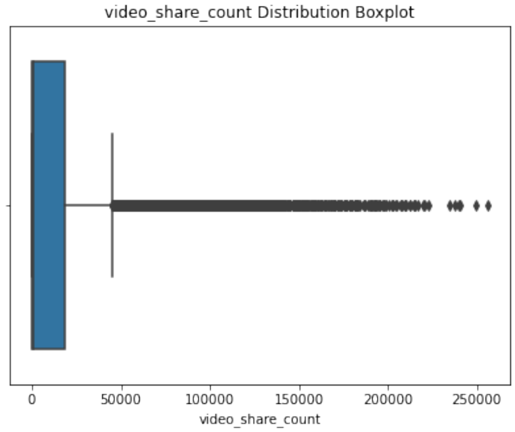
```
def plot_boxplots_and_histograms(dataframe, numeric_columns):
    # Loop through each numeric column in the list.
    for column in numeric_columns:
        # Create a subplot with two side-by-side axes to plot the boxplot and␣
 ↪histogram.
        fig, ax = plt.subplots(1, 2, figsize=(15, 5))

        # Boxplot: visualize the distribution of the column using a boxplot.
        sns.boxplot(dataframe[column], ax=ax[0])
        ax[0].set_title(f"{column} Distribution Boxplot")
        # Histogram: visualize the distribution of the column using a histogram with 10 bins.
        sns.histplot(dataframe[column], bins=10, ax=ax[1])
        ax[1].set_title(f"{column} Distribution Histogram")
        plt.show()

# List of numeric columns to be plotted.
numeric_columns = ['video_duration_sec', 'video_view_count', 'video_like_count', 'video_share_count',
                   'video_download_count', 'video_comment_count']

# Call the function to plot boxplots and histograms for the specified columns in the 'data' DataFrame.
plot_boxplots_and_histograms(data, numeric_columns)
```

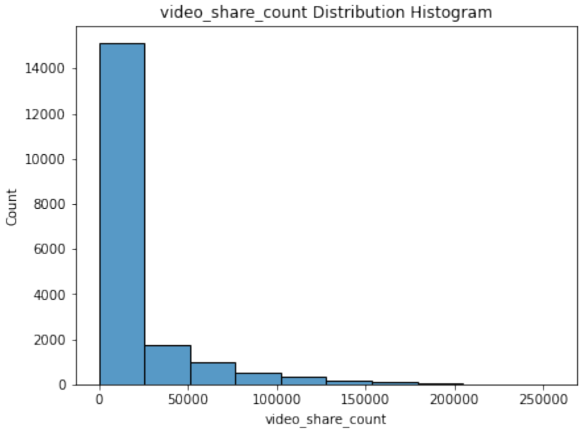


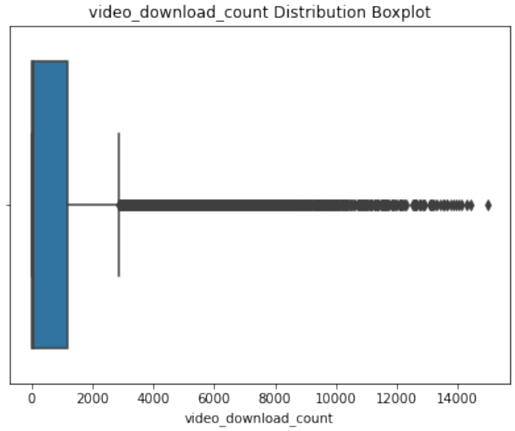


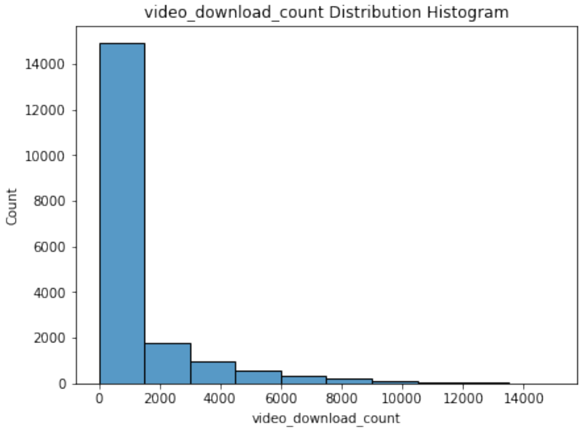

video_like_count Distribution Boxplot
0
100000
200000
300000
400000
500000
600000
video_like_count

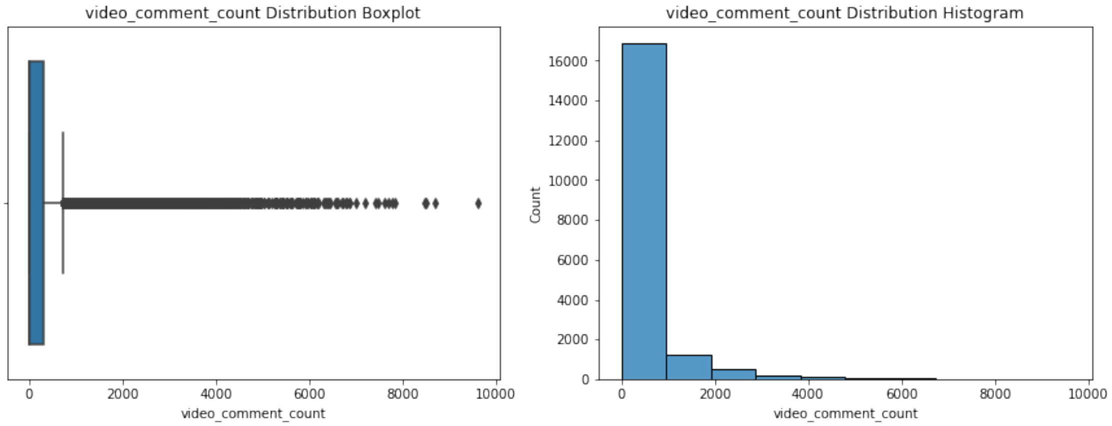

video_like_count Distribution Histogram
Count
0
2000
4000
6000
8000
10000
12000
0
100000
200000
300000
400000
500000
600000
video_like_count


video_share_count Distribution Boxplot
0
50000
100000
150000
200000
250000
video_share_count


video_share_count Distribution Histogram
Count
0
2000
4000
6000
8000
10000
12000
14000
0
50000
100000
150000
200000
250000
video_share_count


video_download_count Distribution Boxplot
0
2000
4000
6000
8000
10000
12000
14000
video_download_count

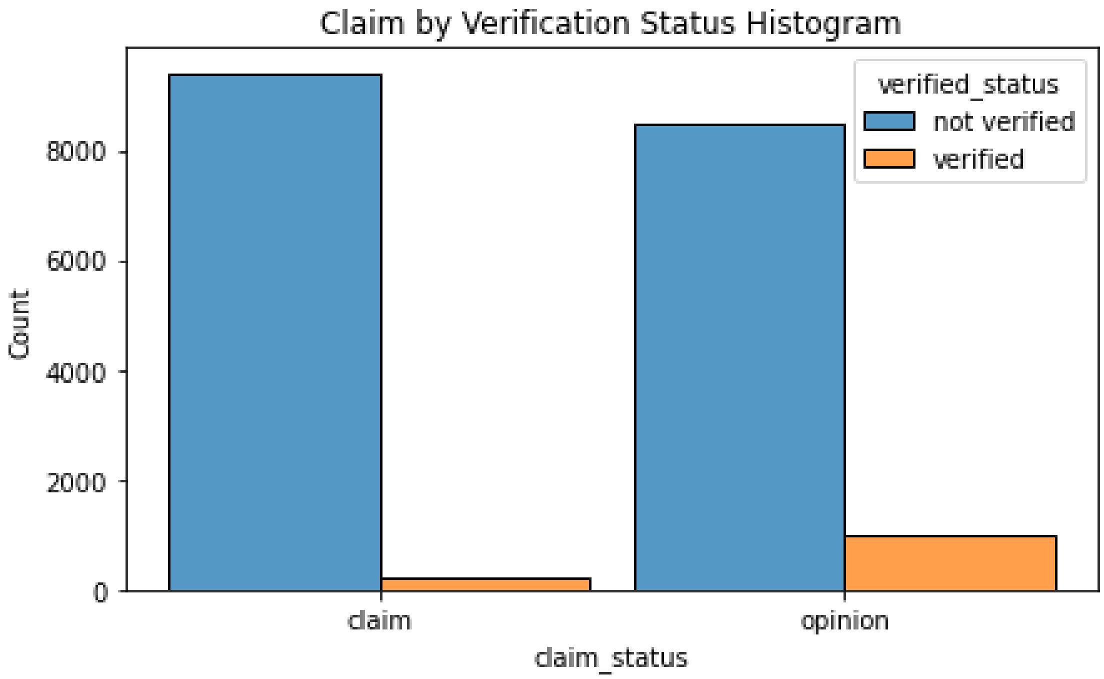

video_download_count Distribution Histogram
Count
0
2000
4000
6000
8000
10000
12000
14000
0
2000
4000
6000
8000
10000
12000
14000
video_download_count



Claim status by verification status

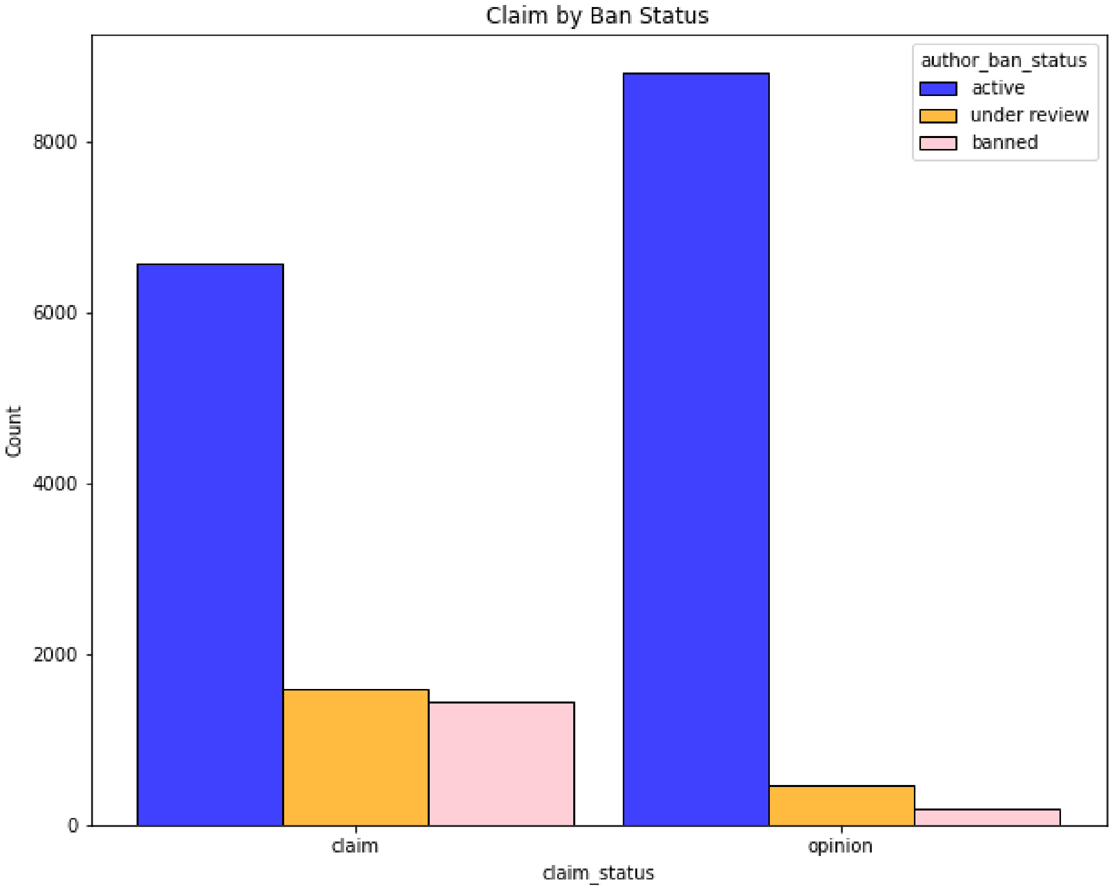
[29]:

```
# Create a histogram with dodge-style bars to compare 'claim_status' based on␣
 ↪'verified_status'.
plt.figure(figsize=(7, 4))
sns.histplot(x=data["claim_status"], hue=data["verified_status"],␣
 ↪multiple="dodge", shrink=0.9)
plt.title("Claim by Verification Status Histogram")
plt.show()
```



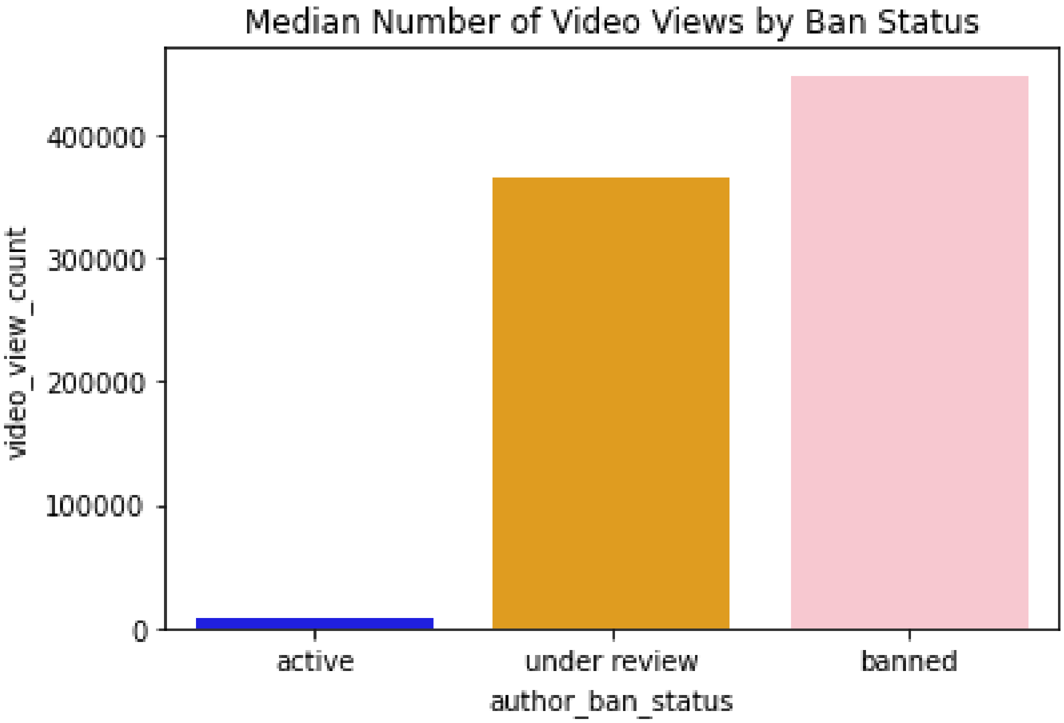
Claim status by author ban status

[21]:
```
# Create a histogram with dodge-style bars to compare 'claim_status' based on 'author_ban_status'.
plt.figure(figsize=(10,8))

# 'order' and 'palette' are optional arguments to customize the order and color palette of the bars.
order=["active", "under review", "banned"]
color_palette={"active":"blue", "under review":"orange", "banned":"pink"}
sns.histplot(data=data, x="claim_status", hue="author_ban_status", multiple="dodge", hue_order=order, palette=color_palette, shrink=0.9)
plt.title("Claim by Ban Status")
plt.show()
```



Median view counts by ban status

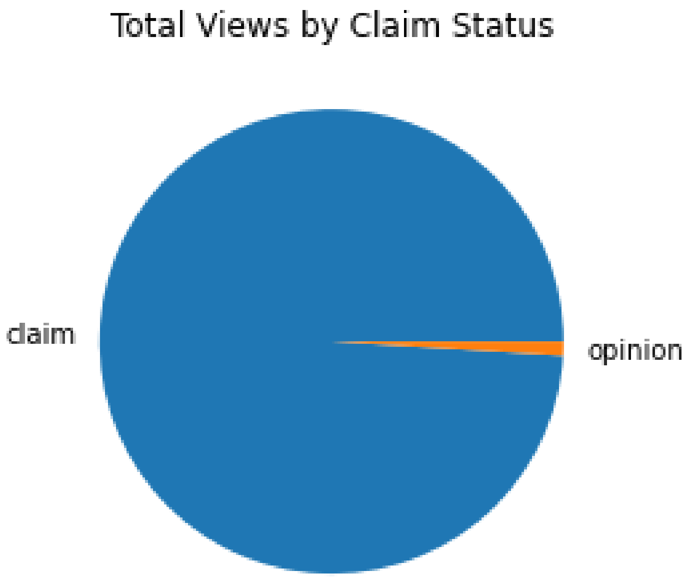
[22]:
```
# Create a bar plot to visualize the median number of video views based on 'author_ban_status'.
# Calculate the median values for numeric columns grouped by␣
↪'author_ban_status'.
ban_status_median = data.groupby(['author_ban_status']).
↪median(numeric_only=True).reset_index()

# 'x' represents the categorical variable 'author_ban_status', 'y' represents␣
↪the numeric variable 'video_view_count'.
sns.barplot(data=ban_status_median, x="author_ban_status",␣
↪y="video_view_count", order=order, palette=color_palette)

# Set the title for the bar plot.
plt.title("Median Number of Video Views by Ban Status")

# Display the plot.
plt.show()
```



Total views by claim status

[25]:

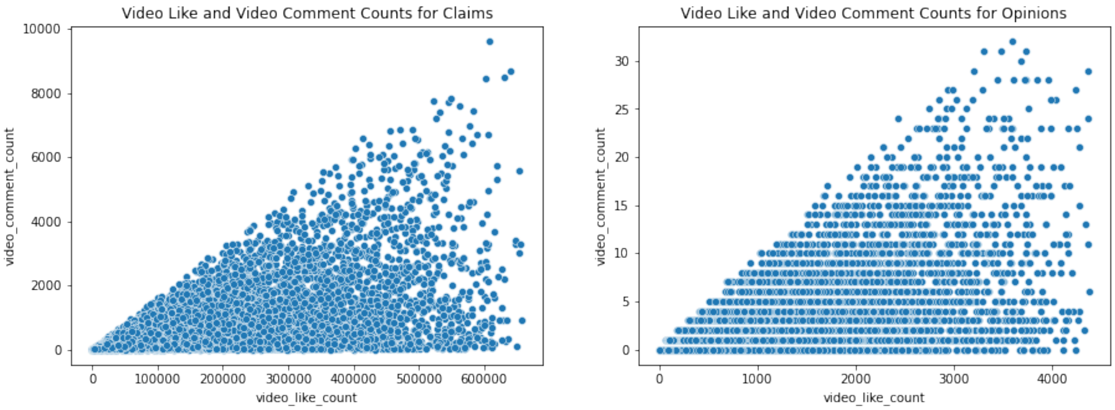
```
# Create a pie graph
plt.pie(data.groupby("claim_status")["video_view_count"].sum(),␣
↪labels=["claim", "opinion"])
plt.title("Total Views by Claim Status")
plt.show()
```



User Interaction by claim_status

[30]:

```
# Create a scatterplot of `video_like_count` versus `video_comment_count` according to 'claim_status'.

# Define the two claim status categories: claim and opinion.
cl_op = ["claim", "opinion"]

# Create subplots with two side-by-side axes to plot the scatterplots.
fig, ax = plt.subplots(1, 2, figsize=(15, 5))

# Loop through each claim status category.
x = 0
for i in cl_op:
    # Filter the data for the current claim status category.
    data_1 = data[data["claim_status"] == i]

    # Create the scatterplot using seaborn's scatterplot function.
    sns.scatterplot(x=data_1["video_like_count"], y=data_1["video_comment_count"], ax=ax[x])

    # Set the title for the scatterplot with the corresponding claim status category.
    ax[x].set_title("Video Like and Video Comment Counts for " + i.capitalize()␣
 ↪+ "s")

    # Increment the subplot index.
    x += 1

plt.show()
```



1.0.4 Task 4. Determine outliers

[26]:

```
# Define a list of column names to analyze for outliers.
col_names = ['video_view_count',
             'video_like_count',
             'video_share_count',
             'video_download_count',
             'video_comment_count']

# Loop through each column in the list to detect outliers using the IQR method.
for column in col_names:
    # Calculate the first quartile (Q1), third quartile (Q3), and interquartile␣
 ↪range (IQR) for the column.
    q1 = data[column].quantile(0.25)
    q3 = data[column].quantile(0.75)
    iqr = q3 - q1

    # Calculate the median value for the column.
    median = data[column].median()

    # Define the threshold for identifying outliers (1.5 times the IQR above␣
 ↪the third quartile).
    outlier = median + 1.5 * iqr
    # Count the number of outliers in the column (values greater than the outlier threshold).
    outlier_num = data[data[column] > outlier].shape[0]

    # Print the number of outliers for the current column.
    print(f"Number of outliers, {column}: {outlier_num}")
```

```
Number of outliers, video_view_count 2343
Number of outliers, video_like_count 3468
Number of outliers, video_share_count 3732
Number of outliers, video_download_count 3733
Number of outliers, video_comment_count 3882
```

1.0.5 Task 5. Conclusion

- Exploratory data analysis (EDA) delves into the impact of videos on TikTok users, focusing on engagement metrics like view, like, and comment counts. Key insights from EDA highlight the need to address null values and the imbalance in opinion video counts within the future classification model.

- Verified users are significantly outnumbered by unverified users, but when a user is verified, they are more inclined to post opinion videos.

- Both claim and opinion videos have a higher number of active authors compared to banned or authors under review. However, the proportion of active authors is notably higher for opinion videos than for claim videos, indicating that authors posting claim videos are more likely to come under review or be banned.

- Non-active authors have a substantially higher median view count than active authors. Considering that non-active authors are more likely to post claim videos, and their videos receive more views overall than videos by active authors, the video_view_count could be a valuable indicator of claim status.

- Although the dataset contains a roughly equal number of each video type, claim videos dominate the overall view count.

- Given the dataset's size of approximately 20,000 entries, it's important to note that there are a considerable number of outliers in different columns, ranging between around 2,400 and 3,900. These outliers should be carefully considered when selecting and constructing the machine learning model.

- The EDA reveals crucial aspects to be addressed in the claim classification model, including handling null values and acknowledging data distribution characteristics. By incorporating these insights, we will build an accurate and effective machine learning model for TikTok's claim classification.