1 **TikTok Project By Mohammad Talha Gulzar

Course 5 - Regression Analysis: Simplify complex data relationships

Scenario Presented : As a data professional at TikTok. The data team is working towards building a machine learning model that can be used to determine whether a video contains a claim or whether it offers an opinion. With a successful prediction model, TikTok can reduce the backlog of user reports and prioritize them more efficiently.

The team is getting closer to completing the project, having completed an initial plan of action, initial Python coding work, EDA, and hypothesis testing.

The TikTok team has reviewed the results of the hypothesis testing. TikTok's Operations Lead, Maika Abadi, is interested in how different variables are associated with whether a user is verified. Earlier, the data team observed that if a user is verified, they are much more likely to post opinions. Now, the data team has decided to explore how to predict verified status to help them understand how video characteristics relate to verified users. Therefore, I have been asked to conduct a logistic regression using verified status as the outcome variable. The results may be used to inform the final model related to predicting whether a video is a claim vs an opinion.

2 Regression modeling**

The purpose of this project is to demostrate knowledge of EDA and regression models.

The goal is to build a logistic regression model and evaluate the model. The activity has three parts

Part 1: EDA & Checking Model Assumptions * What are some purposes of EDA before constructing a logistic regression model?

Part 2: Model Building and Evaluation

Part 3: Interpreting Model Results

- What key insights emerged from this model(s)?
- What business recommendations are proposed based on the models built?

3 Building a regression model

3.0.1 Imports and loading**

[18]:
```
# Import packages for data manipulation
import pandas as pd
import numpy as np

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

# Import packages for data preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.utils import resample

# Import packages for data modeling
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

Tiktok Data Set Loading

[2]:
```
# Load dataset into dataframe
data = pd.read_csv("tiktok_dataset.csv")
```

Exploratory Data Analysis (EDA) is essential before building a logistic regression model as it helps in understanding the distribution of variables, identifying missing data, and detecting outliers. It checks for assumptions like multicollinearity to avoid issues that could distort model predictions. EDA also explores the relationships between features and the target variable, guiding the selection of impactful predictors. Additionally, it helps assess the balance of the target classes, which is crucial in classification tasks. By visualizing data patterns and identifying potential issues, EDA ensures the dataset is ready for effective logistic regression modeling.

3.0.2 Exploring Data with EDA**

Inspect the first five rows of the dataframe.

[3]:
```
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]:
```
num1= data.shape[0]
num2 = data.shape[1]
print (num1, num2)
```

```
19382 12
```

[5]:
```
columntypes= data.dtypes
print(columntypes)
```

```
#                             int64
claim_status                 object
video_id                      int64
video_duration_sec            int64
video_transcription_text     object
verified_status              object
author_ban_status            object
video_view_count            float64
video_like_count            float64
video_share_count           float64
video_download_count        float64
video_comment_count         float64
dtype: object
```

[6]:
```
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
```

Generate basic descriptive statistics about the dataset.

[7]:
```
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
```

Check for and handle missing values.

[8]:
```
print(data.isnull().sum())
# checking for any missing values
```

```
#                           0
claim_status              298
video_id                    0
video_duration_sec          0
video_transcription_text  298
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
```

[9]:
```
# Dropping rows with missing values
data.dropna(axis=0, inplace= True)
data.reset_index(inplace=True, drop=True)
```

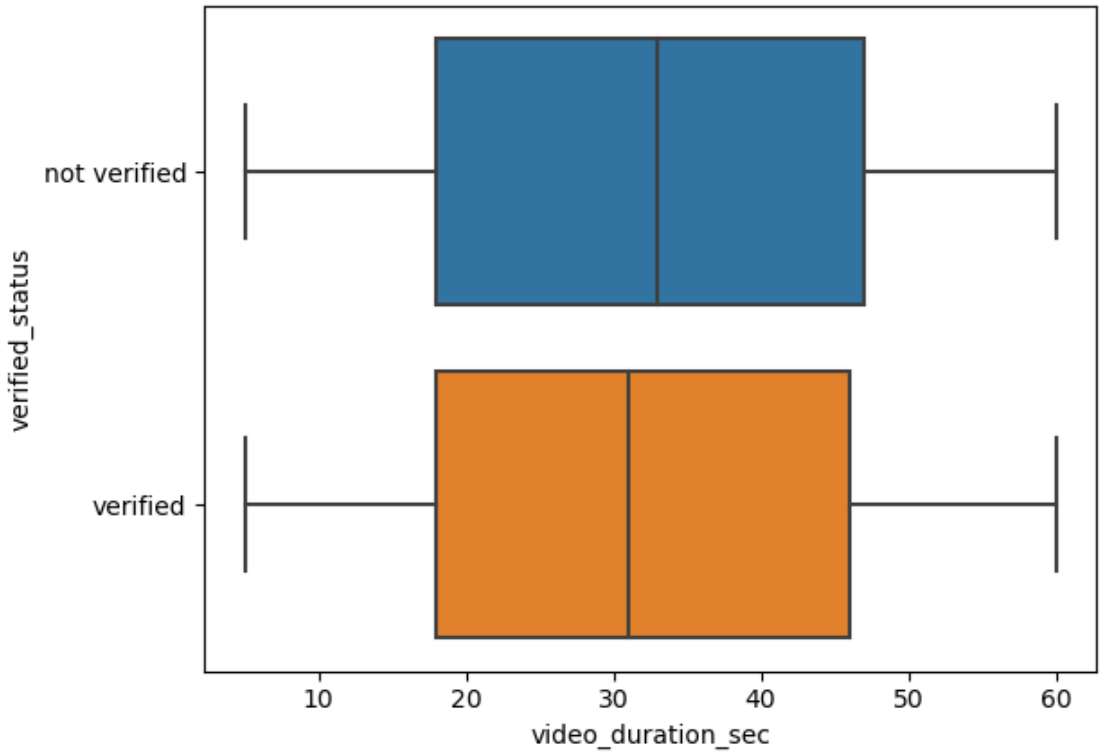
Check for and handle duplicates.

[10]:
```
# Checking for duplicates
data.duplicated()
data.duplicated().sum()
```

[10]:
```
0
```

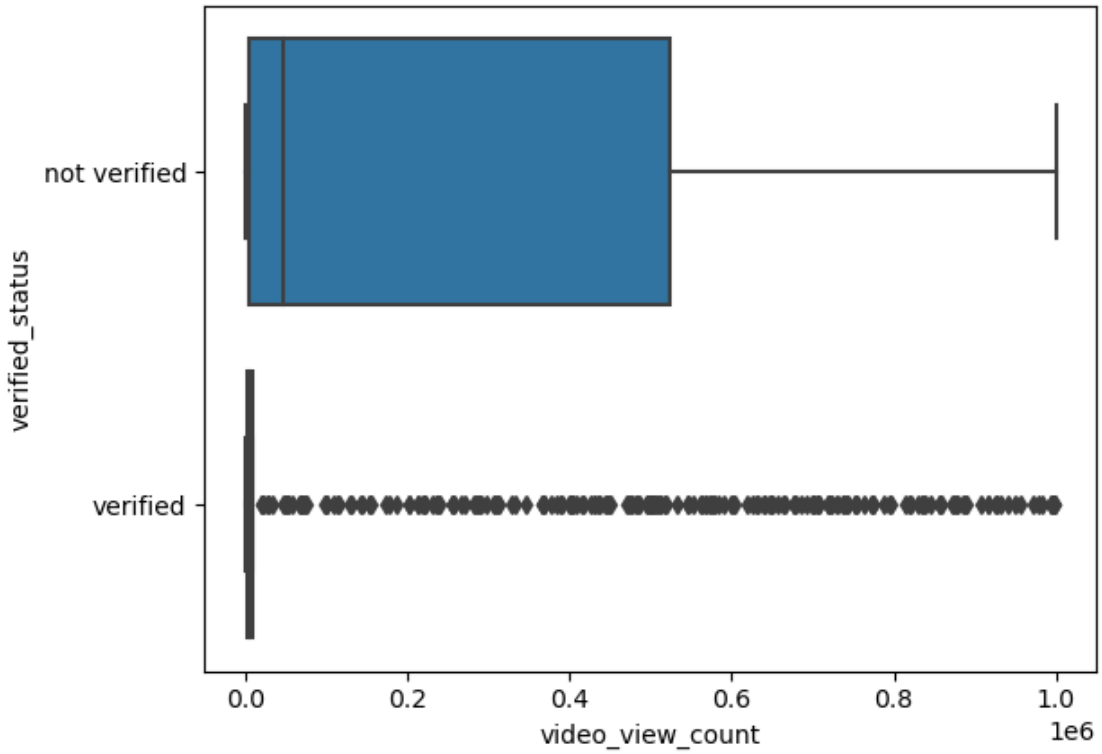
[11]:
```
# Creating a boxplot to visualize distribution of `video_duration_sec`
sns.boxplot(x = "video_duration_sec", y = "verified_status", data = data)
```

[11]:
```
<Axes: xlabel='video_duration_sec', ylabel='verified_status'>
```



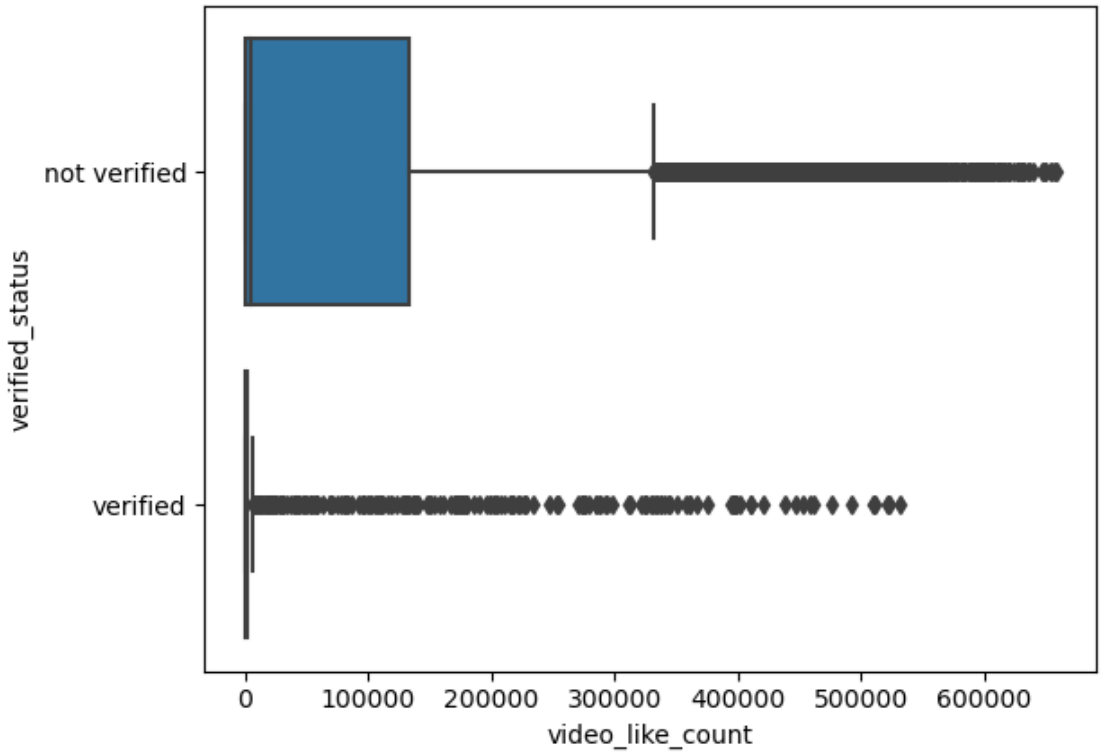
[12]:
```
# Creating a boxplot to visualize distribution of `video_view_count`
sns.boxplot(x = "video_view_count", y = "verified_status", data = data)
```

[12]: <Axes: xlabel='video_view_count', ylabel='verified_status'>



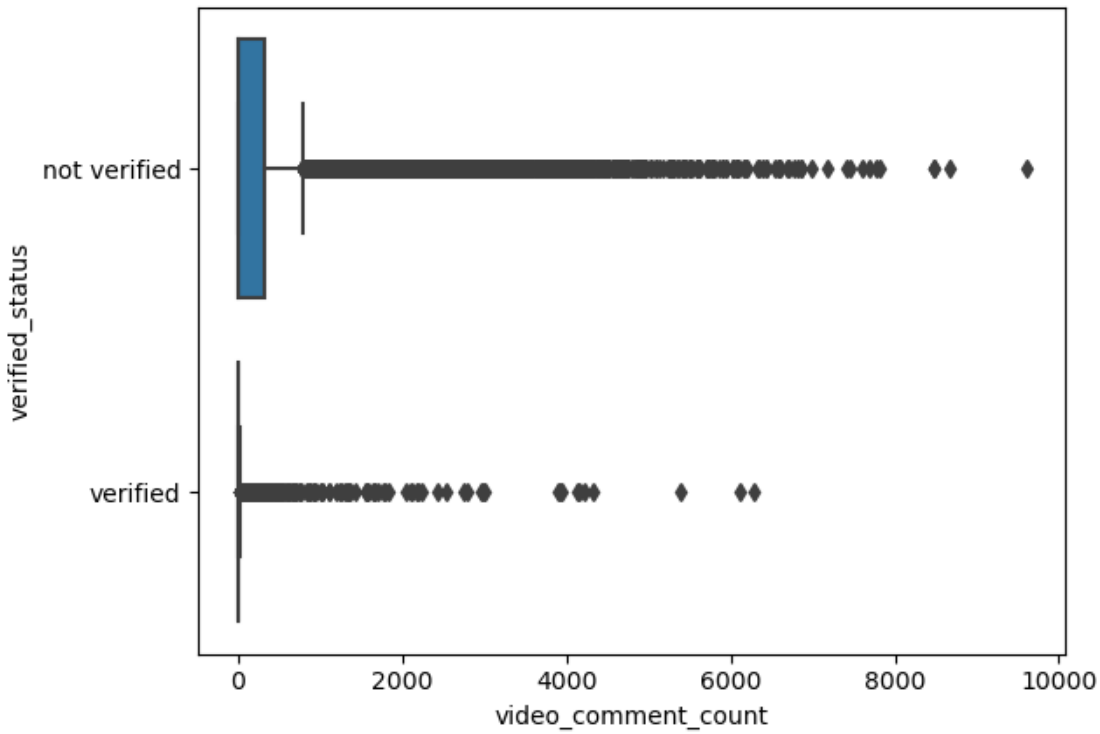
[13]:
```
# Creating a boxplot to visualize distribution of `video_like_count`
sns.boxplot(x = "video_like_count", y = "verified_status", data = data)
```

[13]: <Axes: xlabel='video_like_count', ylabel='verified_status'>



[14]:
```
# Creating a boxplot to visualize distribution of `video_comment_count`
sns.boxplot( x= "video_comment_count", y= 'verified_status', data = data)
```

[14]:
```
<Axes: xlabel='video_comment_count', ylabel='verified_status'>
```



[15]:
```
# Checking for and handling outliers for video_like_count
Q1 = data['video_like_count'].quantile(0.25)
Q3 = data['video_like_count'].quantile(0.75)
IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

outliers = data[(data['video_like_count'] < lower_bound) |␣
 ↪(data['video_like_count'] > upper_bound)]
print("Outliers based on IQR:")
print(outliers)

# Removing outliers
data[(data['video_like_count'] >= lower_bound) & (data['video_like_count'] <=␣
 ↪upper_bound)]
print("DataFrame after removing outliers based on IQR:")
print(data)
```

```
Outliers based on IQR:
         # claim_status    video_id  video_duration_sec  \
6        7        claim  4958886992                  16
12      13        claim  3609761483                  51
```

```
13      14        claim  3850678773                  20
27      28        claim  6569363811                  22
28      29        claim  6301836558                  21
…    …          …         …                  …
9586  9587        claim  3469997668                   7
9587  9588        claim  4032914023                  58
9597  9598        claim  9440207084                  33
9603  9604        claim  3883493316                  49
9604  9605        claim  4765029942                   9

                                video_transcription_text verified_status  \
6     someone shared with me that elvis presley has …    not verified
12    someone shared with me that the longest record…    not verified
13    someone shared with me that 1920 was the last …    not verified
27    someone shared with me that sneezing while tra…    not verified
28    someone shared with me that people don't sneez…    not verified
…                                                   …            …
9586  a colleague discovered on the radio a claim th…    not verified
9587  a colleague discovered on the radio a claim th…    not verified
9597  a colleague discovered on the radio a claim th…    not verified
9603  a colleague discovered on the radio a claim th…    not verified
9604  a colleague discovered on the radio a claim th…        verified

     author_ban_status  video_view_count  video_like_count  video_share_count  \
6               active          750345.0          486192.0           193911.0
12              active          700081.0          434565.0            97995.0
13        under review          929685.0          497236.0           154917.0
27        under review          812056.0          329068.0             3515.0
28              active          677855.0          332569.0            97961.0
…                …               …                 …                 …
9586            active          931007.0          455662.0           164314.0
9587            banned          706385.0          456631.0           174090.0
9597            active          885151.0          568550.0            79845.0
9603            active          737177.0          460743.0            54550.0
9604            active          546987.0          360080.0            79346.0

      video_download_count  video_comment_count
6                   8616.0               5446.0
12                  2408.0               1411.0
13                  1225.0                805.0
27                  5200.0               1108.0
28                  5531.0               2386.0
…                    …                  …
9586               10216.0               6809.0
9587                9027.0               5300.0
9597                9700.0               2875.0
9603                8119.0               3372.0
9604                4537.0               2432.0
```

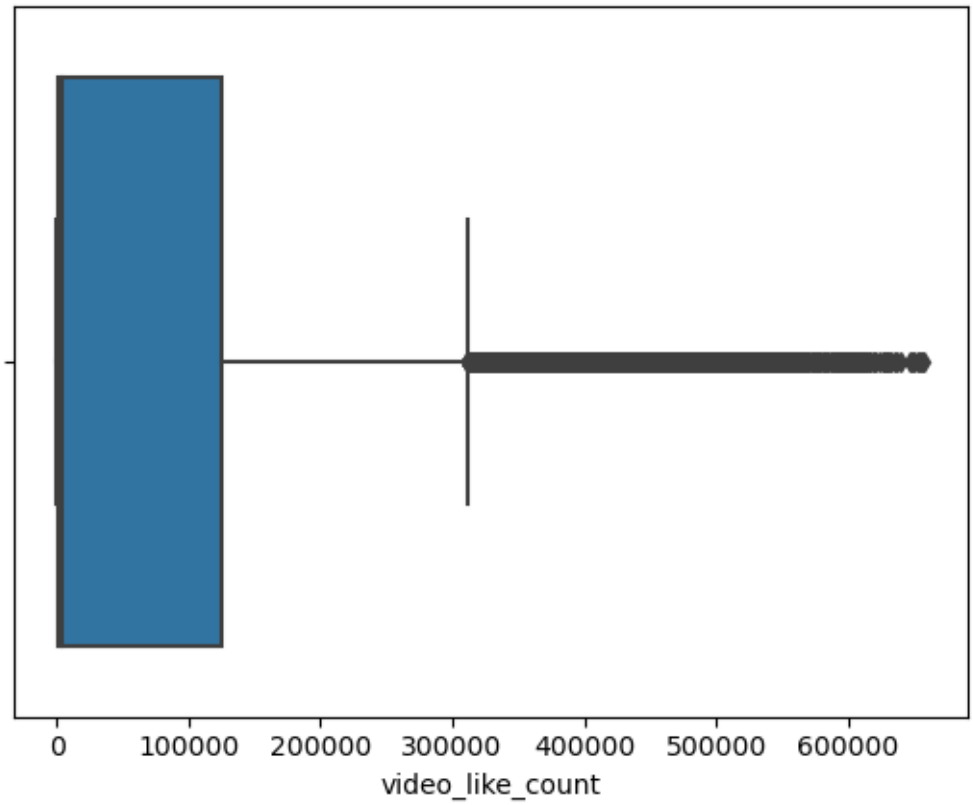
```
[1726 rows x 12 columns]
DataFrame after removing outliers based on IQR:
           # claim_status    video_id  video_duration_sec  \
0          1        claim  7017666017                  59
1          2        claim  4014381136                  32
2          3        claim  9859838091                  31
3          4        claim  1866847991                  25
4          5        claim  7105231098                  19
…      …          …         …                 …
19079  19080      opinion  1492320297                  49
19080  19081      opinion  9841347807                  23
19081  19082      opinion  8024379946                  50
19082  19083      opinion  7425795014                   8
19083  19084      opinion  4094655375                  58

                               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
…                                                   …            …
19079  in our opinion the earth holds about 11 quinti…    not verified
19080  in our opinion the queens in ant colonies live…    not verified
19081  in our opinion the moon is moving away from th…    not verified
19082  in our opinion lightning strikes somewhere on …    not verified
19083  in our opinion a pineapple plant can only prod…    not verified

      author_ban_status  video_view_count  video_like_count  \
0          under review          343296.0           19425.0
1                active          140877.0           77355.0
2                active          902185.0           97690.0
3                active          437506.0          239954.0
4                active           56167.0           34987.0
…                 …               …               …
19079            active            6067.0             423.0
19080            active            2973.0             820.0
19081            active             734.0             102.0
19082            active            3394.0             655.0
19083            active            5034.0             815.0

       video_share_count  video_download_count  video_comment_count
0                  241.0                   1.0                  0.0
1                19034.0                1161.0                684.0
2                 2858.0                 833.0                329.0
3                34812.0                1234.0                584.0
4                 4110.0                 547.0                152.0
```

```
...                 ...                 ...                 ...
19079               81.0                 8.0                 2.0
19080               70.0                 3.0                 0.0
19081                7.0                 2.0                 1.0
19082              123.0                11.0                 4.0
19083              281.0                11.0                 1.0

[19084 rows x 12 columns]
```

Checking class balance.

[16]:
```
sns.boxplot(x = "video_like_count", data = data)
```

[16]:
```
<Axes: xlabel='video_like_count'>
```



Approximately 94.2% of the dataset represents videos posted by unverified accounts and 5.8% represents videos posted by verified accounts. So the outcome variable is not very balanced.

Use resampling to create class balance in the outcome variable, if needed.

[19]:
```
# Identifing data points from majority and minority classes
data_majority = data[data["verified_status"] == "not verified"]
data_minority = data[data["verified_status"] == "verified"]
```

```
# Upsample the minority class (which is "verified")
data_minority_upsampled = resample(data_minority,
                                   replace=True,                 # to sample with␣
 ↪replacement
                                   n_samples=len(data_majority), # to match␣
 ↪majority class
                                   random_state=0)               # to create␣
 ↪reproducible results

# Combining majority class with upsampled minority class
data_upsampled = pd.concat([data_majority, data_minority_upsampled]).
 ↪reset_index(drop=True)

# Displaying new class counts
data_upsampled["verified_status"].value_counts()
```

[19]:
```
verified_status
not verified    17884
verified        17884
Name: count, dtype: int64
```

Get the average `video_transcription_text` length for videos posted by verified accounts and the average `video_transcription_text` length for videos posted by unverified accounts.

[20]:
```
# Getting the average `video_transcription_text` length for claims and the␣
 ↪average `video_transcription_text` length for opinions
data_upsampled[["verified_status", "video_transcription_text"]].
 ↪groupby(by="verified_status")[["video_transcription_text"]].agg(func=lambda␣
 ↪array: np.mean([len(text) for text in array]))
```

[20]:

verified_status	video_transcription_text
not verified	89.401141
verified	84.569559

Extracting the length of each `video_transcription_text` and adding this as a column to the dataframe, so that it can be used as a potential feature in the model.

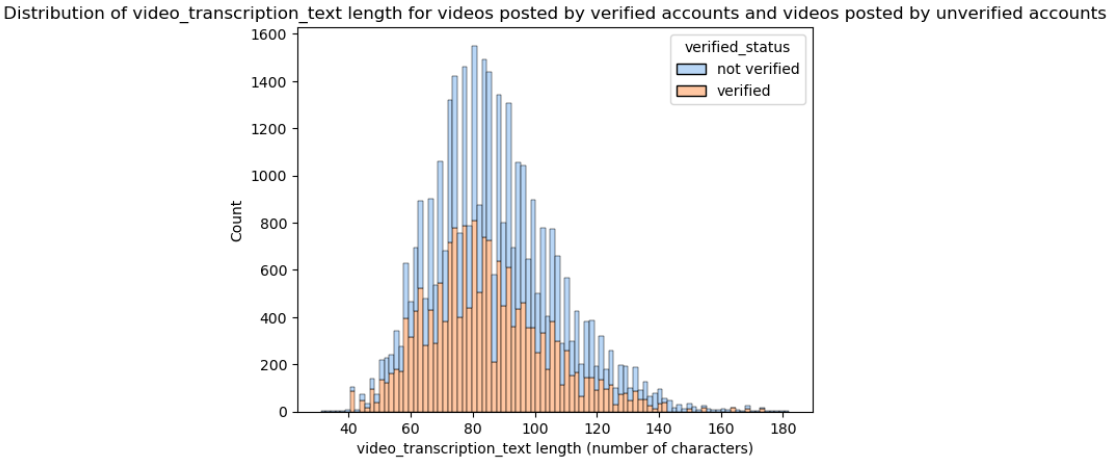
[21]:
```
# Extracting the length of each `video_transcription_text` and adding this as a␣
 ↪column to the dataframe
data_upsampled["text_length"] = data_upsampled["video_transcription_text"].
 ↪apply(func=lambda text: len(text))
```

[22]:
```
data_upsampled.head()
```

[22]:
```
   # 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  text_length
0                   1.0                  0.0           97
1                1161.0                684.0          107
2                 833.0                329.0          137
3                1234.0                584.0          131
4                 547.0                152.0          128
```

Visualizing the distribution of `video_transcription_text` length for videos posted by verified accounts and videos posted by unverified accounts.

[23]:
```
sns.histplot(data=data_upsampled, stat="count", multiple="stack",␣
 ↪x="text_length", kde=False, palette="pastel",
             hue="verified_status", element="bars", legend=True)
plt.title("Seaborn Stacked Histogram")
plt.xlabel("video_transcription_text length (number of characters)")
plt.ylabel("Count")
plt.title("Distribution of video_transcription_text length for videos posted by␣
 ↪verified accounts and videos posted by unverified accounts")
plt.show()
```



3.0.3 Task 2b. Examine correlations

Next, code a correlation matrix to help determine most correlated variables.

[24]:
```
# Coding a correlation matrix to help determine most correlated variable
data_upsampled.corr(numeric_only=True)
```

[24]:

	#	video_id	video_duration_sec	\
#	1.000000	-0.000853	-0.011729	
video_id	-0.000853	1.000000	0.011859	
video_duration_sec	-0.011729	0.011859	1.000000	
video_view_count	-0.697007	0.002554	0.013589	
video_like_count	-0.581483	0.006507	0.004890	
video_share_count	-0.504015	0.010515	0.002206	
video_download_count	-0.487096	0.008753	0.003989	
video_comment_count	-0.413799	0.013983	-0.004586	
text_length	-0.193677	-0.007083	-0.002981	

	video_view_count	video_like_count	video_share_count	\
#	-0.697007	-0.581483	-0.504015	
video_id	0.002554	0.006507	0.010515	
video_duration_sec	0.013589	0.004890	0.002206	
video_view_count	1.000000	0.832832	0.711313	
video_like_count	0.832832	1.000000	0.850053	
video_share_count	0.711313	0.850053	1.000000	
video_download_count	0.690048	0.828082	0.710117	
video_comment_count	0.583485	0.706140	0.620182	
text_length	0.244693	0.202386	0.171651	

	video_download_count	video_comment_count	text_length
#	-0.487096	-0.413799	-0.193677

video_id	0.008753	0.013983	-0.007083
video_duration_sec	0.003989	-0.004586	-0.002981
video_view_count	0.690048	0.583485	0.244693
video_like_count	0.828082	0.706140	0.202386
video_share_count	0.710117	0.620182	0.171651
video_download_count	1.000000	0.857679	0.173396
video_comment_count	0.857679	1.000000	0.149750
text_length	0.173396	0.149750	1.000000

Visualizing a correlation heatmap of the data.

[25]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(
    data_upsampled[["video_duration_sec", "claim_status", "author_ban_status",␣
 ↪"video_view_count",
                    "video_like_count", "video_share_count",␣
 ↪"video_download_count", "video_comment_count", "text_length"]]
    .corr(numeric_only=True),
    annot=True,
    cmap="crest")
plt.title("Heatmap of the dataset")
plt.show()
```

One of the model assumptions for logistic regression is no severe multicollinearity among the features. One of the model assumptions for logistic regression is no severe multicollinearity among the features. To build a logistic regression model that meets this assumption, I excluded video_like_count. And among the variables that quantify video metrics, I am keeping video_view_count, video_share_count, video_download_count, and video_comment_count as features.

3.0.4 Selecting Variables

[30]:
```
# Selecting outcome variable

y = data_upsampled["verified_status"]
```

Select the features.

[31]:
```
# Select features
```

```
X = data_upsampled[["video_duration_sec", "claim_status", "author_ban_status",␣
↪"video_view_count", "video_share_count", "video_download_count",␣
↪"video_comment_count"]]

X.head()
```

[31]:

	video_duration_sec	claim_status	author_ban_status	video_view_count	video_share_count	video_download_count	video_comment_count
0	59	claim	under review	343296.0	241.0	1.0	0.0
1	32	claim	active	140877.0	19034.0	1161.0	684.0
2	31	claim	active	902185.0	2858.0	833.0	329.0
3	25	claim	active	437506.0	34812.0	1234.0	584.0
4	19	claim	active	56167.0	4110.0	547.0	152.0

Split the data into training and testing sets.

[33]:
```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=0)
```

Confirming that the dimensions of the training and testing sets are in alignment.

[34]:
```
# Getting Shape of Data to know its alignment
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

[34]:
```
((26826, 7), (8942, 7), (26826,), (8942,))
```

3.0.5 Encoding variables**

[35]:
```
X_train.dtypes
```

[35]:
```
video_duration_sec        int64
claim_status             object
author_ban_status        object
video_view_count        float64
video_share_count       float64
video_download_count    float64
video_comment_count     float64
dtype: object
```

[36]:
```
X_train["claim_status"].unique()
```

[36]:
```
array(['opinion', 'claim'], dtype=object)
```

[37]:
```
# Getting unique values in `author_ban_status`
X_train["author_ban_status"].unique()
```

[37]:
```
array(['active', 'under review', 'banned'], dtype=object)
```

As shown above, the `claim_status` and `author_ban_status` features are each of data type `object` currently. In order to work with the implementations of models through `sklearn`, these categorical features will need to be made numeric. One way to do this is through one-hot encoding.

Encoding categorical features in the training set using an appropriate method.

[38]:
```
X_train_to_encode = X_train[["claim_status", "author_ban_status"]]
X_train_to_encode.head()
```

[38]:

	claim_status	author_ban_status
33058	opinion	active
20491	opinion	active
25583	opinion	active
18474	opinion	active
27312	opinion	active

[39]:
```
X_encoder = OneHotEncoder(drop='first', sparse_output=False)
```

[40]:
```
# Fitting and transforming the training features using the encoder
X_train_encoded = X_encoder.fit_transform(X_train_to_encode)
```

[41]:
```
# Getting feature names from encoder
X_encoder.get_feature_names_out()
```

[41]:
```
array(['claim_status_opinion', 'author_ban_status_banned',
       'author_ban_status_under review'], dtype=object)
```

[42]:
```
X_train_encoded
```

[42]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [1., 0., 0.],
       …,
       [1., 0., 0.],
       [1., 0., 0.],
       [0., 1., 0.]])
```

[43]:
```
# Place encoded training features (which is currently an array) into a dataframe
X_train_encoded_df = pd.DataFrame(data=X_train_encoded, columns=X_encoder.
 ↪get_feature_names_out())
```

```
# Display first few rows
X_train_encoded_df.head()
```

[43]:
```
   claim_status_opinion  author_ban_status_banned  \
0                   1.0                       0.0
1                   1.0                       0.0
2                   1.0                       0.0
3                   1.0                       0.0
4                   1.0                       0.0

   author_ban_status_under review
0                            0.0
1                            0.0
2                            0.0
3                            0.0
4                            0.0
```

[45]:
```
# Displaying first few rows of `X_train` with `claim_status` and␣
 ↪`author_ban_status` columns dropped (since these features are being␣
 ↪transformed to numeric)
X_train.drop(columns=["claim_status", "author_ban_status"]).head()
```

[45]:
```
       video_duration_sec  video_view_count  video_share_count  \
33058                  33            2252.0               23.0
20491                  52            6664.0              550.0
25583                  37            6327.0              257.0
18474                  57            1702.0               28.0
27312                  21            3842.0              101.0

       video_download_count  video_comment_count
33058                   4.0                  0.0
20491                  53.0                  2.0
25583                   3.0                  0.0
18474                   0.0                  0.0
27312                   1.0                  0.0
```

[47]:
```
 #Concatenating `X_train` and `X_train_encoded_df` to form the final dataframe␣
 ↪for training data (`X_train_final`)
# Note: Using `.reset_index(drop=True)` to reset the index in X_train after␣
 ↪dropping `claim_status` and `author_ban_status`,
# so that the indices align with those in `X_train_encoded_df` and `count_df`
X_train_final = pd.concat([X_train.drop(columns=["claim_status",␣
 ↪"author_ban_status"]).reset_index(drop=True), X_train_encoded_df], axis=1)

# Displaying first few rows
X_train_final.head()
```

[47]:
```
   video_duration_sec  video_view_count  video_share_count  \
0                  33            2252.0               23.0
1                  52            6664.0              550.0
2                  37            6327.0              257.0
3                  57            1702.0               28.0
4                  21            3842.0              101.0

   video_download_count  video_comment_count  claim_status_opinion  \
0                   4.0                 0.0                   1.0
1                  53.0                 2.0                   1.0
2                   3.0                 0.0                   1.0
3                   0.0                 0.0                   1.0
4                   1.0                 0.0                   1.0

   author_ban_status_banned  author_ban_status_under review
0                       0.0                            0.0
1                       0.0                            0.0
2                       0.0                            0.0
3                       0.0                            0.0
4                       0.0                            0.0
```

[48]:
```
# Checking data type of outcome variable
y_train.dtype
```

[48]:
```
dtype('O')
```

[49]:
```
# Getting unique values of outcome variable
y_train.unique()
```

[49]:
```
array(['verified', 'not verified'], dtype=object)
```

A shown above, the outcome variable is of data type `object` currently. One-hot encoding can be used to make this variable numeric.

[50]:
```
# Seting up an encoder for one-hot encoding the categorical outcome variable

y_encoder = OneHotEncoder(drop='first', sparse_output=False)
```

[51]:
```
# Encode the training outcome variable
# Notes:
#   - Adjusting the shape of `y_train` before passing into `.fit_transform()`, since it takes in 2D array
#   - Using `.ravel()` to flatten the array returned by `.fit_transform()`, so that it can be used later to train the model
y_train_final = y_encoder.fit_transform(y_train.values.reshape(-1, 1)).ravel()

# Display the encoded training outcome variable
```

```
y_train_final
```

[51]:
```
array([1., 1., 1., …, 1., 1., 0.])
```

3.0.6 Model building

Constructing a model and fit it to the training set.

[59]:
```
# Constructing a logistic regression model and fit it to the training set
log_clf = LogisticRegression(random_state=0, max_iter=800).fit(X_train_final,␣
 ↪y_train_final)
# Selecting the testing features that needs to be encoded
X_test_to_encode = X_test[["claim_status", "author_ban_status"]]

X_test_to_encode.head()
```

[59]:

	claim_status	author_ban_status
21061	opinion	active
31748	opinion	active
20197	claim	active
5727	claim	active
11607	opinion	active

3.1 PACE: Execute

3.1.1 Results and evaluation**

Evaluate the model.

Encoding categorical features in the testing set using an appropriate method.

[53]:
```
# Transforming the testing features using the encoder
X_test_encoded = X_encoder.transform(X_test_to_encode)

X_test_encoded
```

[53]:
```
array([[1., 0., 0.],
       [1., 0., 0.],
       [0., 0., 0.],
       …,
       [1., 0., 0.],
       [0., 0., 1.],
       [1., 0., 0.]])
```

[55]:
```
# Placing encoded testing features (which is currently an array) into a␣
 ↪dataframe
X_test_encoded_df = pd.DataFrame(data=X_test_encoded, columns=X_encoder.
 ↪get_feature_names_out())
```

```
X_test_encoded_df.head()


X_test.drop(columns=["claim_status", "author_ban_status"]).head()
```

[55]:

	video_duration_sec	video_view_count	video_share_count	video_download_count	video_comment_count
21061	41	2118.0	57.0	5.0	2.0
31748	27	5701.0	157.0	1.0	0.0
20197	31	449767.0	75385.0	5956.0	1789.0
5727	19	792813.0	56597.0	5146.0	3413.0
11607	54	2044.0	68.0	19.0	2.0

```
# Concatenating `X_test` and `X_test_encoded_df` to form the final dataframe␣
 ↪for training data (`X_test_final`)
# Note: Using `.reset_index(drop=True)` to reset the index in X_test after␣
 ↪dropping `claim_status`, and `author_ban_status`,
# so that the indices align with those in `X_test_encoded_df` and␣
 ↪`test_count_df`
X_test_final = pd.concat([X_test.drop(columns=["claim_status",␣
 ↪"author_ban_status"]).reset_index(drop=True), X_test_encoded_df], axis=1)

X_test_final.head()
```

[57]:

	video_duration_sec	video_view_count	video_share_count	video_download_count	video_comment_count	claim_status_opinion	author_ban_status_banned	author_ban_status_under review
0	41	2118.0	57.0	5.0	2.0	1.0	0.0	0.0
1	27	5701.0	157.0	1.0	0.0	1.0		
2	31	449767.0	75385.0	5956.0	1789.0	0.0		
3	19	792813.0	56597.0	5146.0	3413.0	0.0		
4	54	2044.0	68.0	19.0	2.0	1.0		

```
1                    0.0                    0.0
2                    0.0                    0.0
3                    0.0                    0.0
4                    0.0                    0.0
```

Test the logistic regression model. Use the model to make predictions on the encoded testing set.

[60]:
```
# Using the logistic regression model to get predictions on the encoded testing␣
 ↪set
y_pred = log_clf.predict(X_test_final)
```

Displaying the predictions on the encoded testing set.

[61]:
```
# Display the predictions on the encoded testing se

y_pred
```

[61]:
```
array([1., 1., 0., …, 1., 0., 1.])
```

Display the true labels of the testing set.

[62]:
```
y_test
```

[62]:
```
21061        verified
31748        verified
20197        verified
5727     not verified
11607    not verified
             …
14756    not verified
26564        verified
14800    not verified
35705        verified
31060        verified
Name: verified_status, Length: 8942, dtype: object
```

Encode the true labels of the testing set so it can be compared to the predictions.

[63]:
```
# Encoding the testing outcome variable
# Notes:
#   - Adjusting the shape of `y_test` before passing into `.transform()`, since␣
 ↪it takes in 2D array
#   - Using `.ravel()` to flatten the array returned by `.transform()`, so that␣
 ↪it can be used later to compare with predictions
y_test_final = y_encoder.transform(y_test.values.reshape(-1, 1)).ravel()

# Displaying the encoded testing outcome variable
y_test_final
```

[63]:
```
array([1., 1., 1., …, 0., 1., 1.])
```

Confirm again that the dimensions of the training and testing sets are in alignment since additional features were added.

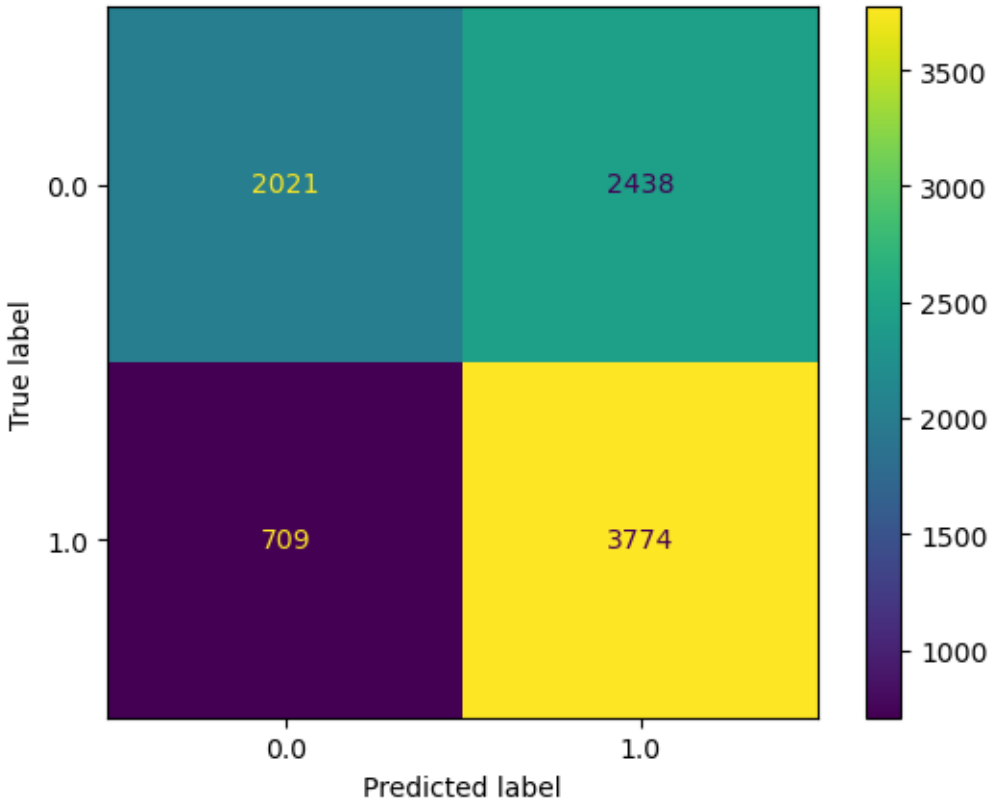
[64]:
```
# Get shape of each training and testing set
X_train_final.shape, y_train_final.shape, X_test_final.shape, y_test_final.shape
```

[64]:
```
((26826, 8), (26826,), (8942, 8), (8942,))
```

3.1.2 Visualizing model results**

Creating a confusion matrix to visualize the results of the logistic regression model.

[65]:
```
# Computing values for confusion matrix
log_cm = confusion_matrix(y_test_final, y_pred, labels=log_clf.classes_)

# Create display of confusion matrix
log_disp = ConfusionMatrixDisplay(confusion_matrix=log_cm,␣
 ↪display_labels=log_clf.classes_)

# Plot confusion matrix
log_disp.plot()

# Display plot
plt.show()
```



Creating a classification report that includes precision, recall, f1-score, and accuracy metrics to evaluate the performance of the logistic regression model.

[66]:
```
# Create a classification report
(3774+2021) / (3774 + 709 + 2021 + 2438)
```

[66]:
```
0.6480653097740997
```

3.1.3 Interpreting model coefficients**

[67]:
```
# Getting the feature names from the model and the model coefficients (which␣
↪represent log-odds ratios)
# Creating classification report for logistic regression model
target_labels = ["verified", "not verified"]
print(classification_report(y_test_final, y_pred, target_names=target_labels))
```

```
              precision    recall  f1-score   support

    verified       0.74      0.45      0.56      4459
not verified       0.61      0.84      0.71      4483
```

```
    accuracy                           0.65      8942
   macro avg       0.67      0.65      0.63      8942
weighted avg       0.67      0.65      0.63      8942
```

3.2 Conclusion**

1. What are the key takeaways from this project?
2. What results can be presented from this project?

==> Key takeaways:

- The dataset has a few strongly correlated variables, which might lead to multicollinearity issues when fitting a logistic regression model. We decided to drop `video_like_count` from the model building.
- Based on the logistic regression model, each additional second of the video is associated with 0.009 increase in the log-odds of the user having a verified status.
- The logistic regression model had not great, but acceptable predictive power: a precision of 74% is less than ideal, but a recall of 84% is very good.

I developed a logistic regression model for verified status based on video features. The model had decent predictive power. Based on the estimated model coefficients from the logistic regression, longer videos tend to be associated with higher odds of the user being verified. Other video features have small estimated coefficients in the model, so their association with verified status seems to be small.