In [1]:
```
import pandas as pd
```

In [2]:
```
# Load the dataset to understand its structure and first few entries
file_path = '../data/dataset.csv'
data = pd.read_csv(file_path)

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

Out[2]:

	verifiedby	country	class	title	published_date	country1	country2
0	La Silla Vacía	Colombia	FALSE	The coronavirus is an amplified bacteria rela...	2020/06/17	Colombia	NaN
1	Newtral.es	Spain	FALSE	A law allows people to go for a run during th...	2020/04/09	Spain	NaN
2	FactCrescendo	India	False	Chinese converting to Islam after realizing t...	2020/02/20	India	NaN
3	France 24 Observers	France	False	Bat market and bat meat are being sold in Wuhan.	2020/01/27	France	NaN
4	Agência Lupa	Brazil	False	You can self-diagnose COVID-19 by holding you...	2020/03/16	Brazil	NaN

In [3]:
```
# Assessing missing values across the dataset
missing_values = data.isnull().sum()

# Checking the unique values in the 'class' column to ensure consistency
class_values = data['class'].unique()

missing_values, class_values
```

```
Out[3]: (verifiedby           0
         country            172
         class                2
         title                0
         published_date       0
         country1           172
         country2          6343
         country3          6801
         country4          6873
         article_source       7
         ref_source           0
         source_title       454
         content_text       775
         category          6730
         lang               342
         dtype: int64,
         array(['FALSE', 'False', 'MISLEADING', 'Mostly True', 'No Evidence',
                'Misleading', 'partly false', 'Mostly false', 'Partly false',
                'PARTLY FALSE', 'Explanatory', 'No evidence', 'Mostly False',
                'misleading', 'Labeled Satire', 'Partly False', 'Misleading/False',
                'no evidence', 'True', 'MOSTLY FALSE', 'Unproven', 'false', 'Fake',
                'Half True', 'PARTLY TRUE', 'Not true', 'Scam',
                "(Org. doesn't apply rating)", 'Partially false', 'MOSTLY TRUE',
                'Partly true', 'mislEADING', 'half true', 'false and misleading',
                'mostly false', nan, 'HALF TRUE', 'Two Pinocchios', 'Partly FALSE',
                'PANTS ON FIRE', 'Correct', 'Misinformation / Conspiracy theory',
                'HALF TRUTH', 'MiSLEADING', 'Partially correct', 'Unlikely',
                'Fake news', 'True but', 'Mostly true', 'Collections',
                'Unverified', 'Partially true'], dtype=object))
```

The dataset has various missing values across different columns. Most notably:

- country, country1: There are 172 missing values in each, which might be the same entries.
- country2, country3, country4: A large number of missing values, indicating that many news articles are not associated with multiple countries.
- article_source: 7 missing values, which is crucial because it's the source link to the article.
- source_title: 454 missing values, indicating missing titles in the original language.
- content_text: 775 missing entries, which might affect the analysis if the text content is needed for model training.
- category: A significant number of missing values (6730), suggesting that many articles are not categorized.
- lang: 342 missing values indicating the language is unknown for some entries.

For the 'class' column, which is our target variable, there are inconsistencies and missing values that need to be addressed. The column contains a variety of labels that are not strictly 'True' or 'False', such as:

- Different capitalizations of 'False' (e.g., 'FALSE', 'False', 'false') and similar issues with 'True'.

- Categories like 'Misleading', 'Mostly True', 'No Evidence', etc., which do not conform to a binary classification without further interpretation.
- Some labels are clearly irrelevant or not useful for binary classification, such as 'Explanatory', 'Labeled Satire', or 'Collections'.

Let's start by cleaning the 'class' column to have a uniform binary target. For the purpose of this binary classification task, we could consider labels that signify any degree of falsehood, such as 'False', 'Misleading', 'Mostly false', etc., as 'False', and labels that signify truth, such as 'True', 'Mostly True', etc., as 'True'. Ambiguous labels like 'No Evidence' or 'Unproven' could be carefully analyzed to decide on their inclusion as 'False' or removal from the dataset.

In [4]:
```
# Impute 'No Country' where values are missing for the country columns
country_columns = ['country','country1', 'country2', 'country3', 'country4']
for column in country_columns:
    data[column].fillna('No Country', inplace=True)

# Check the result of the imputation
data[country_columns].head()
```

Out[4]:

	country	country1	country2	country3	country4
0	Colombia	Colombia	No Country	No Country	No Country
1	Spain	Spain	No Country	No Country	No Country
2	India	India	No Country	No Country	No Country
3	France	France	No Country	No Country	No Country
4	Brazil	Brazil	No Country	No Country	No Country

In [5]:
```
# For simplicity, we will map various forms of 'True' and 'False' to a binar
# We will treat 'Mostly True', 'True' and similar as True and 'False', 'Most
# Ambiguous labels like 'No Evidence', 'Unproven' will be excluded.

# First, we define the mappings for 'True' and 'False'
true_labels = ['True', 'Mostly True', 'true', 'mostly true', 'True but', 'Mo
               'Partially correct', 'HALF TRUE', 'Half True', 'PARTLY TRUE',

false_labels = ['False', 'FALSE', 'false', 'Mostly False', 'MOSTLY FALSE', '
                'Misleading', 'misleading', 'mislEADING', 'MiSLEADING', 'Mis
                'FALSE AND MISLEADING', 'Partly false', 'PARTLY FALSE', 'Par
                'Partially false', 'Partially False', 'PARTIALLY FALSE', 'Fa
                'PANTS ON FIRE', 'Misinformation / Conspiracy theory', 'Two

# Now, we'll create a function to map the class to binary
def map_class(label):
    if label in true_labels:
        return 'True'
    elif label in false_labels:
        return 'False'
    else:
```

```
        return 'Ambiguous'

# Apply the mapping function to the class column
data['binary_class'] = data['class'].apply(map_class)

# Remove ambiguous entries
data_binary = data[data['binary_class'] != 'Ambiguous']

# Count the distribution of the binary classes
binary_class_distribution = data_binary['binary_class'].value_counts()

binary_class_distribution
```

Out[5]:
```
binary_class
False    6670
True       65
Name: count, dtype: int64
```

The dataset has been mapped to binary classes with the following distribution:

- False: 6670 entries
- True: 65 entries

It's evident that the dataset is highly imbalanced, with a vast majority of entries classified as 'False'. This imbalance will need to be addressed during the model training process to prevent the model from being biased towards predicting 'False' for nearly all inputs.

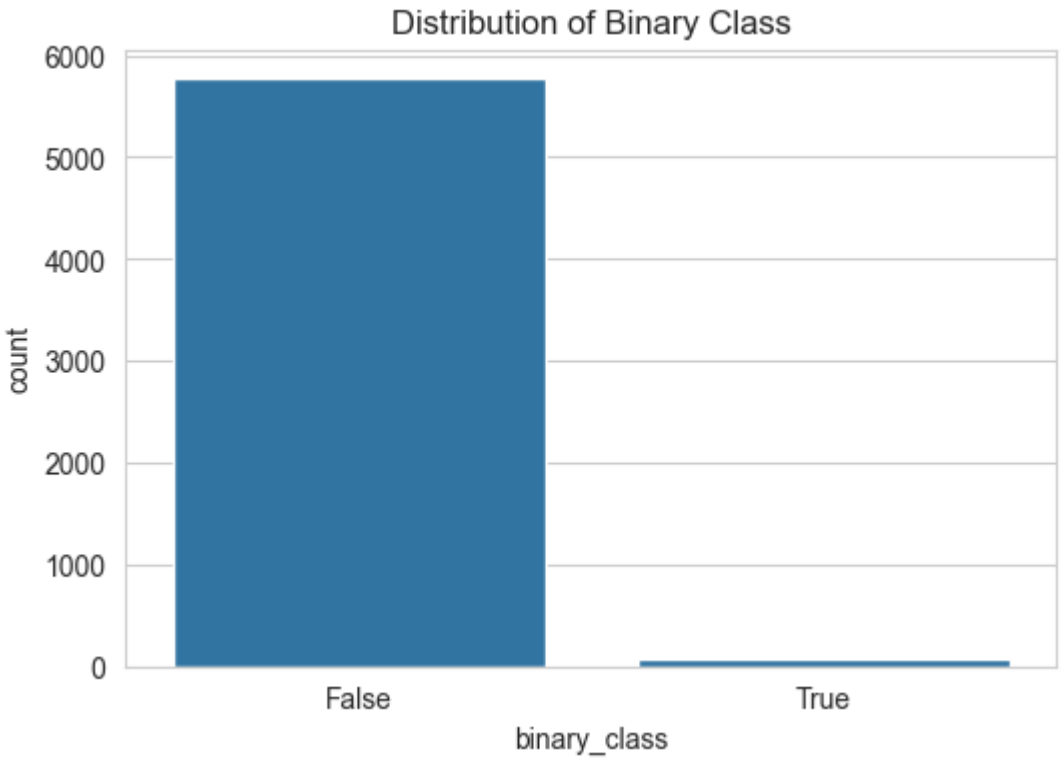
In [6]:
```
# Assessing missing values across the dataset
missing_values = data.isnull().sum()

# Checking the unique values in the 'class' column to ensure consistency
class_values = data['class'].unique()

missing_values, class_values
```

Out[6]:
```
(verifiedby          0
 country             0
 class               2
 title               0
 published_date      0
 country1            0
 country2            0
 country3            0
 country4            0
 article_source      7
 ref_source          0
 source_title      454
 content_text      775
 category         6730
 lang              342
 binary_class        0
 dtype: int64,
 array(['FALSE', 'False', 'MISLEADING', 'Mostly True', 'No Evidence',
        'Misleading', 'partly false', 'Mostly false', 'Partly false',
        'PARTLY FALSE', 'Explanatory', 'No evidence', 'Mostly False',
        'misleading', 'Labeled Satire', 'Partly False', 'Misleading/False',
        'no evidence', 'True', 'MOSTLY FALSE', 'Unproven', 'false', 'Fake',
        'Half True', 'PARTLY TRUE', 'Not true', 'Scam',
        "(Org. doesn't apply rating)", 'Partially false', 'MOSTLY TRUE',
        'Partly true', 'mislEADING', 'half true', 'false and misleading',
        'mostly false', nan, 'HALF TRUE', 'Two Pinocchios', 'Partly FALSE',
        'PANTS ON FIRE', 'Correct', 'Misinformation / Conspiracy theory',
        'HALF TRUTH', 'MiSLEADING', 'Partially correct', 'Unlikely',
        'Fake news', 'True but', 'Mostly true', 'Collections',
        'Unverified', 'Partially true'], dtype=object))
```

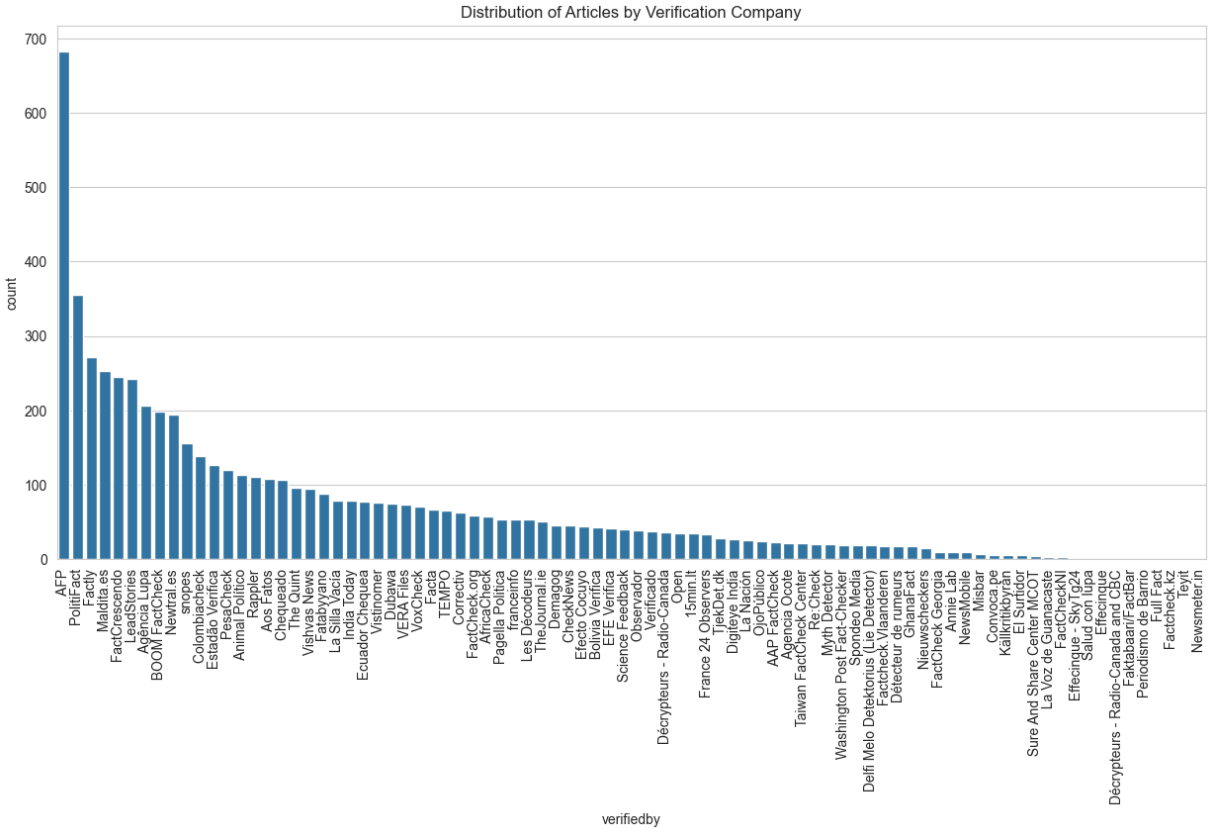
In [7]:
```
# We will proceed with handling missing values. The strategy will be:
# 1. Drop columns with a high percentage of missing values as they may not p
# 2. For columns with a lower percentage of missing values, we might impute

# Let's define a threshold for dropping columns. If a column has more than 5
threshold = 0.5 * len(data_binary)

# Dropping columns with missing values exceeding the threshold
columns_to_drop = missing_values[missing_values > threshold].index.tolist()

# Drop the identified columns
data_cleaned = data_binary.drop(columns=columns_to_drop)

# Now, let's handle the remaining columns with missing values by dropping ro
# This is a strict approach, typically we might want to impute these, but wi
data_cleaned = data_cleaned.dropna()

# Let's check the shape of the cleaned data and the remaining missing values
cleaned_shape = data_cleaned.shape
remaining_missing_values = data_cleaned.isnull().sum()

cleaned_shape, remaining_missing_values
```

Out[7]:
```
((5834, 15),
 verifiedby        0
 country           0
 class             0
 title             0
 published_date    0
 country1          0
 country2          0
 country3          0
 country4          0
 article_source    0
 ref_source        0
 source_title      0
 content_text      0
 lang              0
 binary_class      0
 dtype: int64)
```

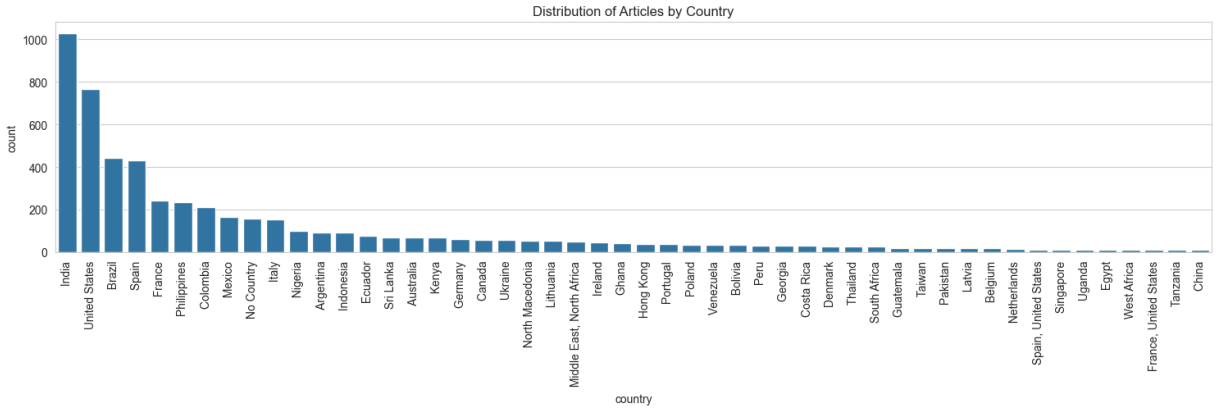
In [8]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Detailed Exploratory Data Analysis (EDA)

# Setting up the aesthetic style of the plots
sns.set_style("whitegrid")

# Summary statistics for numerical features
numerical_summary = data_cleaned.describe()

# Distribution of the binary class
plt.figure(figsize=(6, 4))
sns.countplot(data=data_cleaned, x='binary_class')
plt.title('Distribution of Binary Class')
plt.show()
```

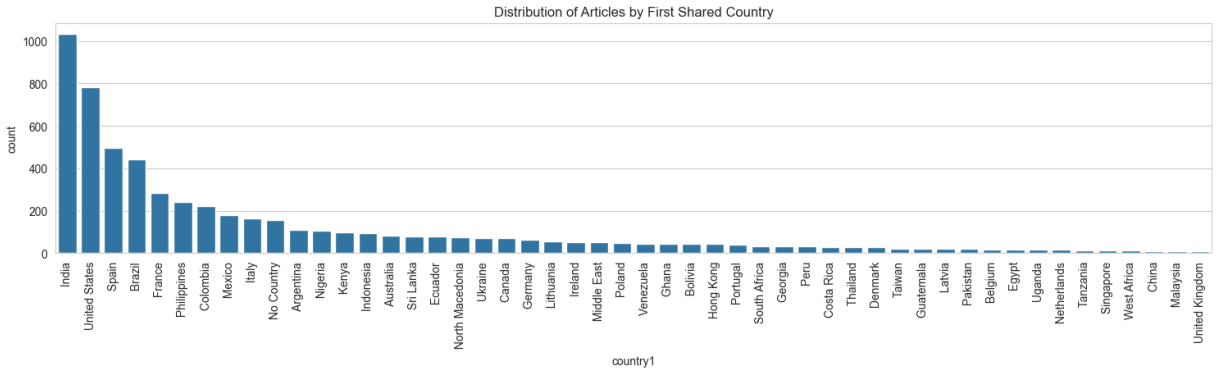


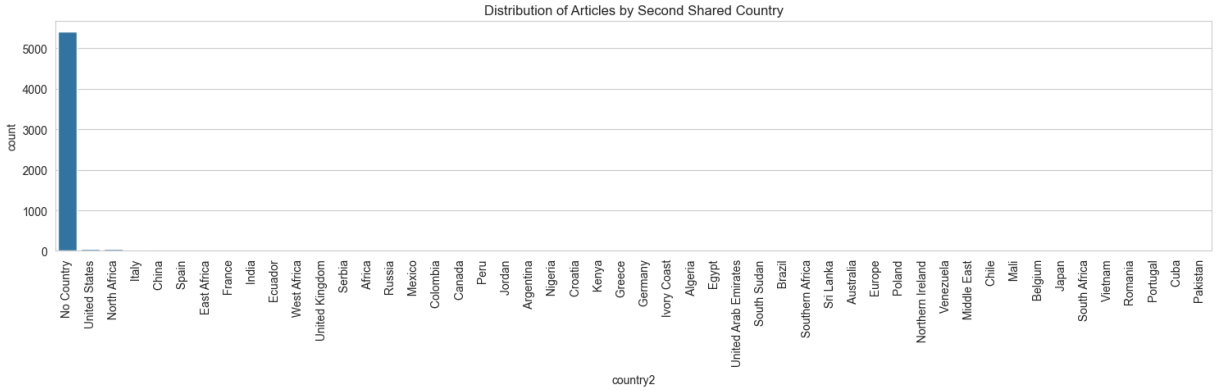
In [9]:
```
# To further understand the dataset and find insights for feature engineerin

# We will look into the following:
# 1. Distribution of articles by verification company (verifiedby)
# 2. Distribution of articles by country (country and country1)
# 3. Distribution of articles by language (lang)
# 4. Temporal distribution of articles (published_date)

# First, let's convert 'published_date' to datetime for temporal analysis
data_cleaned['published_date'] = pd.to_datetime(data_cleaned['published_date
data_cleaned['year'] = data_cleaned['published_date'].dt.year
data_cleaned['month'] = data_cleaned['published_date'].dt.month

# Distribution of articles by verification company
plt.figure(figsize=(15, 7))
sns.countplot(data=data_cleaned, x='verifiedby', order=data_cleaned['verifie
plt.xticks(rotation=90)
plt.title('Distribution of Articles by Verification Company')
plt.show()
```



In [10]:
```
# Distribution of articles by country
fig, ax = plt.subplots(3, 1, figsize=(15, 15))
sns.countplot(data=data_cleaned, x='country', order=data_cleaned['country'].
ax[0].set_title('Distribution of Articles by Country')
ax[0].tick_params(axis='x', rotation=90)
sns.countplot(data=data_cleaned, x='country1', order=data_cleaned['country1'
ax[1].set_title('Distribution of Articles by First Shared Country')
ax[1].tick_params(axis='x', rotation=90)
sns.countplot(data=data_cleaned, x='country2', order=data_cleaned['country2'
ax[2].set_title('Distribution of Articles by Second Shared Country')
ax[2].tick_params(axis='x', rotation=90)
plt.tight_layout()
plt.show()
```

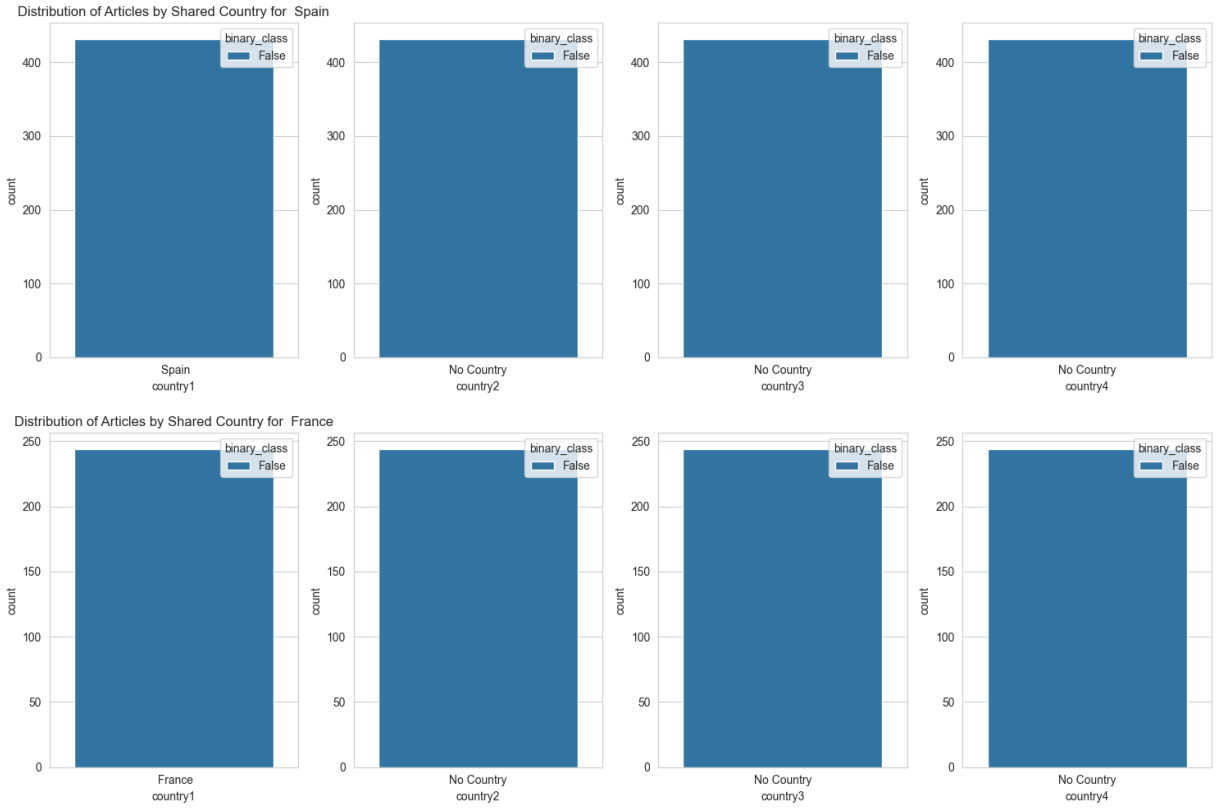




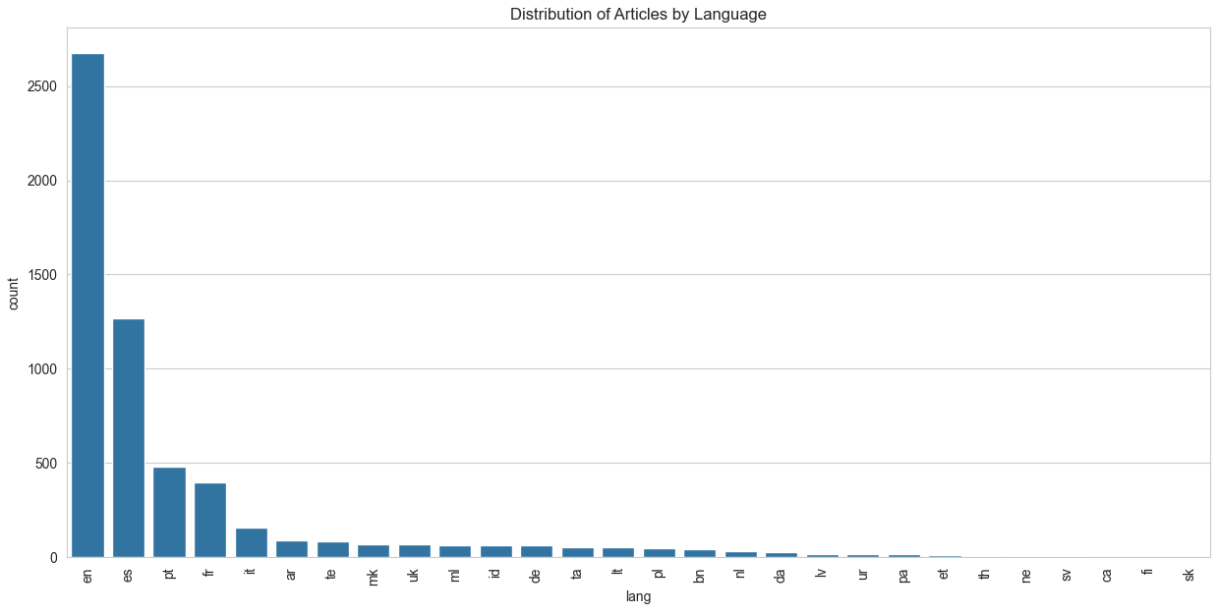


In [11]:
```
# Let's first get the top 50 countries by "country" value count.
top_5_countries = data_cleaned['country'].value_counts().head(5).index.tolis

# Then, for each of these countries, we'll identify the distribution of coun
# along with the class of fake news "binary class".

# We'll create a dictionary to hold the data
country_sharing_distribution = {}

# Loop through each of the top 50 countries
for country in top_5_countries:
    # Filter the dataset for the current country
    country_data = data_cleaned[data_cleaned['country'] == country]
    # Filtering the data to exclude entries where all four country columns h
    country_filtered_data = country_data[
        (country_data['country1'] != 'No Country') |
        (country_data['country2'] != 'No Country') |
        (country_data['country3'] != 'No Country') |
        (country_data['country4'] != 'No Country')
```

```
    ]

    # Distribution of articles by language
    fig, ax = plt.subplots(1, 4, figsize=(15, 5))
    title = "Distribution of Articles by Shared Country for {}".format(count
    ax[0].set_title(title)
    sns.countplot(ax=ax[0],data=country_filtered_data, x='country1', order=c

    #ax[1].set_title(title)
    sns.countplot(ax=ax[1],data=country_filtered_data, x='country2', order=c

    #ax[2].set_title(title)
    sns.countplot(ax=ax[2], data=country_filtered_data, x='country3', order=

    #ax[3].set_title(title)
    sns.countplot(ax=ax[3],data=country_filtered_data, x='country4', order=c
    plt.tight_layout()
    plt.show()
```

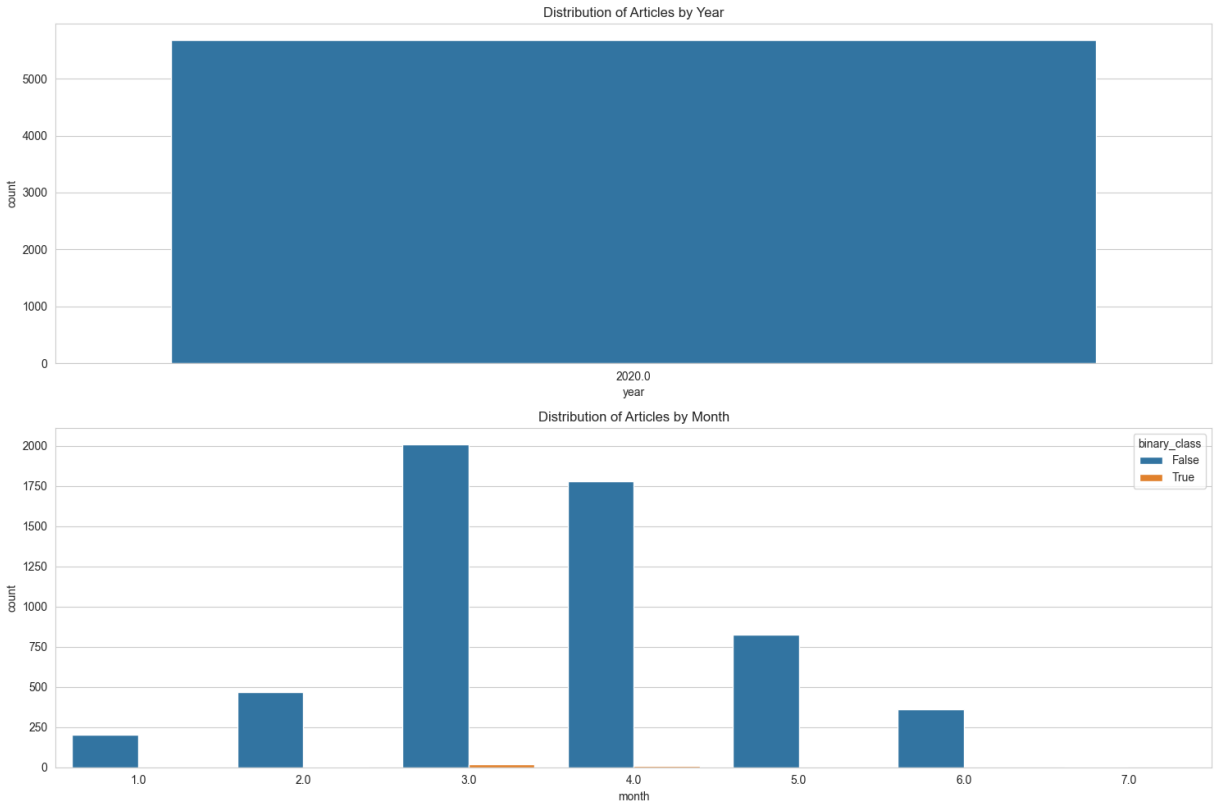


In [12]:
```
# Distribution of articles by language
plt.figure(figsize=(15, 7))
sns.countplot(data=data_cleaned, x='lang', order=data_cleaned['lang'].value_
plt.title('Distribution of Articles by Language')
plt.xticks(rotation=90)
plt.show()
```

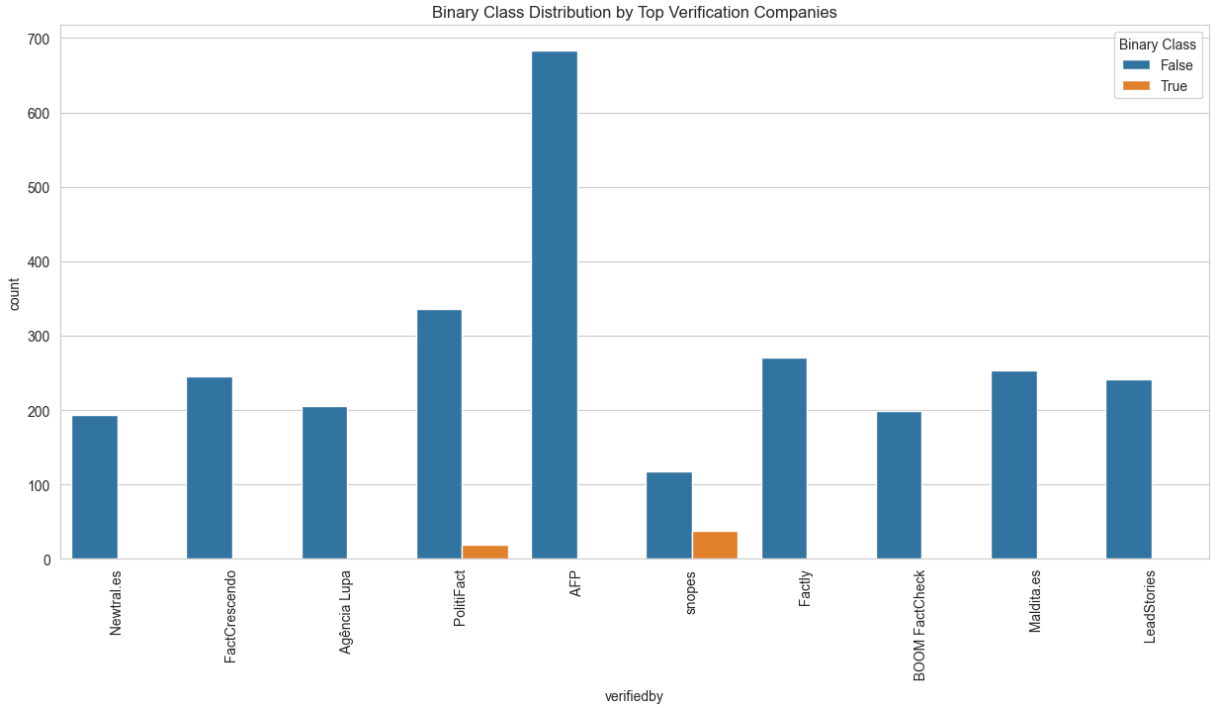


In [13]:
```
# Temporal distribution of articles by year and month
fig, ax = plt.subplots(2, 1, figsize=(15, 10))
sns.countplot(data=data_cleaned, x='year', ax=ax[0])
ax[0].set_title('Distribution of Articles by Year')
sns.countplot(data=data_cleaned, x='month', ax=ax[1],hue='binary_class')
```

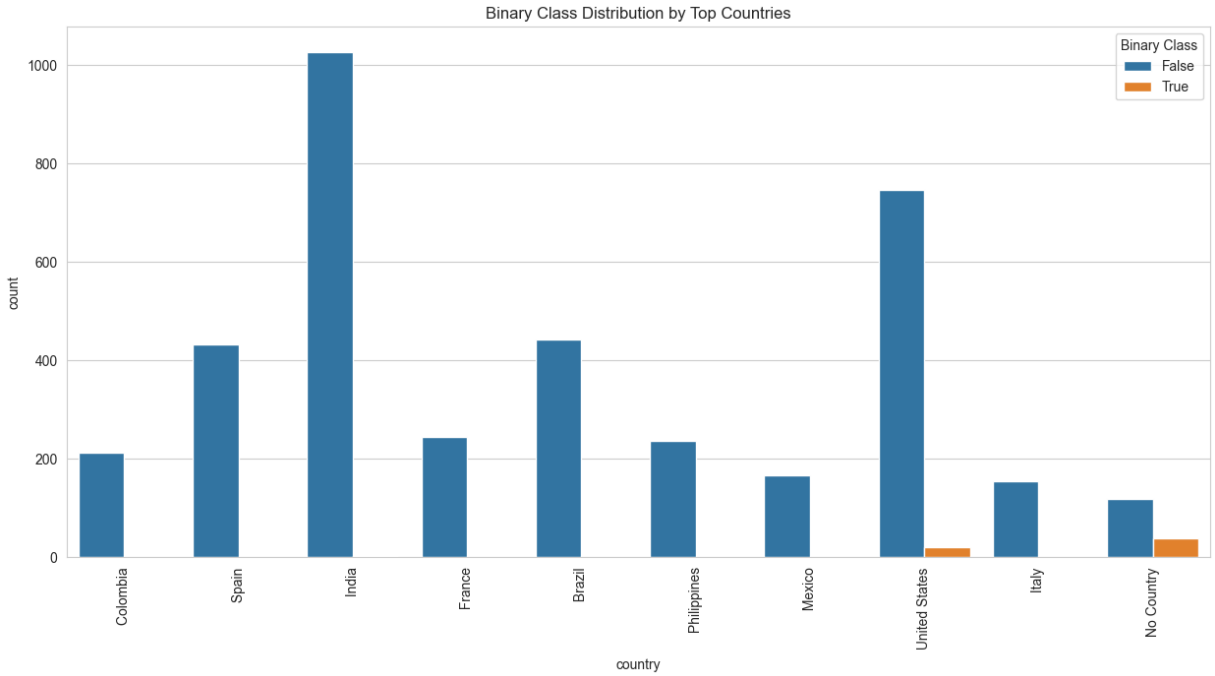
```
ax[1].set_title('Distribution of Articles by Month')
plt.tight_layout()
plt.show()
```



In [14]:
```
# Let's also look at the distribution of the binary class within some of the
# This will help us to build hypotheses about the relationship between these

# Binary class distribution by top verification companies
top_verifiedby = data_cleaned['verifiedby'].value_counts().index[:10]
plt.figure(figsize=(15, 7))
sns.countplot(data=data_cleaned[data_cleaned['verifiedby'].isin(top_verified
plt.title('Binary Class Distribution by Top Verification Companies')
plt.xticks(rotation=90)
plt.legend(title='Binary Class')
plt.show()
```

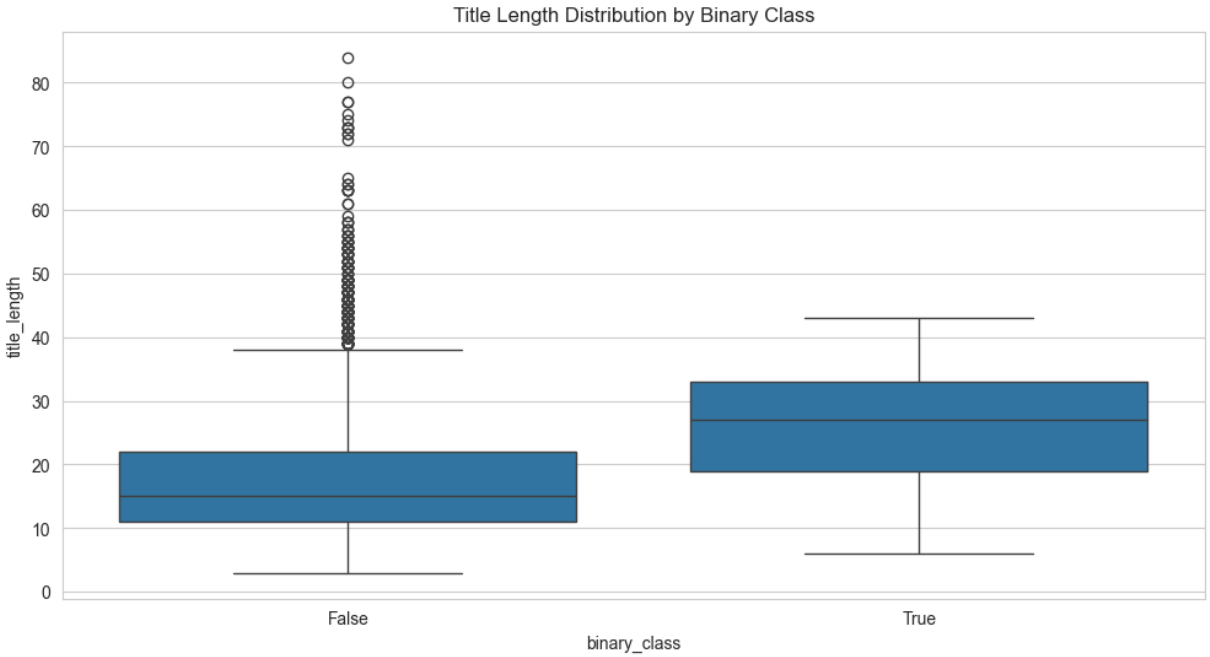


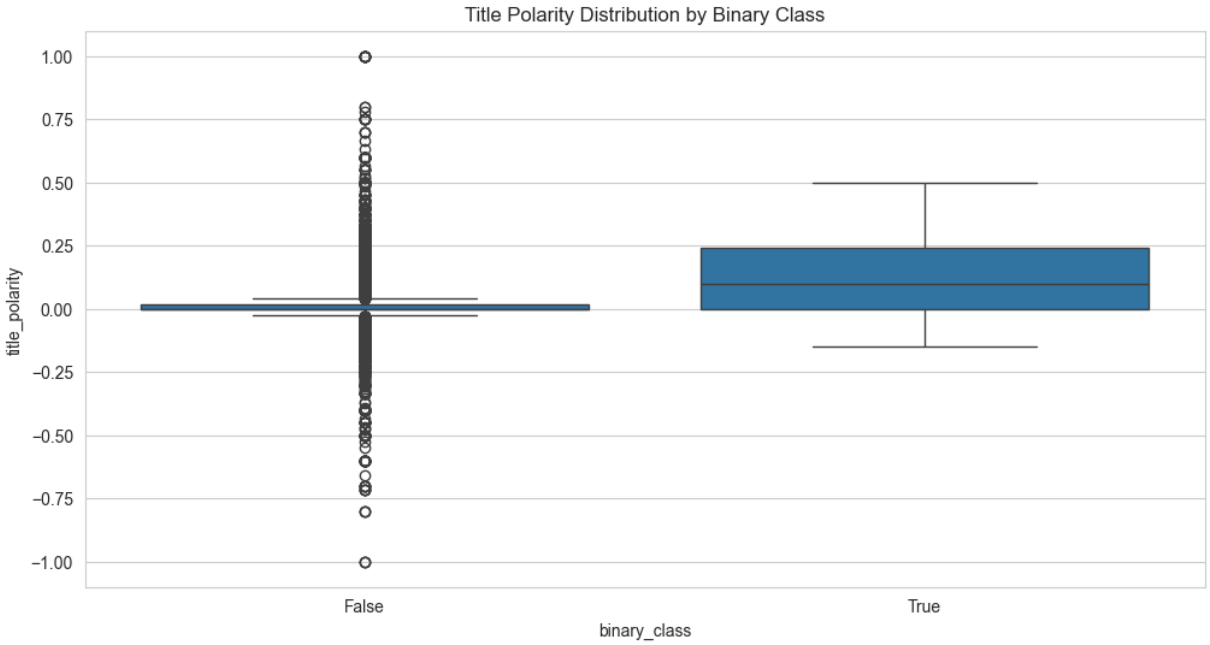
In [15]:
```
# Binary class distribution by top countries
top_countries = data_binary['country'].value_counts().index[:10]
plt.figure(figsize=(15, 7))
sns.countplot(data=data_cleaned[data_cleaned['country'].isin(top_countries)]
plt.title('Binary Class Distribution by Top Countries')
plt.xticks(rotation=90)
plt.legend(title='Binary Class')
plt.show()
```

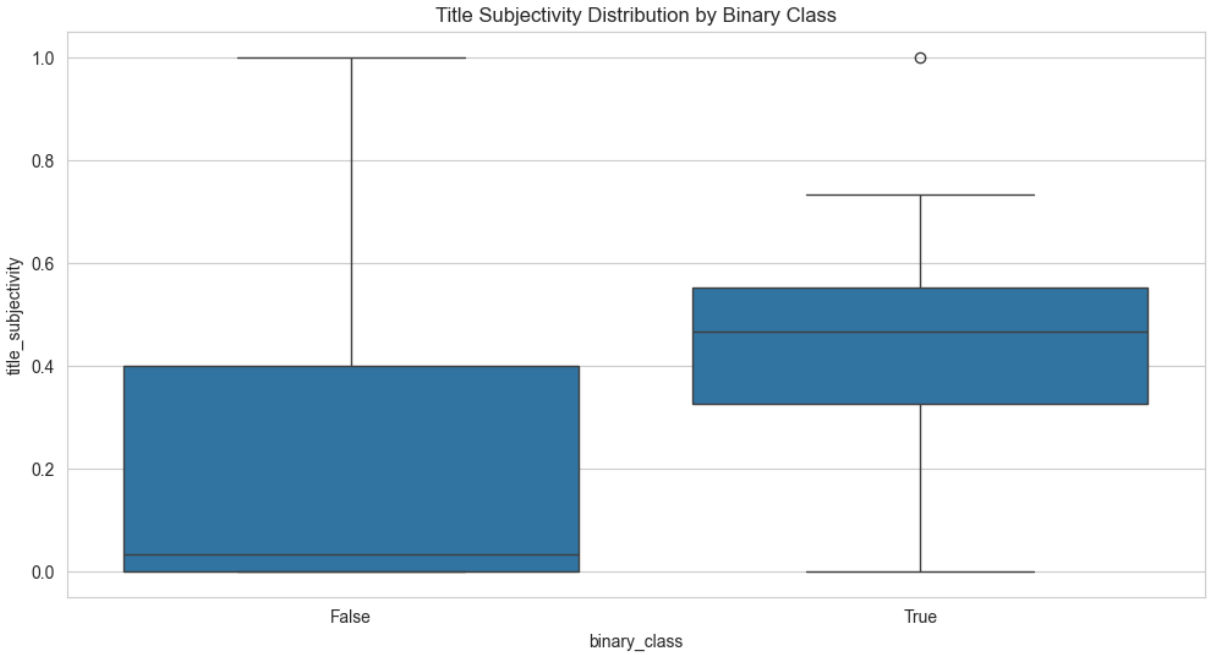


In [16]:
```
# Sentiment Analysis
from textblob import TextBlob

data_cleaned['title_length'] = data_cleaned['title'].apply(lambda x: len(x.s
```

```
# Add features for the polarity and subjectivity of the title
data_cleaned['title_polarity'] = data_cleaned['title'].apply(lambda x: TextB
data_cleaned['title_subjectivity'] = data_cleaned['title'].apply(lambda x: T

# Now, let's visualize the newly created features in relation to the binary
plt.figure(figsize=(12, 6))
sns.boxplot(x='binary_class', y='title_length', data=data_cleaned)
plt.title('Title Length Distribution by Binary Class')
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(x='binary_class', y='title_polarity', data=data_cleaned)
plt.title('Title Polarity Distribution by Binary Class')
plt.show()

plt.figure(figsize=(12, 6))
sns.boxplot(x='binary_class', y='title_subjectivity', data=data_cleaned)
plt.title('Title Subjectivity Distribution by Binary Class')
plt.show()
```



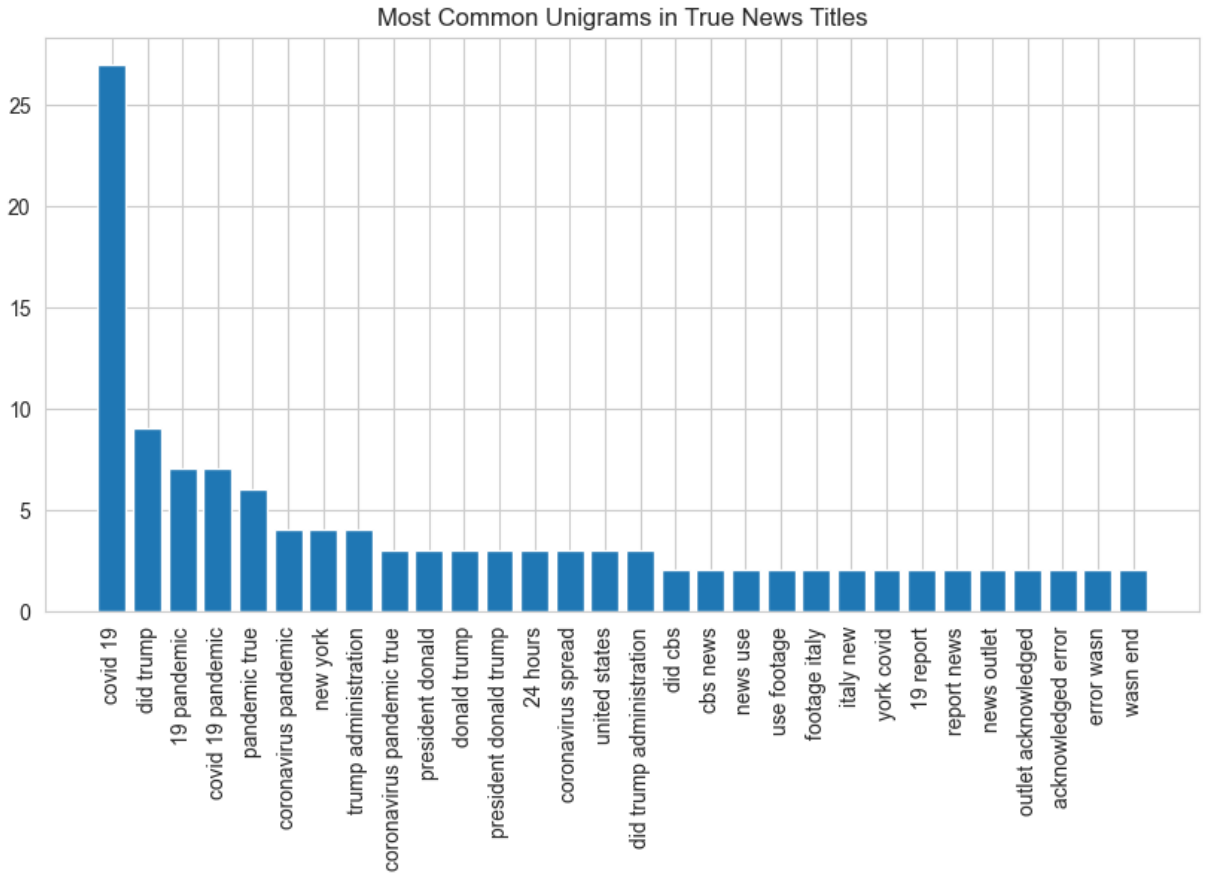


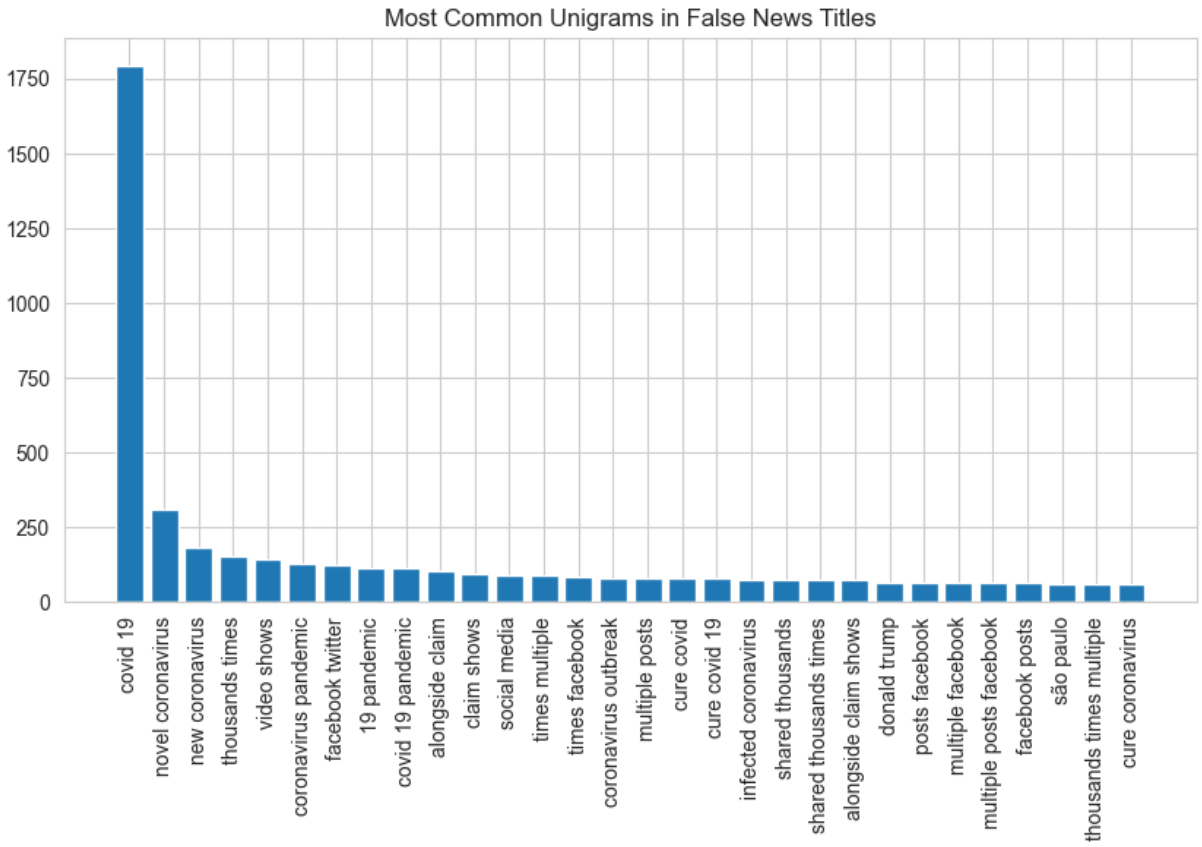


- This suggests that the length of a news title could be a useful feature in a predictive model, as there is a clear distinction in the length distribution between 'True' and 'False' classes.
- The polarity feature seems to have some predictive power, as there is a noticeable difference in distribution between the two classes, despite some overlap.
- title subjectivity seems to show a different distribution between the two classes and could therefore be considered a predictive feature.

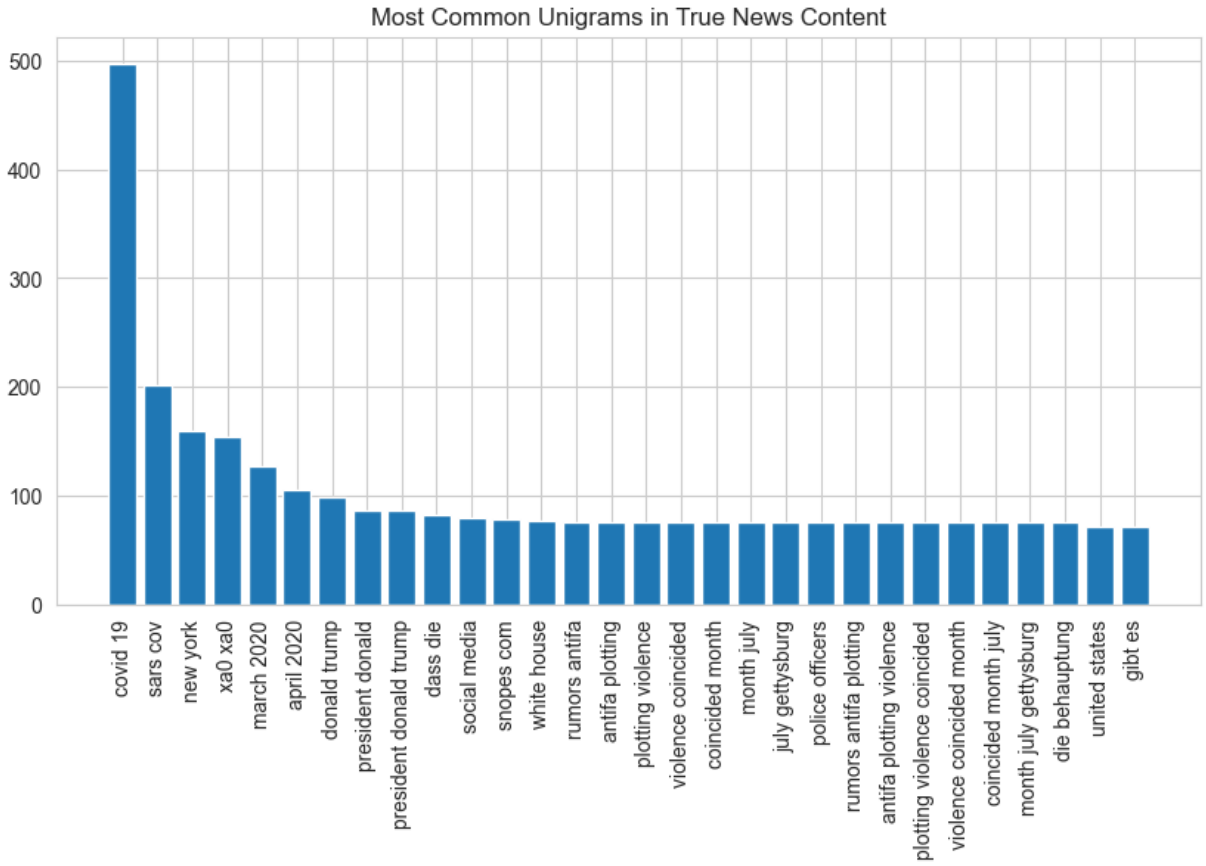
In [17]:
```
from sklearn.feature_extraction.text import CountVectorizer

# We will use CountVectorizer to analyze unigrams without stopwords

# Function to plot most common unigrams
```

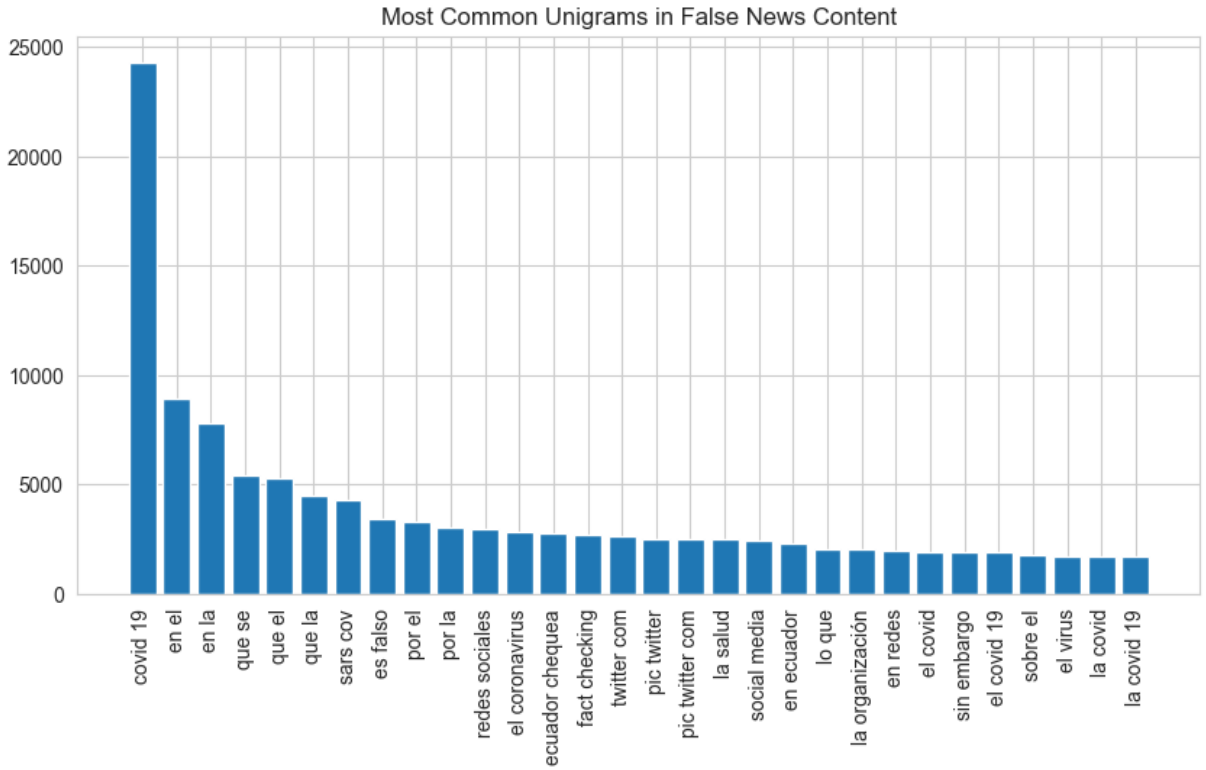
```python
def plot_most_common_unigrams(data, title, n=30):
    # Initialize the CountVectorizer
    count_vect = CountVectorizer(stop_words='english', ngram_range = (2, 3))
    # Fit and transform the data
    word_count = count_vect.fit_transform(data)
    # Sum up the counts of each vocabulary word
    sum_words = word_count.sum(axis=0)
    # Map from indices to words
    words_freq = [(word, sum_words[0, idx]) for word, idx in count_vect.voca
    # Sort the words by frequency
    sorted_words_freq = sorted(words_freq, key=lambda x: x[1], reverse=True)
    # Create a bar plot for the most common words
    words, counts = zip(*sorted_words_freq)
    plt.figure(figsize=(10, 5))
    plt.bar(words, counts)
    plt.title(title)
    plt.xticks(rotation=90)
    plt.show()

# Extract titles for 'True' and 'False' classes
titles_true = data_cleaned[data_cleaned['binary_class'] == 'True']['title']
titles_false = data_cleaned[data_cleaned['binary_class'] == 'False']['title'

# Plot the most common unigrams in titles for 'True' class
plot_most_common_unigrams(titles_true, 'Most Common Unigrams in True News Ti

# Plot the most common unigrams in titles for 'False' class
plot_most_common_unigrams(titles_false, 'Most Common Unigrams in False News
```





In [18]:
```
# Extract titles for 'True' and 'False' classes
titles_true = data_cleaned[data_cleaned['binary_class'] == 'True']['content_
titles_false = data_cleaned[data_cleaned['binary_class'] == 'False']['conten

# Plot the most common unigrams in titles for 'True' class
plot_most_common_unigrams(titles_true, 'Most Common Unigrams in True News Co

# Plot the most common unigrams in titles for 'False' class
plot_most_common_unigrams(titles_false, 'Most Common Unigrams in False News
```





In [19]:
```
def lexical_diversity(text):
    tokens = text.split()
    return len(set(tokens)) / len(tokens) if tokens else 0

data_cleaned['lexical_diversity'] = data_cleaned['content_text'].apply(lexic
```

In [20]:
```
from urllib.parse import urlparse

# Define a function to extract domain from URL
def extract_domain(url):
    parsed_uri = urlparse(url)
    # Extract the domain name (base URL)
    domain = '{uri.scheme}://{uri.netloc}/'.format(uri=parsed_uri)
    return domain

# Apply the function to your dataset
data_cleaned['source_domain'] = data_cleaned['article_source'].apply(extract
```

Identifying credible domains based on the source_domain and its association with the binary class indicating fake news involves a data-driven approach. You'll be analyzing the patterns in the source_domain relative to the binary_class column, which indicates whether news is fake (0 or 1, assuming 0 is for credible and 1 is for fake news).

Here's a conceptual outline of the steps you'd take:

- Calculate Ratios: Calculate the ratio of fake to credible news for each domain.

In [25]:
```
# Group by source_domain and calculate the number of fake and credible news
domain_summary = data_cleaned.groupby('source_domain')['binary_class'].agg([

# Calculate the ratio of fake news
domain_summary['fake_ratio'] = domain_summary['fake_count'] / (domain_summar

# You may decide on a threshold to determine credibility; this is arbitrary
threshold = 0.01
domain_summary['is_credible'] = domain_summary['fake_ratio'] < threshold
```

In [30]:
```
domain_summary
```

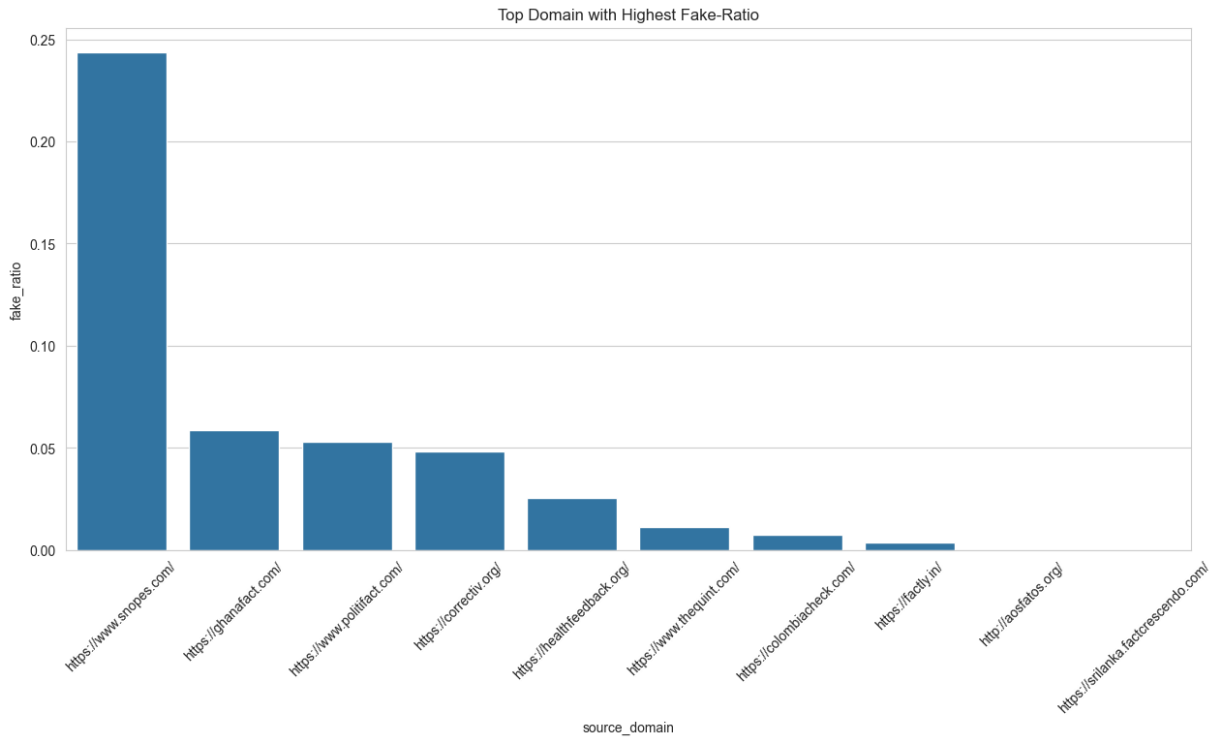
Out[30]:

source_domain	fake_count	credible_count	fake_ratio	is_credible
http://aosfatos.org/	0	1	0.000000	True
http://boliviaverifica.bo/	0	5	0.000000	True
http://factuel.afp.com/	0	3	0.000000	True
http://mythdetector.ge/	0	11	0.000000	True
http://newsmobile.in/	0	9	0.000000	True
...	...	...	...	...
https://www.snopes.com/	38	118	0.243590	False
https://www.thejournal.ie/	0	38	0.000000	True
https://www.thequint.com/	1	87	0.011364	False
https://www.vishvasnews.com/	0	88	0.000000	True
https://www.washingtonpost.com/	0	19	0.000000	True

116 rows × 4 columns

In [40]:
```
domain_summary.sort_values(by='fake_ratio',ascending=False, inplace=True)
```

In [52]:
```
domain_summary['source_domain'] = domain_summary.index
plt.figure(figsize=(15, 7))
sns.barplot(data=domain_summary[0:10], x='source_domain',y='fake_ratio')
plt.title('Top Domain with Highest Fake-Ratio')
plt.xticks(rotation=45)
plt.show()
```



In [26]:
```
# Map the domain credibility back onto the original dataset
domain_credibility = domain_summary['is_credible'].to_dict()
data_cleaned['domain_is_credible'] = data_cleaned['source_domain'].map(domai
```

In [47]:
```
domain_fake_ratio = domain_summary['fake_ratio'].to_dict()
data_cleaned['domain_fake_ratio'] = data_cleaned['source_domain'].map(domain
```

In [48]:
```
data_cleaned
```

Out[48]:

	verifiedby	country	class	title	published_date	country1	cour
0	La Silla Vacía	Colombia	FALSE	The coronavirus is an amplified bacteria rela...	2020-06-17	Colombia	Co
1	Newtral.es	Spain	FALSE	A law allows people to go for a run during th...	2020-04-09	Spain	Co
2	FactCrescendo	India	False	Chinese converting to Islam after realizing t...	2020-02-20	India	Co
3	France 24 Observers	France	False	Bat market and bat meat are being sold in Wuhan.	2020-01-27	France	Co
4	Agência Lupa	Brazil	False	You can self-diagnose COVID-19 by holding you...	2020-03-16	Brazil	Co
...	...	...	...	...	...	...	
6894	AFP	Malaysia	FALSE	A video has been viewed thousands of times on...	2020-04-09	Malaysia	Co
6895	Convoca.pe	Peru	False	An audio shared on WhatsApp indicates that th...	2020-03-19	Peru	Co
6896	Agência Lupa	Brazil	FALSE	While the world was worried with COVID-19, Ch...	2020-06-18	Brazil	Co
6899	LeadStories	United States	FALSE	Doctors encouraged by hospitals and AMA to ov...	2020-04-25	United States	Co

	verifiedby	country	class	title	published_date	country1	cour
6901	Rappler	Philippines	False	Inhaling the steam from boiled water with dis...	2020-03-27	Philippines	Co

5834 rows × 24 columns

In [50]:
```
data_cleaned.to_excel('../data/clean_data.xlsx',index=False)
```

In [ ]: