PHASE 5 CAPSTONE PROJECT

Magdalene Ondimu

Najma Abdi

Leon Maina

Brian Kariithi

Wilfred Lekisherumogi

1. INTRODUCTION

Protests are significant socio-political events that can shape the trajectory of nations and influence global dynamics. Understanding the dynamics of protests, including their causes, demands, and state responses, is critical for policymakers, researchers, and international organizations. Previous research has demonstrated the importance of protests in driving political change and social movements. For instance, Tilly (2004) highlights how protests have historically served as a mechanism for marginalized groups to voice their demands and effect change. Additionally, studies by Beissinger (2002) and Chenoweth and Stephan (2011) have shown the impact of mass mobilizations on political outcomes and the conditions under which nonviolent protests are more likely to succeed.

In the contemporary world, the need to understand protest dynamics remains as crucial as ever. The global landscape is marked by significant political, economic, and social upheavals. Protests continue to play a pivotal role in challenging injustices, advocating for rights, and prompting governmental reforms. The Arab Spring, the Black Lives Matter movement, and recent protests in response to economic policies and climate change are testament to the enduring power and relevance of collective action.

Analyzing modern protest data can provide invaluable insights into current socio-political climates, helping to forecast potential unrest, understand public sentiment, and guide policy responses. By studying this dataset, which spans protests worldwide from 1990 onwards, we aim to uncover patterns and trends in protests, identify the key issues being protested, and understand how governments typically respond to such events.

2.BUSINESS UNDERSTANDING

Understanding the dynamics of protests, including their causes, demands, and state responses, is critical for policymakers, researchers, and international organizations. This project aims to analyze protest events globally from 1990 to March 2020, focusing on identifying underlying factors, geographical distribution, and temporal trends. By leveraging NLP techniques, it will uncover common themes in protester demands and evaluate the effectiveness and impact of various state responses. The insights gained will inform actionable policy recommendations, enhance the understanding of social movements, and improve strategies for managing social unrest. Key success metrics include accurately identifying protest trends, categorizing demands, evaluating state responses, and achieving high accuracy in sentiment analysis and topic relevance.

Main Objectives

To Analyze Protest Events:

Identify the underlying factors that lead to mass protests globally. Examine the geographical distribution and temporal trends of protest events from 1990 to March 2020. Understand the patterns and characteristics of protests, including their scale and intensity.

To Understand Protester Demands:

Analyze the diversity of demands made by protesters across different regions and countries. Investigate common themes and variations in protester motivations and grievances. Apply Natural Language Processing (NLP) techniques to extract and analyze textual data related to protester demands.

To Evaluate State Responses:

Assess the effectiveness and impact of government and state responses to protests. Classify and analyze types of responses from governments, including their strategies and outcomes. Provide insights into how state responses influence the outcomes and trajectories of protest movements.

To Provide Actionable Insights:

Offer actionable policy recommendations to policymakers and government officials based on the findings. Enhance the understanding of social movements and political unrest among researchers, academics, and civil society. Foster informed decision-making and improve strategies for managing social unrest and public grievances.

Key Questions

- How has the frequency of protests changed over the years?
- What are the most common demands made by protesters?
- How do state responses vary by region and type of protest?
- What sentiments and topics are prevalent in the narratives around protests?

SPECIFIC OBJECTIVES

1. Data Collection and Preprocessing:

- Extract and preprocess textual data on protester demands from the dataset.
- Prepare data by removing noise, stop words, and tokenizing for LDA analysis.

2. Topic Modelling Using LDA:

- Implement LDA to identify underlying topics and themes within protester demands, optimizing parameters for coherence and interpretability.

3. Sentiment Analysis and Machine Learning:

- Utilize NLP techniques for sentiment analysis on protester demands, integrating LDA-derived topics for enhanced classification accuracy.
- Apply machine learning models like Logistic Regression to classify state responses based on textual data, incorporating LDA topics as features.

4. Geospatial and Temporal Analysis:

- Map protest hotspots and analyse trends over time using the dataset's location and date information.
- Conduct time series analysis to detect patterns and trends in protest occurrences.
- Leverage LDA topics to understand variations in protester demands across regions and time periods

5. Policy Recommendations:

- Derive actionable policy recommendations to improve state-citizen relations and manage social unrest effectively, informed by comprehensive insights from sentiment analysis, state response classification, and LDA-derived topics.
- Focus on proactive measures to address common demands and grievances.

3.DATA UNDERSTANDING

id: Unique identifier for each protest event.

country: The country where the protest occurred.

ccode: Country code.

year: The year the protest occurred.

region: The region where the country is located.

protest: Indicator if there was a protest (1) or not (0).

protestnumber: Sequential number of protests in the dataset.

startday: The day the protest started.

startmonth: The month the protest started.

startyear: The year the protest started.

protesterdemand1: Primary protester demands.

protesterdemand2: Secondary protester demands.

protesterdemand3: Tertiary protester demands.

protesterdemand4: Additional protester demands.

stateresponse1: Primary state response.

stateresponse2: Secondary state response.

stateresponse3: Tertiary state response.

stateresponse4: Quaternary state response.

stateresponse5: Quinary state response.

stateresponse6: Senary state response.

stateresponse7: Septenary state response.

sources: Sources of information about the protest.

notes: Additional notes about the protest.

4.METRICS OF SUCCESS

Trend Identification: The ability to accurately identify and visualize trends in protest frequency over time.

Demand Categorization: Successfully categorizing and quantifying common protester demands.

Response Evaluation: Assessing the effectiveness and variation of state responses.

Sentiment Accuracy: Achieving high accuracy in sentiment analysis of protest notes.

Topic Relevance: Effectively identifying and summarizing key topics from the protest notes

By conducting these analyses, we can gain a comprehensive understanding of global protest dynamics, which can inform policy decisions and future research directions.

Data Exploration

```
# Importing relevant libraries
#Basic libraries
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib import rcParams
%matplotlib inline
import seaborn as sns
import re


#NLTK libraries
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
import string
import wordcloud
from wordcloud import WordCloud, STOPWORDS
from nltk.stem.porter import PorterStemmer

# Machine Learning libraries
import sklearn
from sklearn import svm, datasets
from sklearn import preprocessing
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, StandardScaler,
MinMaxScaler, label_binarize
from sklearn.ensemble import ExtraTreesClassifier,
RandomForestClassifier
from sklearn.pipeline import Pipeline, make_pipeline
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.multiclass import OneVsRestClassifier

import tensorflow
import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences


#Metrics libraries
from sklearn import metrics
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, roc_auc_score,
roc_curve, auc


#Visualization libraries
from plotly import tools
import plotly.graph_objs as go
from plotly.offline import iplot

#Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\USER\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!

# Loading the Dataset
df = pd.read_csv('mass_mobilization.csv')
df.head()

          id country  ccode  year         region  protest
protestnumber  \
0  201990001  Canada     20  1990  North America        1
1
1  201990002  Canada     20  1990  North America        1
2
2  201990003  Canada     20  1990  North America        1
3
3  201990004  Canada     20  1990  North America        1
4
4  201990005  Canada     20  1990  North America        1
5

   startday  startmonth  startyear  ...  protesterdemand4
stateresponse1  \
0      15.0         1.0     1990.0  ...               NaN
ignore
1      25.0         6.0     1990.0  ...               NaN
ignore
2       1.0         7.0     1990.0  ...               NaN
ignore
3      12.0         7.0     1990.0  ...               NaN
accomodation
4      14.0         8.0     1990.0  ...               NaN  crowd
dispersal

   stateresponse2  stateresponse3 stateresponse4 stateresponse5  \
0             NaN             NaN            NaN            NaN
1             NaN             NaN            NaN            NaN
2             NaN             NaN            NaN            NaN
3             NaN             NaN            NaN            NaN
4         arrests    accomodation            NaN            NaN

  stateresponse6 stateresponse7  \
0            NaN            NaN
1            NaN            NaN
2            NaN            NaN
3            NaN            NaN
4            NaN            NaN
```

```
                                             sources  \
0  1. great canadian train journeys into history;...
1  1. autonomy s cry revived in quebec the new yo...
2  1. quebec protest after queen calls for unity ...
3  1. indians gather as siege intensifies; armed ...
4  1. dozens hurt in mohawk blockade protest the ...

                                               notes
0  canada s railway passenger system was finally ...
1  protestors were only identified as young peopl...
2  the queen, after calling on canadians to remai...
3  canada s federal government has agreed to acqu...
4  protests were directed against the state due t...

[5 rows x 31 columns]
```

```
df.shape
```

```
(17145, 31)
```

```
df.columns
```

```
Index(['id', 'country', 'ccode', 'year', 'region', 'protest',
'protestnumber',
       'startday', 'startmonth', 'startyear', 'endday', 'endmonth',
'endyear',
       'protesterviolence', 'location', 'participants_category',
       'participants', 'protesteridentity', 'protesterdemand1',
       'protesterdemand2', 'protesterdemand3', 'protesterdemand4',
       'stateresponse1', 'stateresponse2', 'stateresponse3',
'stateresponse4',
       'stateresponse5', 'stateresponse6', 'stateresponse7',
'sources',
       'notes'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17145 entries, 0 to 17144
Data columns (total 31 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   id                     17145 non-null  int64
 1   country                17145 non-null  object
 2   ccode                  17145 non-null  int64
 3   year                   17145 non-null  int64
 4   region                 17145 non-null  object
 5   protest                17145 non-null  int64
 6   protestnumber          17145 non-null  int64
 7   startday               15239 non-null  float64
 8   startmonth             15239 non-null  float64
 9   startyear              15239 non-null  float64
 10  endday                 15239 non-null  float64
 11  endmonth               15239 non-null  float64
 12  endyear                15239 non-null  float64
 13  protesterviolence      15758 non-null  float64
 14  location               15218 non-null  object
 15  participants_category  9887 non-null   object
 16  participants           15746 non-null  object
 17  protesteridentity      14684 non-null  object
 18  protesterdemand1       15238 non-null  object
 19  protesterdemand2       2977 non-null   object
 20  protesterdemand3       383 non-null    object
 21  protesterdemand4       831 non-null    object
 22  stateresponse1         15208 non-null  object
 23  stateresponse2         2888 non-null   object
 24  stateresponse3         930 non-null    object
 25  stateresponse4         244 non-null    object
 26  stateresponse5         849 non-null    object
 27  stateresponse6         16 non-null     object
 28  stateresponse7         920 non-null    object
 29  sources                15235 non-null  object
 30  notes                  15193 non-null  object
dtypes: float64(7), int64(5), object(19)
memory usage: 4.1+ MB
```

```
df.describe()
```

```
                 id         ccode          year       protest
protestnumber  \
count  1.714500e+04  17145.000000  17145.000000  17145.000000
17145.000000
mean   4.380888e+09    437.888189   2006.171654      0.888831
7.406299
std    2.320550e+09    232.054953      8.987378      0.314351
11.854041
min    2.019900e+08     20.000000   1990.000000      0.000000
0.000000
25%    2.202010e+09    220.000000   1998.000000      1.000000
1.000000
50%    4.342008e+09    434.000000   2007.000000      1.000000
3.000000
75%    6.512005e+09    651.000000   2014.000000      1.000000
8.000000
max    9.102020e+09    910.000000   2020.000000      1.000000
143.000000

           startday    startmonth     startyear        endday
endmonth  \
count   15239.000000  15239.000000  15239.000000  15239.000000
15239.000000
mean       15.455935      6.227836   2006.326465     15.580616
6.243520
std         8.817037      3.461912      8.958007      8.803944
3.461745
min         1.000000      1.000000   1990.000000      1.000000
1.000000
25%         8.000000      3.000000   1999.000000      8.000000
3.000000
50%        15.000000      6.000000   2007.000000     16.000000
6.000000
75%        23.000000      9.000000   2014.000000     23.000000
9.000000
max        31.000000     12.000000   2020.000000     31.000000
12.000000

            endyear  protesterviolence
count  15239.000000       15758.000000
mean    2006.329221           0.256060
std        8.959254           0.436469
min     1990.000000           0.000000
25%     1999.000000           0.000000
50%     2007.000000           0.000000
75%     2014.000000           1.000000
max     2020.000000           1.000000
```

- The dataset has 17,145 rows with 31 columns.
- It is comprised of both categorical and numerical data, with many columns having a large number of missing values.
- The numerical columns is mainly binary and date data.

Data Preprocessing

```
# Aggregation of protester demands and state responses.
df['demands'] = df[['protesterdemand1', 'protesterdemand2',
'protesterdemand3', 'protesterdemand4']].apply(lambda x: ',
'.join(x.dropna().astype(str)), axis=1)
df['response'] = df[['stateresponse1', 'stateresponse2',
'stateresponse3', 'stateresponse4', 'stateresponse5',
'stateresponse6', 'stateresponse7']].apply(lambda x: ',
'.join(x.dropna().astype(str)), axis=1)

df.head()

          id country  ccode  year         region  protest
protestnumber  \
0  201990001  Canada     20  1990  North America        1
1
```

```
1  201990002  Canada     20  1990  North America        1
2
2  201990003  Canada     20  1990  North America        1
3
3  201990004  Canada     20  1990  North America        1
4
4  201990005  Canada     20  1990  North America        1
5

   startday  startmonth  startyear  ...  stateresponse2
stateresponse3  \
0      15.0         1.0     1990.0  ...             NaN
NaN
1      25.0         6.0     1990.0  ...             NaN
NaN
2       1.0         7.0     1990.0  ...             NaN
NaN
3      12.0         7.0     1990.0  ...             NaN
NaN
4      14.0         8.0     1990.0  ...         arrests
accomodation

   stateresponse4  stateresponse5 stateresponse6 stateresponse7  \
0             NaN             NaN            NaN            NaN
1             NaN             NaN            NaN            NaN
2             NaN             NaN            NaN            NaN
3             NaN             NaN            NaN            NaN
4             NaN             NaN            NaN            NaN

                                             sources  \
0  1. great canadian train journeys into history;...
1  1. autonomy s cry revived in quebec the new yo...
2  1. quebec protest after queen calls for unity ...
3  1. indians gather as siege intensifies; armed ...
4  1. dozens hurt in mohawk blockade protest the ...

                                               notes  \
0  canada s railway passenger system was finally ...
1  protestors were only identified as young peopl...
2  the queen, after calling on canadians to remai...
3  canada s federal government has agreed to acqu...
4  protests were directed against the state due t...

                                          demands  \
0  political behavior, process, labor wage dispute
1                        political behavior, process
2                        political behavior, process
3                                    land farm issue
4                        political behavior, process
```

```
                                    response
0                                     ignore
1                                     ignore
2                                     ignore
3                               accomodation
4  crowd dispersal, arrests, accomodation

[5 rows x 33 columns]

# Function to create a date string from day, month, and year
def create_date_string(row, col_prefix):
    try:
        return f"{int(row[f'{col_prefix}year']):04d}-
{int(row[f'{col_prefix}month']):02d}-
{int(row[f'{col_prefix}day']):02d}"
    except ValueError:
        return None

# Apply the function to create date strings
df['start_date'] = df.apply(lambda row: create_date_string(row,
'start'), axis=1)
df['end_date'] = df.apply(lambda row: create_date_string(row, 'end'),
axis=1)

# Convert the date strings to datetime
df['start_date'] = pd.to_datetime(df['start_date'], errors='coerce')
df['end_date'] = pd.to_datetime(df['end_date'], errors='coerce')

# Calculate protest_duration in days
df['protest_duration'] = (df['end_date'] - df['start_date']).dt.days
# Display the new columns
df[['start_date', 'end_date', 'protest_duration']].head()

  start_date   end_date  protest_duration
0 1990-01-15 1990-01-15               0.0
1 1990-06-25 1990-06-25               0.0
2 1990-07-01 1990-07-01               0.0
3 1990-07-12 1990-09-06              56.0
4 1990-08-14 1990-08-15               1.0

# Converting textual representations of participants to numeric.
# The following code maintains over 96% of the data in 'participants'.

def parse_texts(x):
    """
    Parses specific textual representations of participant counts into
numeric values.
    Handles predefined text patterns like 'dozens', 'hundreds', etc.
    """
    x = x.lower()
```

```python
    text_mapping = {
        "dozens": 50,
        "hundreds": 500,
        "thousands": 5000,
        "tens of thousands": 50000,
        "hundreds of thousands": 250000,
        "millions": 2000000,
        "million": 1000000,
        "a group": 10,
        "busloads": 50,
        "widespread": 500,
        "scores": 50,
        "a few dozen": 36,
        "a few hundred": 300,
        "a few thousand": 3000,
        "several 1000s": 5000,
        "few thousand": 3000,
        "few dozen": 24,
    }

    for key, value in text_mapping.items():
        if key in x:
            return value

    if "about " in x:
        match = re.search(r'\d+', x)
        if match:
            return int(match.group())
    if "more than " in x:
        match = re.search(r'\d+', x)
        if match:
            return int(match.group())

    if "several" in x:
        if "dozen" in x:
            return 50
        elif "hundred" in x:
            return 500
        elif "thousand" in x:
            return 5000

    return x

def strip_chars(x):
    """
    Removes unwanted characters from the string and converts to
integer if possible.
    Specifically handles values ending in 's' by multiplying the
```

```
preceding number by 5.
    """
    banned_chars = "+><,"
    x = "".join([c for c in x if c not in banned_chars])

    if x.endswith('s') and x[:-1].isdigit():
        return int(x[:-1]) * 5

    try:
        return int(x)
    except ValueError:
        return x

def avg_hyphen(x):
    """
    Calculates the average for values specified as a range (e.g.,
'100-200').
    """
    accepted_chars = "1234567890-"
    x = "".join([c for c in x if c in accepted_chars])

    if "-" in x:
        lower, upper = x.split("-")
        if lower.isdigit() and upper.isdigit():
            return (int(lower) + int(upper)) // 2

    return np.nan

def map_participants(x):
    """
    Sequentially applies parsing, stripping, and averaging to convert
text representations
    of participant counts into numeric values.
    """
    while isinstance(x, str):
        x = parse_texts(x)
        if isinstance(x, str):
            x = strip_chars(x)
        if isinstance(x, str):
            x = avg_hyphen(x)
        if isinstance(x, str):
            x = np.nan
    return x

# Converting 'partcipants' to usable values.
df['participants_numeric'] = df["participants"].map(map_participants)
df[['participants', 'participants_numeric']].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17145 entries, 0 to 17144
```

```
Data columns (total 2 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   participants          15746 non-null  object 
 1   participants_numeric  15137 non-null  float64
dtypes: float64(1), object(1)
memory usage: 268.0+ KB

df.columns

Index(['id', 'country', 'ccode', 'year', 'region', 'protest',
'protestnumber',
       'startday', 'startmonth', 'startyear', 'endday', 'endmonth',
'endyear',
       'protesterviolence', 'location', 'participants_category',
       'participants', 'protesteridentity', 'protesterdemand1',
       'protesterdemand2', 'protesterdemand3', 'protesterdemand4',
       'stateresponse1', 'stateresponse2', 'stateresponse3',
'stateresponse4',
       'stateresponse5', 'stateresponse6', 'stateresponse7',
'sources',
       'notes', 'demands', 'response', 'start_date', 'end_date',
       'protest_duration', 'participants_numeric'],
      dtype='object')
```

- Aggregated protester demands and state responses into 'demands' and 'response' columns respectively.
- Created 'start_date', 'end_date' and 'protest_duration' columns using the date data.
- Converted the participants textual representations into numerical format, stored them in 'participants_numeric'.

Data Cleaning

```
# Columns to drop
time_drops = [
    'startday', 'startmonth', 'startyear', 'endday', 'endmonth',
'endyear'
]  #Redundant as we have 'start_date'&'end_date'
other_drops = [
    'id',  #Not useful to prediction.
    'ccode',  #Not useful to prediction as we already have country.
    'protest',  #Binary column with '0' values resulting in empty rows

    'protestnumber',  #No. of protests per year.(incrementing per
protest per year)
    'location',  #Not extremely useable given how it's already being
broken by region.
    'participants_category',  #Too many null values to be of great
value. The data is also captured in 'participants_numeric'
```

```
    'participants',  #'participants_numeric' has the numeric values of
this column.
]
demand_drops = [
    'protesterdemand1', 'protesterdemand2', 'protesterdemand3',
    'protesterdemand4'
]  #Full of null values as individual columns. Aggregated in demands
column.
response_drops = [
    'stateresponse1', 'stateresponse2', 'stateresponse3',
'stateresponse4',
    'stateresponse5', 'stateresponse6', 'stateresponse7'
]  #Full of null values as individual columns. Aggregated in response
column.

columns_to_drop = time_drops + other_drops + demand_drops +
response_drops
df1 = df.drop(columns=columns_to_drop)

# Checking if columns were dropped.
df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17145 entries, 0 to 17144
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   country               17145 non-null  object
 1   year                  17145 non-null  int64
 2   region                17145 non-null  object
 3   protesterviolence     15758 non-null  float64
 4   protesteridentity     14684 non-null  object
 5   sources               15235 non-null  object
 6   notes                 15193 non-null  object
 7   demands               17145 non-null  object
 8   response              17145 non-null  object
 9   start_date            15239 non-null  datetime64[ns]
 10  end_date              15239 non-null  datetime64[ns]
 11  protest_duration      15239 non-null  float64
 12  participants_numeric  15137 non-null  float64
dtypes: datetime64[ns](2), float64(3), int64(1), object(7)
memory usage: 1.7+ MB

# Handling null values.
df1.fillna(value={"protesteridentity":"unspecified"}, inplace=True)
col_with_null = [
    'protesterviolence', #Will drop nulls as this is a crucial column
in analysis
    'sources', #Crucial column in text analysis
    'notes', #Crucial column in text analysis
    'start_date', #Crucial column in terms of temporal analysis
    'end_date', #Crucial column in terms of temporal analysis
    'protest_duration', #Crucial column in terms of temporal analysis
    'participants_numeric',#Important in understanding protester
counts
]

#Dropping null values
for col in col_with_null:
    df1.dropna(subset=[col], inplace=True)

df1.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 15087 entries, 0 to 17141
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   country               15087 non-null  object
 1   year                  15087 non-null  int64
 2   region                15087 non-null  object
 3   protesterviolence     15087 non-null  float64
 4   protesteridentity     15087 non-null  object
 5   sources               15087 non-null  object
 6   notes                 15087 non-null  object
 7   demands               15087 non-null  object
 8   response              15087 non-null  object
 9   start_date            15087 non-null  datetime64[ns]
 10  end_date              15087 non-null  datetime64[ns]
 11  protest_duration      15087 non-null  float64
 12  participants_numeric  15087 non-null  float64
dtypes: datetime64[ns](2), float64(3), int64(1), object(7)
memory usage: 1.6+ MB
```

- Dropped columns based on varying criteria explained in the code.
- Handled missing values by imputing and dropping where imputing wasn't possible.

Miscellaneous Cleaning

```
# Ensure columns have consistent data types
expected_types = {
    'country': 'object',
    'year': 'int64',
    'region': 'object',
    'protesterviolence': 'int64',
    'protesteridentity': 'object',
    'demands': 'object',
    'response': 'object',
    'start_date': 'datetime64[ns]',
    'end_date': 'datetime64[ns]',
    'protest_duration': 'int64',
    'participants_numeric': 'int64',
    'sources': 'object',
    'notes': 'object'
}

# Ensure columns have consistent data types
for column, dtype in expected_types.items():
    if dtype == 'datetime64[ns]':
        df1[column] = pd.to_datetime(df1[column], errors='coerce')
    else:
        df1[column] = df1[column].astype(dtype, errors='ignore')

# Check for duplicates and remove them if any
df1 = df1.drop_duplicates()

df1.shape

(15076, 13)

# Rearrange columns
columns_order = [
    'region', 'country', 'year', 'start_date', 'end_date',
'protest_duration',
    'participants_numeric', 'protesterviolence', 'protesteridentity',
    'demands', 'response', 'sources', 'notes'
]
df1 = df1[columns_order]
```

- Ensured all our columns had consistent data types.
- Dropped 11 duplicated rows.
- At the end of the cleaning process we've maintained 88% of the original data resulting in a shape of (15076, 13).

Feature Engineering

```
# Split the 'demands' and 'response' columns into multiple columns and
apply one-hot encoding
demands_split = df1['demands'].str.get_dummies(sep=', ')
response_split = df1['response'].str.get_dummies(sep=', ')

# Add a prefix to avoid column name clashes
demands_split = demands_split.add_prefix('demand_')
response_split = response_split.add_prefix('response_')

# Concatenate the original DataFrame with the new one-hot encoded
columns
df1 = pd.concat([df1, demands_split, response_split], axis=1)
```

```
# Drop the original 'demands' and 'response' columns
df1 = df1.drop(columns=['demands', 'response'])

df1.head()

          region country  year start_date   end_date  protest_duration  \
0  North America  Canada  1990 1990-01-15 1990-01-15                 0

1  North America  Canada  1990 1990-06-25 1990-06-25                 0

2  North America  Canada  1990 1990-07-01 1990-07-01                 0

3  North America  Canada  1990 1990-07-12 1990-09-06                56

4  North America  Canada  1990 1990-08-14 1990-08-15                 1


   participants_numeric  protesterviolence           protesteridentity  \
0                  5000                  0                 unspecified

1                  1000                  0                 unspecified

2                   500                  0  separatist parti quebecois

3                   500                  1              mohawk indians

4                   950                  1             local residents


                                             sources  ...  \
0  1. great canadian train journeys into history;...  ...
1  1. autonomy s cry revived in quebec the new yo...  ...
2  1. quebec protest after queen calls for unity ...  ...
3  1. indians gather as siege intensifies; armed ...  ...
4  1. dozens hurt in mohawk blockade protest the ...  ...

  demand_social restrictions  demand_tax policy  response_.  \
0                          0                  0           0
1                          0                  0           0
2                          0                  0           0
3                          0                  0           0
4                          0                  0           0

   response_accomodation  response_arrests  response_beatings  \
0                      0                 0                  0
1                      0                 0                  0
2                      0                 0                  0
3                      1                 0                  0
4                      1                 1                  0
```

```
   response_crowd dispersal  response_ignore  response_killings  \
0                         0                1                  0
1                         0                1                  0
2                         0                1                  0
3                         0                0                  0
4                         1                0                  0

   response_shootings
0                   0
1                   0
2                   0
3                   0
4                   0

[5 rows x 29 columns]

df1.columns

Index(['region', 'country', 'year', 'start_date', 'end_date',
       'protest_duration', 'participants_numeric',
'protesterviolence',
       'protesteridentity', 'sources', 'notes', 'demand_.',
       'demand_labor wage dispute', 'demand_land farm issue',
       'demand_police brutality', 'demand_political behavior',
       'demand_price increases', 'demand_process',
       'demand_removal of politician', 'demand_social restrictions',
       'demand_tax policy', 'response_.', 'response_accomodation',
       'response_arrests', 'response_beatings', 'response_crowd
dispersal',
       'response_ignore', 'response_killings', 'response_shootings'],
      dtype='object')

# Drop placeholder columns
df1 = df1.drop(columns=['demand_.', 'response_.']) # The '.' were
actual inputs in the data.(Not useful)

# Column order
col_order = [
    'region', 'country', 'year', 'start_date', 'end_date',
'protest_duration',
    'participants_numeric', 'protesterviolence', 'protesteridentity',
    'demand_labor wage dispute', 'demand_land farm issue',
    'demand_police brutality', 'demand_political behavior',
    'demand_price increases', 'demand_process', 'demand_removal of
politician',
    'demand_social restrictions', 'demand_tax policy',
'response_accomodation',
    'response_arrests', 'response_beatings', 'response_crowd
dispersal',
    'response_ignore', 'response_killings', 'response_shootings',
'sources',
    'notes'
]
df_cleaned = df1[col_order]

# Reset the index of the DataFrame
df_cleaned = df_cleaned.reset_index(drop=True)

# Save the cleaned and dummified data to a new CSV in the specified
directory
save_path = "C:\\Users\\USER\\Desktop\\capstone_project\\
mass_mobilization_cleaned.csv"
df_cleaned.to_csv(save_path, index=False)

print(f"File saved to {save_path}")
```

```
File saved to C:\Users\USER\Desktop\capstone_project\
mass_mobilization_cleaned.csv
```

- Split the 'demands' and 'response' columns to multiple individual columns and applied one-hot encoding on the new columns.
- Reset the index of the cleaned dataframe.
- Saved the csv to specified directory on my local machine.

EXPLORATIVE DATA ANALYSIS

```python
#Importing relevant libraries
#Basic libraries
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
from matplotlib.ticker import ScalarFormatter
from matplotlib import rcParams
%matplotlib inline
import seaborn as sns
import re
from pandas.plotting import scatter_matrix

# Machine Learning libraries
import sklearn
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler



import warnings
warnings.filterwarnings('ignore')

# Loading the dataset
data = pd.read_csv('mass_mobilization_cleaned.csv')
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15076 entries, 0 to 15075
Data columns (total 27 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   region                        15076 non-null  object
 1   country                       15076 non-null  object
 2   year                          15076 non-null  int64
 3   start_date                    15076 non-null  object
 4   end_date                      15076 non-null  object
 5   protest_duration              15076 non-null  int64
 6   participants_numeric          15076 non-null  int64
 7   protesterviolence             15076 non-null  int64
 8   protesteridentity             15076 non-null  object
 9   demand_labor wage dispute     15076 non-null  int64
 10  demand_land farm issue        15076 non-null  int64
 11  demand_police brutality       15076 non-null  int64
 12  demand_political behavior     15076 non-null  int64
 13  demand_price increases        15076 non-null   int64
 14  demand_process                15076 non-null   int64
 15  demand_removal of politician  15076 non-null   int64
 16  demand_social restrictions    15076 non-null   int64
 17  demand_tax policy             15076 non-null   int64
 18  response_accomodation         15076 non-null   int64
 19  response_arrests              15076 non-null   int64
 20  response_beatings             15076 non-null   int64
 21  response_crowd dispersal      15076 non-null   int64
 22  response_ignore               15076 non-null   int64
 23  response_killings             15076 non-null   int64
 24  response_shootings            15076 non-null   int64
 25  sources                       15076 non-null   object
 26  notes                         15076 non-null   object
dtypes: int64(20), object(7)
memory usage: 3.1+ MB

# Convert the 'start_date' and 'end_date' columns
data['start_date'] = pd.to_datetime(data['start_date'],
errors='coerce')
data['end_date'] = pd.to_datetime(data['end_date'], errors='coerce')
```

UNIVARIATE ANALYSIS

Univariate analysis is a statistical technique that examines the distribution and characteristics of a single variable. Its primary purpose is to describe and summarize data to identify patterns and gain insights.

```
# Get summary statistics for numeric columns
summary_stats = data.describe()
summary_stats

               year                     start_date  \
count  15076.000000                          15076
mean    2006.301008  2006-10-10 20:43:37.139824896
min     1990.000000            1990-01-01 00:00:00
25%     1999.000000            1999-01-31 18:00:00
50%     2007.000000            2007-10-13 12:00:00
75%     2014.000000            2014-11-01 00:00:00
max     2020.000000            2020-03-31 00:00:00
std        8.951656                            NaN

                            end_date  protest_duration
participants_numeric  \
count                          15076      15076.000000
1.507600e+04
mean   2006-10-12 10:10:03.661448704          1.560029
1.981867e+04
min              1990-01-01 00:00:00          0.000000
```

```
1.000000e+00
25%              1999-02-02 18:00:00          0.000000
1.000000e+02
50%              2007-10-18 00:00:00          0.000000
5.000000e+02
75%              2014-11-02 00:00:00          0.000000
5.000000e+03
max              2020-03-31 00:00:00        938.000000
7.000000e+06
std                              NaN         15.061353
1.569393e+05

       protesterviolence  demand_labor wage dispute  demand_land farm
issue  \
count       15076.000000               15076.000000
15076.000000
mean            0.264659                   0.145264
0.038405
min             0.000000                   0.000000
0.000000
25%             0.000000                   0.000000
0.000000
50%             0.000000                   0.000000
0.000000
75%             1.000000                   0.000000
0.000000
max             1.000000                   1.000000
1.000000
std             0.441166                   0.352378
0.192179

       demand_police brutality  demand_political behavior  ...  \
count             15076.000000               15076.000000  ...
mean                  0.072300                   0.705161  ...
min                   0.000000                   0.000000  ...
25%                   0.000000                   0.000000  ...
50%                   0.000000                   1.000000  ...
75%                   0.000000                   1.000000  ...
max                   1.000000                   1.000000  ...
std                   0.258993                   0.455986  ...

       demand_removal of politician  demand_social restrictions  \
count                  15076.000000                15076.000000
mean                       0.123773                    0.044972
min                        0.000000                    0.000000
25%                        0.000000                    0.000000
50%                        0.000000                    0.000000
75%                        0.000000                    0.000000
max                        1.000000                    1.000000
std                        0.329333                    0.207250
```

```
       demand_tax policy  response_accomodation  response_arrests  \
count       15076.000000           15076.000000      15076.000000
mean            0.093062               0.099761          0.141019
min             0.000000               0.000000          0.000000
25%             0.000000               0.000000          0.000000
50%             0.000000               0.000000          0.000000
75%             0.000000               0.000000          0.000000
max             1.000000               1.000000          1.000000
std             0.290529               0.299691          0.348053

       response_beatings  response_crowd dispersal  response_ignore  \
count       15076.000000              15076.000000     15076.000000
mean            0.052666                  0.312815         0.543977
min             0.000000                  0.000000         0.000000
25%             0.000000                  0.000000         0.000000
50%             0.000000                  0.000000         1.000000
75%             0.000000                  1.000000         1.000000
max             1.000000                  1.000000         1.000000
std             0.223374                  0.463655         0.498079

       response_killings  response_shootings
count       15076.000000        15076.000000
mean            0.054059            0.061223
min             0.000000            0.000000
25%             0.000000            0.000000
50%             0.000000            0.000000
75%             0.000000            0.000000
max             1.000000            1.000000
std             0.226142            0.239747

[8 rows x 22 columns]
```

The summary statistics provided offers a detailed look at the central tendency, dispersion, and range of each variable in the dataset. Here's an interpretation of each column:

Year Count: 15,076 protests recorded.

Mean: The average year of protests is 2006.3.

Std: The standard deviation is 8.95 years, indicating variability around the mean year.

Min: The earliest recorded protest is in 1990.

25% (Q1): 25% of protests occurred before 1999.

50% (Median): The median year of protests is 2007.

75% (Q3): 75% of protests occurred before 2014.

Max: The latest recorded protest is in 2020.

Protest Duration Count: 15,076 protests recorded. Mean: The average protest duration is 1.56 days. Std: The standard deviation is 15.06 days, indicating high variability. Min: Some protests lasted 0 days (likely indicating same-day protests). 25% (Q1): 25% of protests lasted 0 days. 50% (Median): The median protest duration is 0 days. 75% (Q3): 75% of protests lasted 0 days. Max: The longest protest lasted 938 days.

Participants Numeric Count: 15,076 protests recorded.

Mean: The average number of participants is 19,818.67.

Std: The standard deviation is 156,939.3, indicating significant variability.

Min: The smallest recorded protest had 1 participant.

25% (Q1): 25% of protests had 100 or fewer participants.

50% (Median): The median number of participants is 500.

75% (Q3): 75% of protests had 5,000 or fewer participants.

Max: The largest recorded protest had 7,000,000 participants.

Protester Violence Count: 15,076 protests recorded.

Mean: On average, about 26.47% of protests involved violence (0.264659).

Std: The standard deviation is 0.441166.

Min: No violence (0) in some protests.

25% (Q1): 25% of protests had no violence.

50% (Median): The median value is 0 (no violence).

75% (Q3): 75% of protests had no violence.

Max: Some protests involved violence (1).

Demands (Labor Wage Dispute, Land Farm Issue, etc.) For each demand type, the following statistics are provided:

Count: 15,076 protests recorded.

Mean: The proportion of protests with each demand (e.g., labor wage dispute mean is 0.145264, indicating about 14.53% of protests included this demand).

Std: Standard deviation indicating variability in the presence of each demand.

Min: No (0) demand in some protests.

25% (Q1): 25% of protests did not have this demand.

50% (Median): The median value is 0 (no demand).

75% (Q3): 75% of protests did not have this demand.

Max: Some protests had this demand (1).

Responses (Accommodation, Arrests, Beatings, etc.)

For each response type, the following statistics are provided:

Count: 15,076 protests recorded.

Mean: The proportion of protests with each response (e.g., accommodation mean is 0.099761, indicating about 9.98% of protests resulted in accommodation).

Std: Standard deviation indicating variability in the presence of each response.

Min: No (0) response in some protests.

25% (Q1): 25% of protests did not have this response.

50% (Median): The median value is 0 (no response) . 75% (Q3): 75% of protests did not have this response.

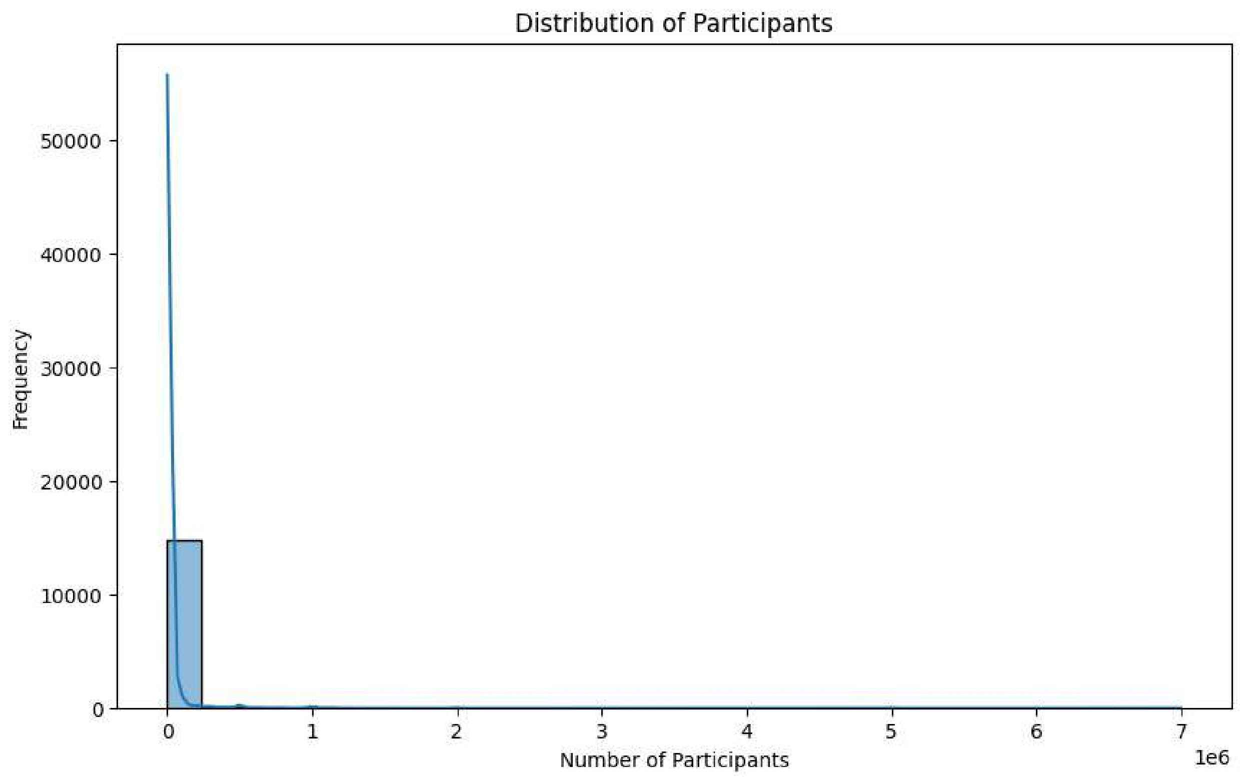
Max: Some protests had this response (1).

This is a summary interpretation of the above statistics

1. Protest Duration: Most protests are short-lived, with a median duration of 0 days.
2. Participants: The number of participants varies widely, with a median of 500 but a mean of around 19,819, indicating some very large protests.
3. Violence: Approximately 26.47% of protests involve violence.
4. Demands: The most common demand is political behavior (about 70.52% of protests), while other demands are less frequent.
5. Responses: Ignoring the protest is the most common response (about 54.40% of protests), while other responses like killings, shootings, and beatings are less common.

These statistics provide an overview of the typical characteristics of protests and the frequency of various demands and responses.

```python
# Plot histogram for participants_numeric
plt.figure(figsize=(10, 6))
sns.histplot(data['participants_numeric'], bins=30, kde=True)
plt.ticklabel_format(style='plain', axis='y')
plt.title('Distribution of Participants')
plt.xlabel('Number of Participants')
plt.ylabel('Frequency')
plt.show()
```

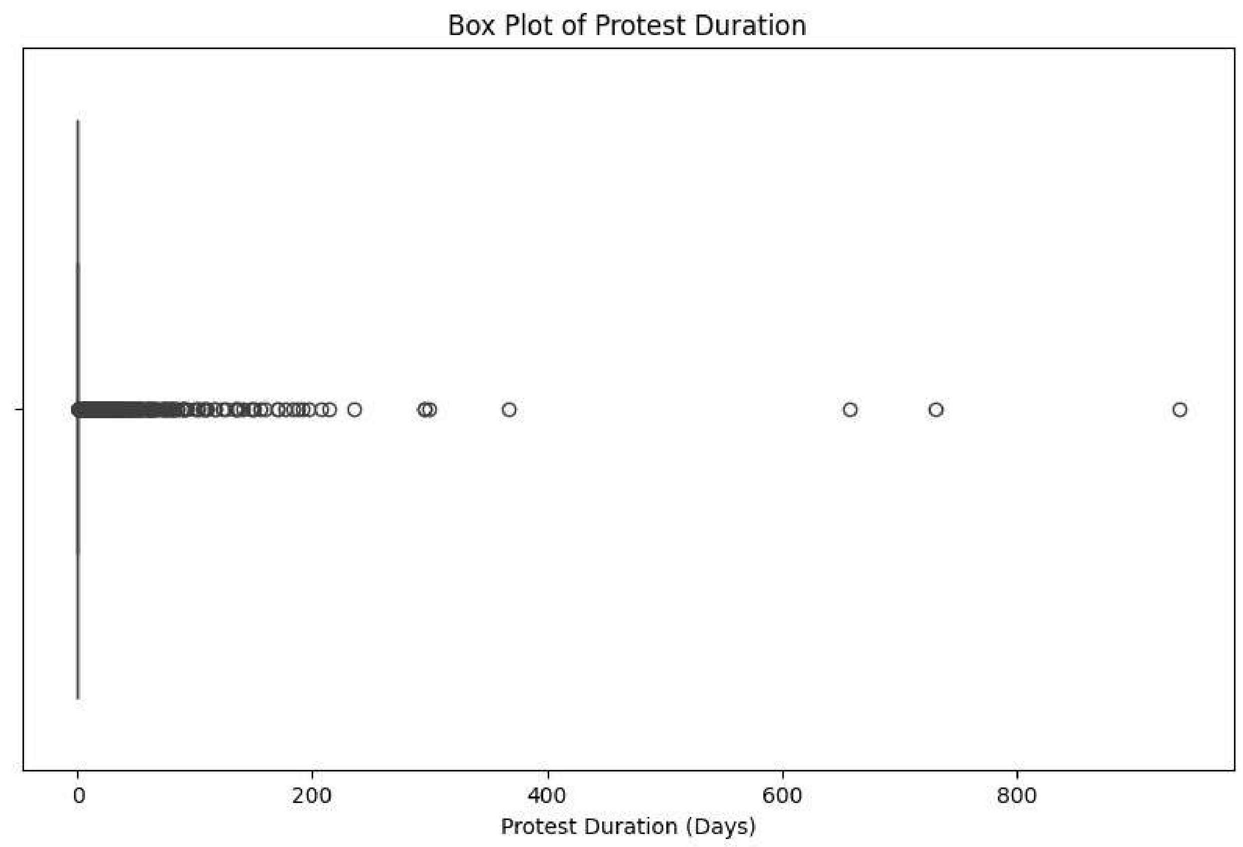


This is a line graph showing the distribution of participants by frequency. The y-axis shows the number of participants, while the x-axis shows the frequency. The x-axis goes from 0 to 7, but it uses scientific notation to represent the largest value (1e6, which means 1,000,000).

Here’s a more detailed interpretation of the graph:

There are very few participants (less than 10,000) who participated in the event 0 to 1 times. The number of participants increases significantly for those who participated 2 times. There are around 30,000 participants in this category. The number of participants continues to increase for those who participated 3 and 4 times. There are around 40,000 and 50,000 participants in these categories, respectively. Participation drops significantly for those who participated 5 times or more. There are less than 10,000 participants in this category. It is important to note that the exact numbers of participants cannot be determined from the graph as the tick marks on the y-axis are not labeled.

```python
# Plot box plot for protest_duration
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['protest_duration'])
plt.title('Box Plot of Protest Duration')
plt.xlabel('Protest Duration (Days)')
plt.show()
```



A box plot is a way to statistically represent the distribution of data. It shows the following:

The center line of the box is the median, which divides the data into two halves. In this case, the median protest duration is around 200 days. The box contains the middle 50% of the data. The upper edge of the box is the 75th percentile, and the lower edge is the 25th percentile. In this case, 75% of the protests lasted less than 400 days, and 25% of the protests lasted less than 100 days. The lines extending from the box are called whiskers. The whiskers extend to the most extreme values in the data that are not considered outliers. Outliers are data points that fall outside of a certain range. In this case, there are outliers on both the left and right side of the plot. The outliers on the left lasted less than 100 days and the outliers on the right lasted more than 600 days. Overall, the box plot shows that most protests lasted between 100 and 400 days. There is a significant number of protests that lasted less than 100 days and a smaller number that lasted longer than 600 days.

```
# Frequency distribution of protesterviolence
violence_counts = data['protesterviolence'].value_counts()
violence_counts
```

```
protesterviolence
0    11086
1     3990
Name: count, dtype: int64
```

The count for the protesterviolence column indicates the number of protests that involved violence (1) and those that did not (0).

Here's a detailed interpretation:

Protester Violence

0 (No Violence): 11,086 protests (73.5%) did not involve violence. 1 (Violence): 3,990 protests (26.5%) involved violence.

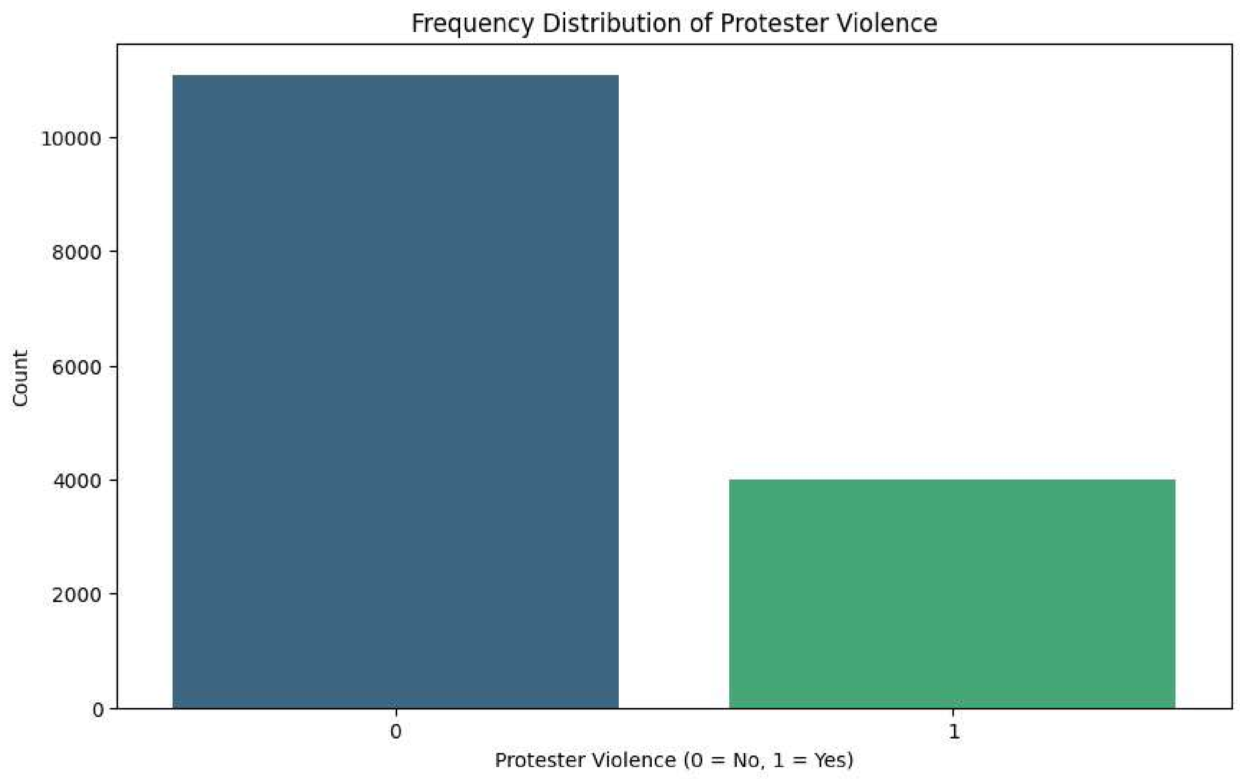
Interpretation

The majority of protests (73.5%) did not involve violence, indicating that most protests were peaceful. About one-quarter (26.5%) of the protests involved some form of violence.

Summary

This data highlights that while a significant portion of protests are non-violent, a notable fraction (over a quarter) involves violence, which is an important aspect to consider when analyzing protest dynamics and responses.

```python
# Plot bar chart for protesterviolence
plt.figure(figsize=(10, 6))
sns.barplot(x=violence_counts.index, y=violence_counts.values,
palette='viridis')
plt.title('Frequency Distribution of Protester Violence')
plt.xlabel('Protester Violence (0 = No, 1 = Yes)')
plt.ylabel('Count')
plt.show()
```



This is a bar chart for the frequency distribution of protester violence.

The Y axis represents the number of protesters which starts at 0 and increases to above 10,000. The X axis categorizes protesters based on their use of violence. It has two bars 0 represents "no violence used" and 1 representing "violence used".

According to the graph we can see the bar of "no violence used" is more than the one where "violence was used".

```
# Get the unique values in the country column
unique_countries = data['country'].unique()
unique_countries
```

```
array(['Canada', 'Cuba', 'Haiti', 'Dominican Republic', 'Jamaica',
       'Mexico', 'Guatemala', 'Honduras', 'El Salvador', 'Nicaragua',
       'Costa Rica', 'Panama', 'Colombia', 'Venezuela', 'Guyana',
       'Suriname', 'Ecuador', 'Peru', 'Brazil', 'Bolivia', 'Paraguay',
       'Chile', 'Argentina', 'Uruguay', 'United Kingdom', 'Ireland',
       'Netherlands', 'Belgium', 'Luxembourg', 'France',
'Switzerland',
       'Spain', 'Portugal', 'Germany', 'Germany West', 'Germany East',
       'Poland', 'Austria', 'Hungary', 'Czechoslovakia', 'Czech
Republic',
       'Slovak Republic', 'Italy', 'Albania', 'Kosovo', 'Serbia',
       'Macedonia', 'Croatia', 'Yugoslavia', 'Bosnia',
       'Serbia and Montenegro', 'Montenegro', 'Slovenia', 'Greece',
       'Cyprus', 'Bulgaria', 'Moldova', 'Romania', 'USSR', 'Russia',
       'Estonia', 'Latvia', 'Lithuania', 'Ukraine', 'Belarus',
'Armenia',
       'Georgia', 'Azerbaijan', 'Finland', 'Sweden', 'Norway',
'Denmark',
       'Cape Verde', 'Guinea-Bissau', 'Equatorial Guinea', 'Gambia',
       'Mali', 'Senegal', 'Benin', 'Mauritania', 'Niger', 'Ivory
Coast',
       'Guinea', 'Burkina Faso', 'Liberia', 'Sierra Leone', 'Ghana',
       'Togo', 'Cameroon', 'Nigeria', 'Gabon', 'Central African
Republic',
       'Chad', 'Congo Brazzaville', 'Congo Kinshasa', 'Uganda',
'Kenya',
       'Tanzania', 'Burundi', 'Rwanda', 'Somalia', 'Djibouti',
       'South Sudan', 'Ethiopia', 'Eritrea', 'Angola', 'Mozambique',
       'Zambia', 'Zimbabwe', 'Malawi', 'South Africa', 'Namibia',
       'Lesotho', 'Botswana', 'Swaziland', 'Madagascar', 'Comoros',
       'Mauritius', 'Morocco', 'Algeria', 'Tunisia', 'Libya', 'Sudan',
       'Iran', 'Turkey', 'Iraq', 'Egypt', 'Syria', 'Lebanon',
'Jordan',
       'Saudi Arabia', 'Yemen', 'Kuwait', 'Bahrain', 'Qatar',
       'United Arab Emirate', 'Oman', 'Afghanistan', 'Turkmenistan',
       'Tajikistan', 'Kyrgyzstan', 'Uzbekistan', 'Kazakhstan',
'China',
       'Mongolia', 'Taiwan', 'North Korea', 'South Korea', 'Japan',
       'India', 'Bhutan', 'Pakistan', 'Bangladesh', 'Myanmar',
       'Sri Lanka', 'Nepal', 'Thailand', 'Cambodia', 'Laos',
'Vietnam',
       'Malaysia', 'Singapore', 'Philippines', 'Indonesia', 'Timor
Leste',
       'Papua New Guinea'], dtype=object)
```

This is a list of countries in the dataset. This dataset has a geographical diversity as the countries span from North America, South America, Europe, Africa, Asia, and Oceania, reflecting a broad geographic scope in the protest dataset.

```
# Count of protests by country
country_counts = data['country'].value_counts()
country_counts

country
United Kingdom           575
France                   546
Ireland                  431
Germany                  364
Kenya                    350
                        ...
Serbia and Montenegro      2
Laos                       2
```

```
Bhutan                           2
South Sudan                      1
Qatar                            1
Name: count, Length: 166, dtype: int64
```

These shows the number of protests by country. The highest being United Kingdom which had 575 protests and the lowest being South Sudan having 1 protest.

```
from IPython.core.display import HTML

# Embed Tableau Public visualization in Jupyter Notebook
HTML("""
<div class='tableauPlaceholder' id='viz1721550756321' style='position:
relative'>
    <noscript>
        <a href='#'>
            <img alt='Count of protests by country from 1990-2020'
src='https://public.tableau.com/static/images/Ca/CapstoneProject_17215
496480900/Sheet1/1_rss.png' style='border: none' />
        </a>
    </noscript>
    <object class='tableauViz' style='display:none;'>
        <param name='host_url' value='https://public.tableau.com/' />
        <param name='embed_code_version' value='3' />
        <param name='site_root' value='' />
        <param name='name'
value='CapstoneProject_17215496480900/Sheet1' />
        <param name='tabs' value='no' />
        <param name='toolbar' value='yes' />
        <param name='static_image'
value='https://public.tableau.com/static/images/Ca/CapstoneProject_172
15496480900/Sheet1/1.png' />
        <param name='animate_transition' value='yes' />
        <param name='display_static_image' value='yes' />
        <param name='display_spinner' value='yes' />
        <param name='display_overlay' value='yes' />
        <param name='display_count' value='yes' />
        <param name='language' value='en-US' />
        <param name='filter' value='publish=yes' />
    </object>
</div>
<script type='text/javascript'>
    var divElement = document.getElementById('viz1721550756321');
    var vizElement = divElement.getElementsByTagName('object')[0];
    vizElement.style.width='100%';
    vizElement.style.height=(divElement.offsetWidth*0.75)+'px';
    var scriptElement = document.createElement('script');
    scriptElement.src =
'https://public.tableau.com/javascripts/api/viz_v1.js';
```

```
    vizElement.parentNode.insertBefore(scriptElement, vizElement);
</script>
""")


<IPython.core.display.HTML object>
```

This is showing the number of protests recorded for each country in the dataset. Here's a summary interpretation of the data:

Top Countries by Number of Protests United Kingdom: 575 protests France: 546 protests Ireland: 431 protests Germany: 364 protests Kenya: 350 protests

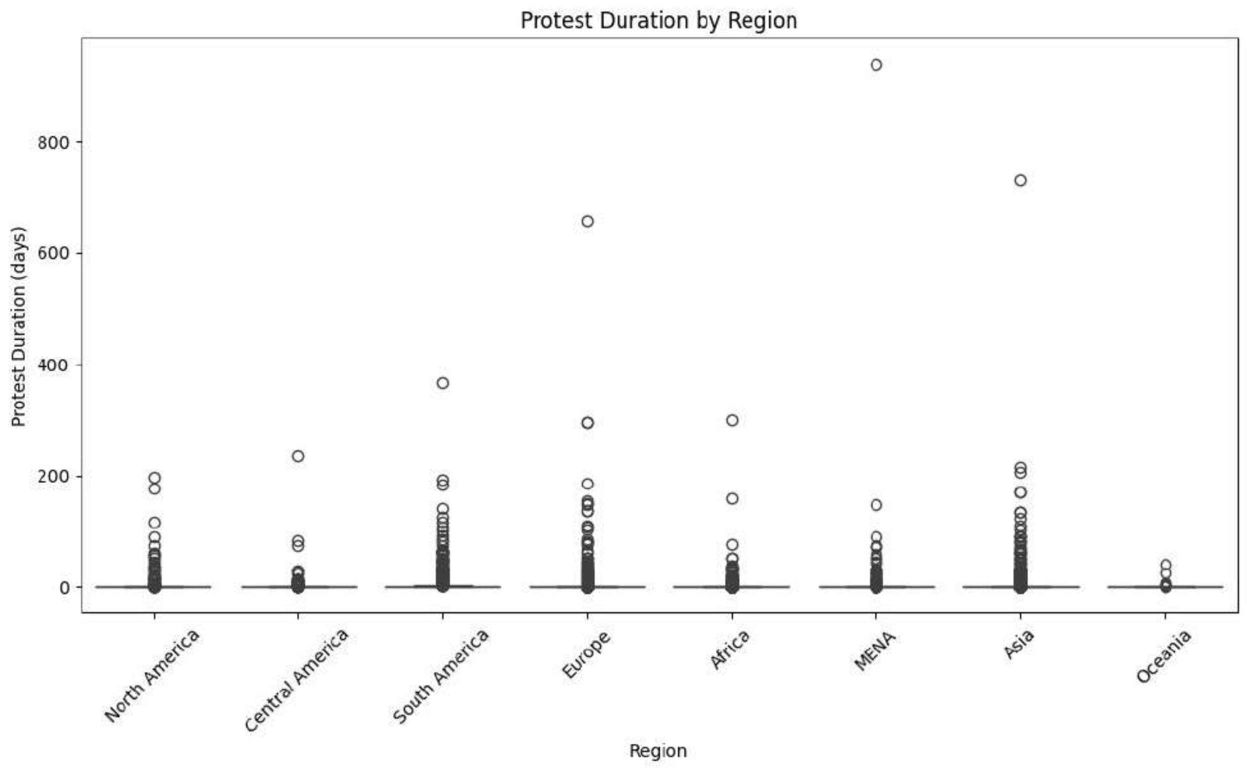
Bottom Countries by Number or Protests Germany West: 2 protests Laos: 2 protests Bhutan: 2 protests Qatar: 1 protest South Sudan: 1 protest

Observations

Distribution: The counts vary widely, with some countries having hundreds of recorded protests while others have only a few.

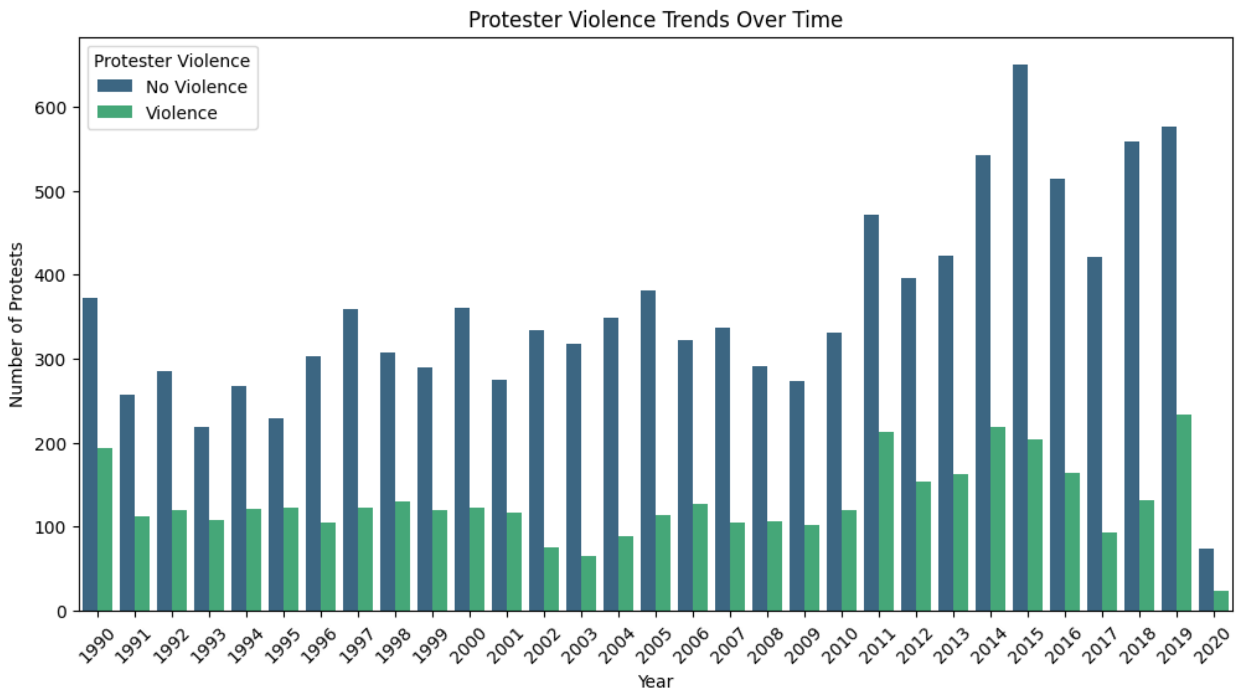
Regional Representation: Countries from various continents appear on the list, indicating protests are a global phenomenon.

```
#  Visualize Protest Duration by Region
plt.figure(figsize=(12, 6))
sns.boxplot(x='region', y='protest_duration', data=data)
plt.title('Protest Duration by Region')
plt.xlabel('Region')
plt.ylabel('Protest Duration (days)')
plt.xticks(rotation=45)
plt.show()
```



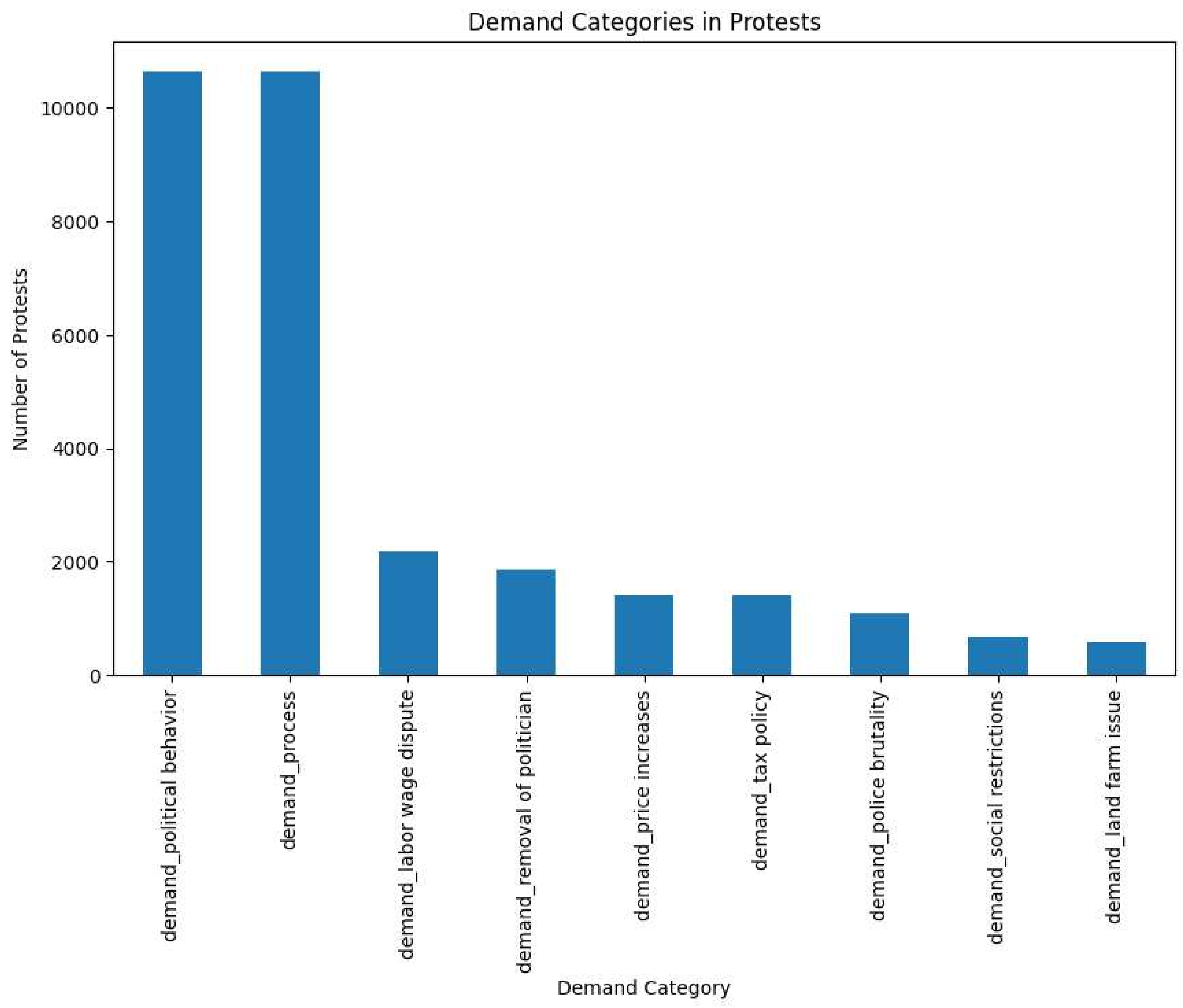
The graph depicting protest durations by region reveals that in most countries, protests lasted from 0 days (indicating they ended on the same day they began) to around 300 days. However, the MENA region notably stands out with protest durations exceeding 800 days, while both Europe and Asia also saw protests lasting over 600 days.

```
# Explore Protester Violence Trends Over Time
plt.figure(figsize=(12, 6))
sns.countplot(x='year', hue='protesterviolence', data=data,
palette='viridis')
plt.title('Protester Violence Trends Over Time')
plt.xlabel('Year')
plt.ylabel('Number of Protests')
plt.legend(title='Protester Violence', labels=['No Violence',
'Violence'])
plt.xticks(rotation=45)
plt.show()
```



The graph illustrates the frequency of protester violence over time across various regions. In 2015, the highest number of protests without violence was recorded, surpassing 600 incidents. Conversely, in 2019, the highest number of violent protests was observed, exceeding 200 incidents.

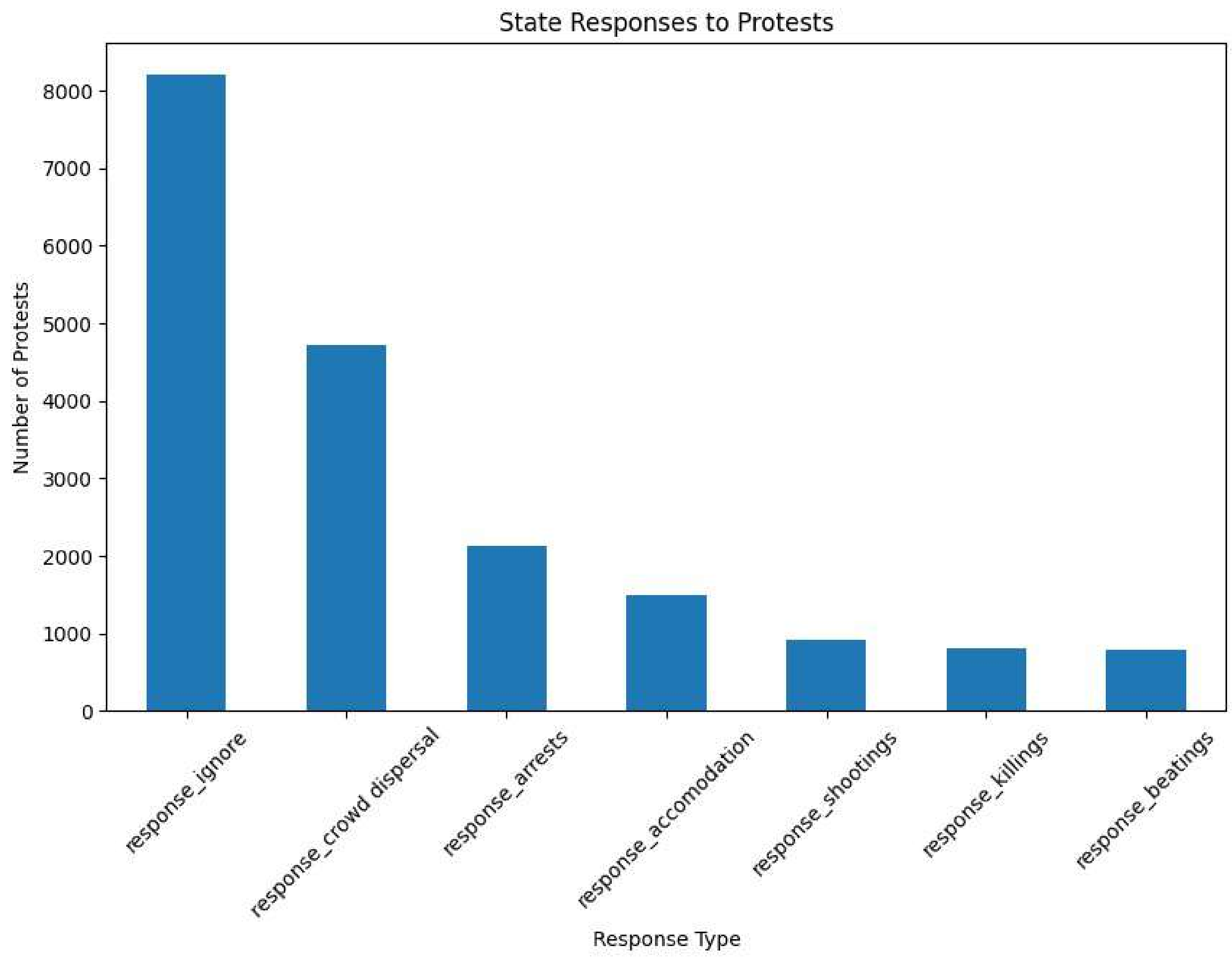
```
# Analyze Demand Categories
demand_columns = [col for col in data.columns if
col.startswith('demand_')]
demand_counts =
data[demand_columns].sum().sort_values(ascending=False)

plt.figure(figsize=(10, 6))
demand_counts.plot(kind='bar')
plt.title('Demand Categories in Protests')
plt.xlabel('Demand Category')
plt.ylabel('Number of Protests')
plt.xticks(rotation=90)
plt.show()
```



The above graph shows the most common protest demands are in the following categories:

1. Economic Justice
2. Social Justice
3. Demand_political behavior
4. Demand_process

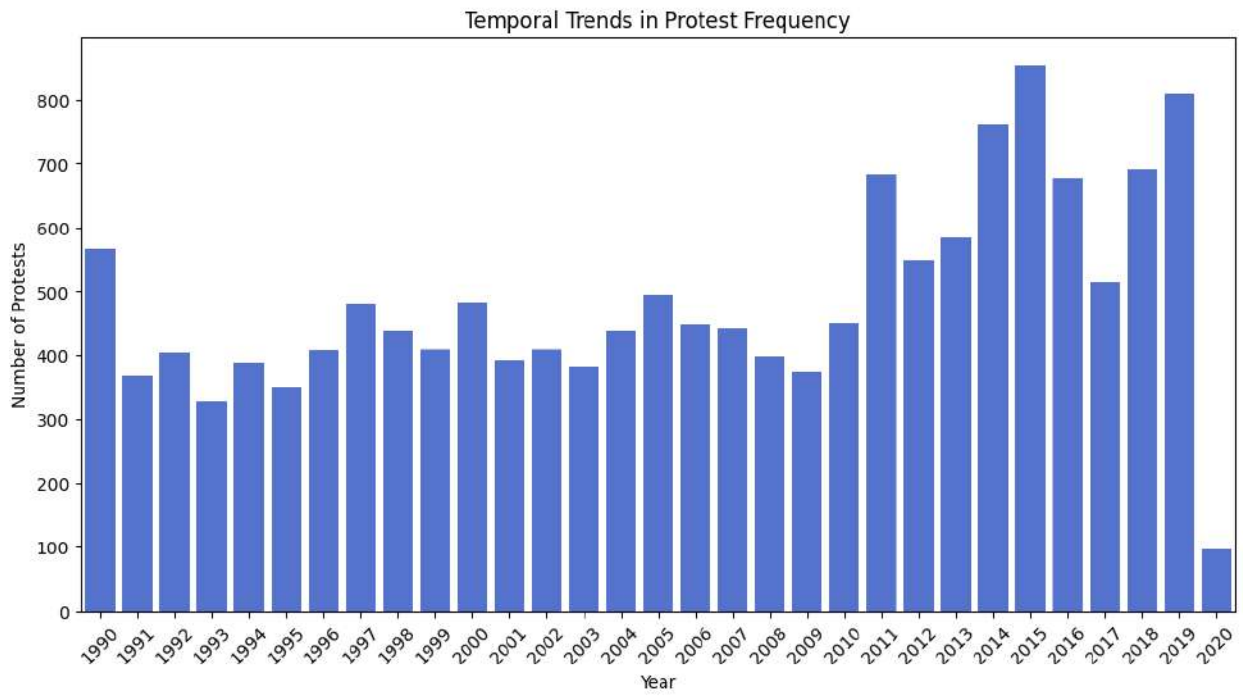
```
# Evaluate State Responses
response_columns = [col for col in data.columns if
col.startswith('response_')]
response_counts =
data[response_columns].sum().sort_values(ascending=False)

plt.figure(figsize=(10, 6))
response_counts.plot(kind='bar')
plt.title('State Responses to Protests')
plt.xlabel('Response Type')
plt.ylabel('Number of Protests')
plt.xticks(rotation=45)
plt.show()
```



The graph shows how states responded to protests:

- Use of force: Arrest is the most common forceful response, followed by shootings, killings, beatings, and crowd dispersal.
- Non-confrontational responses: Ignoring protests is the most common non-confrontational response, followed by accommodation.

```
# Temporal Trends in Protest Frequency
plt.figure(figsize=(12, 6))
sns.countplot(x='year', data=data, color='#4169E1')
plt.title('Temporal Trends in Protest Frequency')
plt.xlabel('Year')
plt.ylabel('Number of Protests')
plt.xticks(rotation=45)
plt.show()
```



The visualization reveals fluctuations in protest frequency over the years, offering insights into patterns and trends in protest activity over time. The lowest frequency of protests occurred in 2020, likely influenced by the global COVID-19 pandemic. Conversely, the highest frequency of protests was observed in 2015.

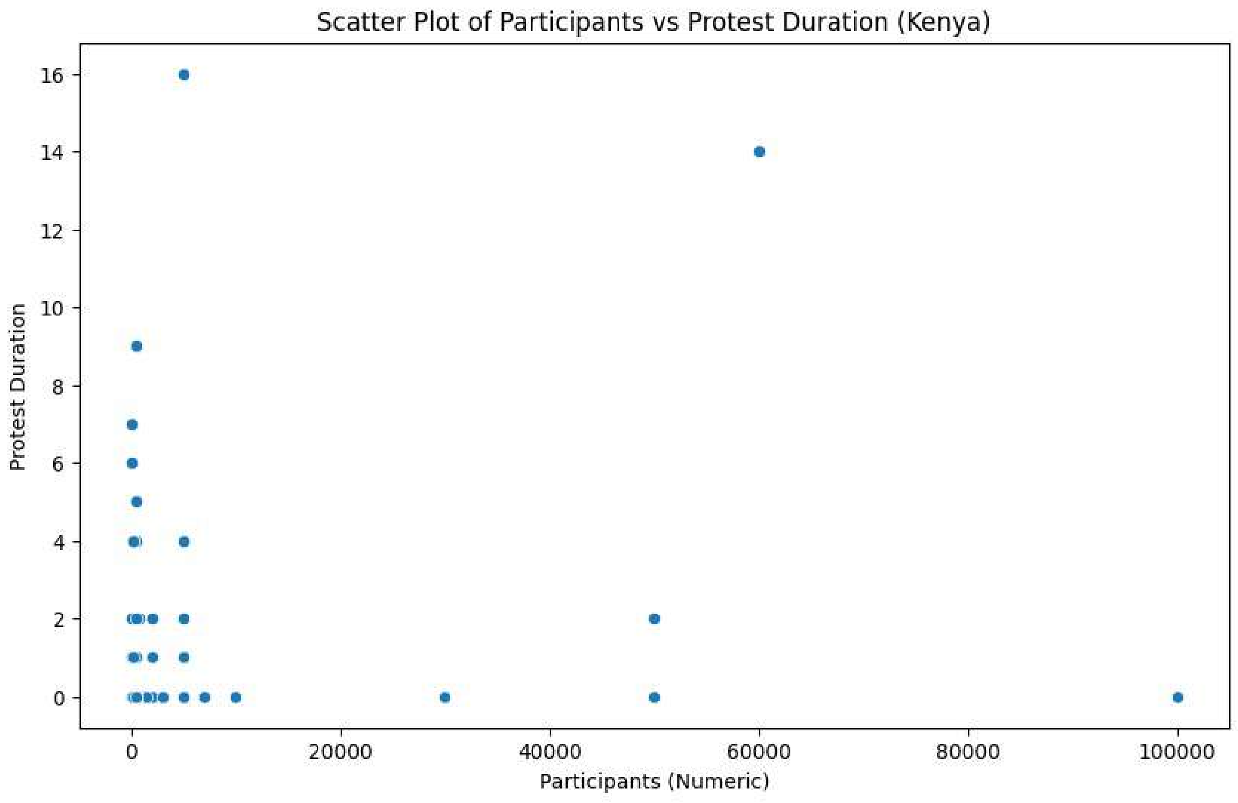
BIVARIATE ANALYSIS

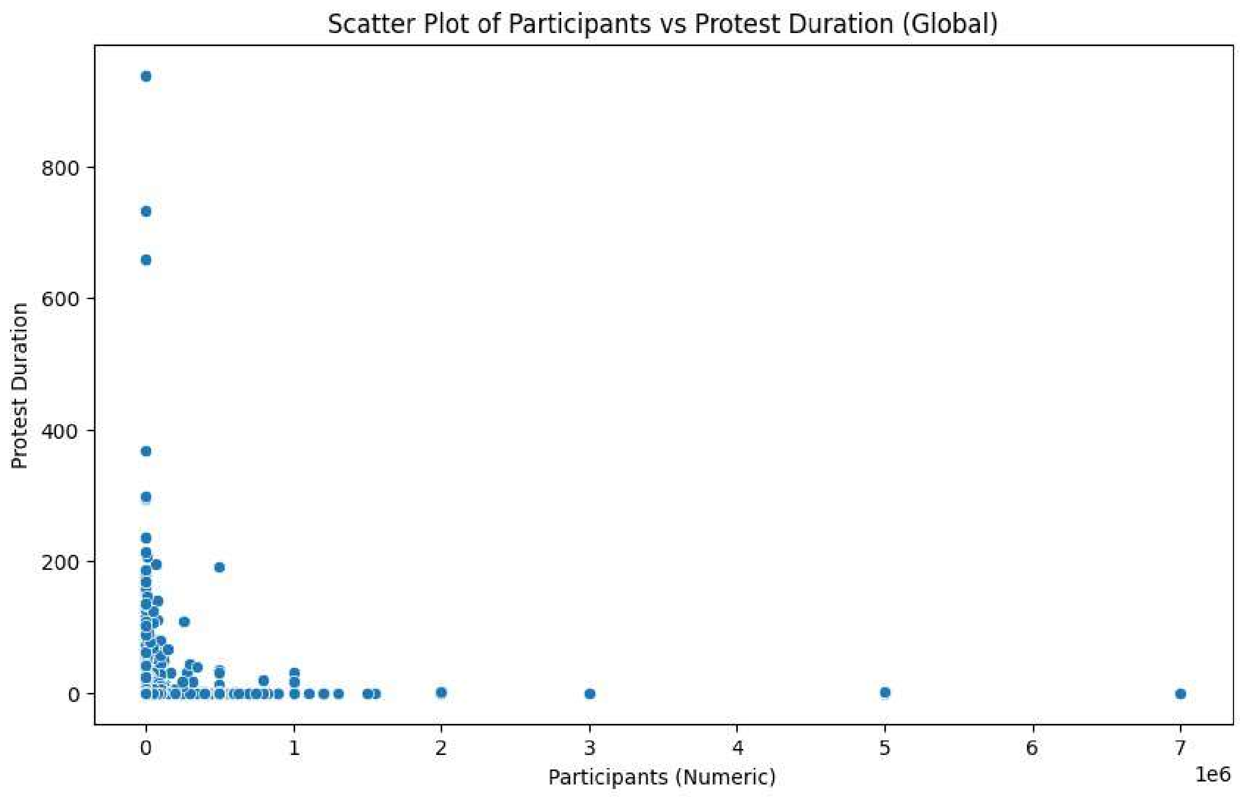
Bivariate analysis is a statisctical method used to examine the relationship between two variables. The goal is to understand whether and how the two variables are related, and if so, to characterize the nature of that relationship.

```
# Filter data for Kenya
kenya_data = data[data['country'] == 'Kenya']
```

NUMERICAL VS NUMERICAL

```
# Kenya
plt.figure(figsize=(10, 6))
sns.scatterplot(data=kenya_data, x='participants_numeric',
y='protest_duration')
plt.title('Scatter Plot of Participants vs Protest Duration (Kenya)')
plt.xlabel('Participants (Numeric)')
plt.ylabel('Protest Duration')
plt.show()

# Global
plt.figure(figsize=(10, 6))
sns.scatterplot(data=data, x='participants_numeric',
y='protest_duration')
plt.title('Scatter Plot of Participants vs Protest Duration (Global)')
plt.xlabel('Participants (Numeric)')
plt.ylabel('Protest Duration')
plt.show()
```





(1e6, means 1,000,000)

For both Kenya and the World, the longer the protest the less the number of participants.

The Kenya plot is showing:

The plot shows individual data points representing each protest. Each point is positioned according to the number of participants and the duration of the respective protest.

Observations:

Spread: The data points are widely spread across the plot, indicating a large range in both the number of participants and protest duration.

Clusters: There appears to be a cluster of protests with a relatively small number of participants and shorter durations, suggesting a concentration of smaller-scale protests.

Outliers: A few data points with significantly higher participant numbers and longer durations can be considered outliers. These might represent particularly large or prolonged protests.

Relationship: It's difficult to discern a clear linear relationship between the number of participants and protest duration. The data points seem scattered without a strong trend.

The plot suggests that protest size (number of participants) doesn't necessarily correlate strongly with protest duration in Kenya.

There is a wide range of protest sizes and durations,indicating diverse protest activities in the country.

Further analysis with additional data points or statistical measures could reveal potential correlations or patterns.

The global plot is showing:

Distribution: The majority of data points cluster in the lower left corner, indicating that most protests involve a relatively small number of participants and have shorter durations.

Outliers: There are a few outliers with a high number of participants and longer durations, suggesting that some protests are significantly larger and more prolonged.

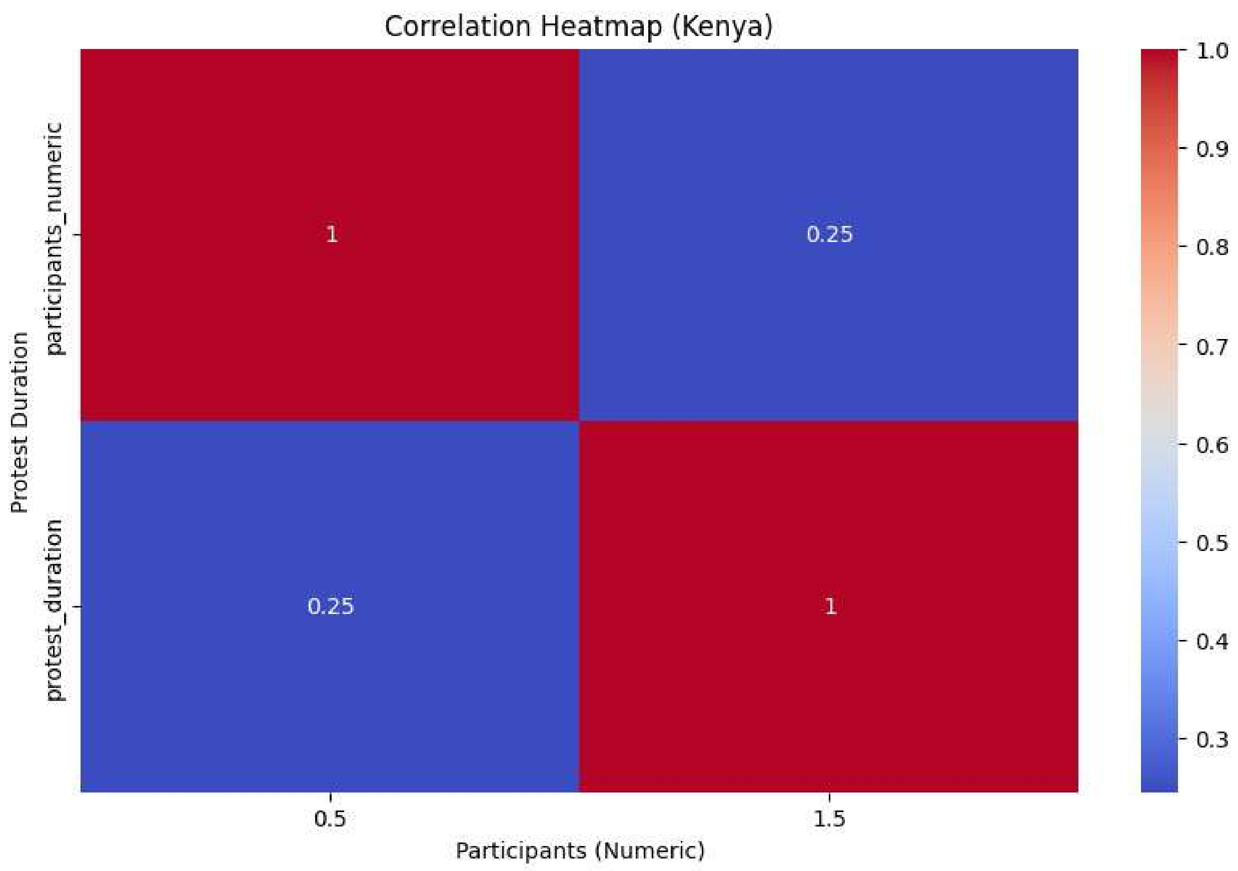
No Clear Correlation: There doesn't seem to be a strong linear relationship between the number of participants and protest duration. The data points are scattered without a clear pattern.

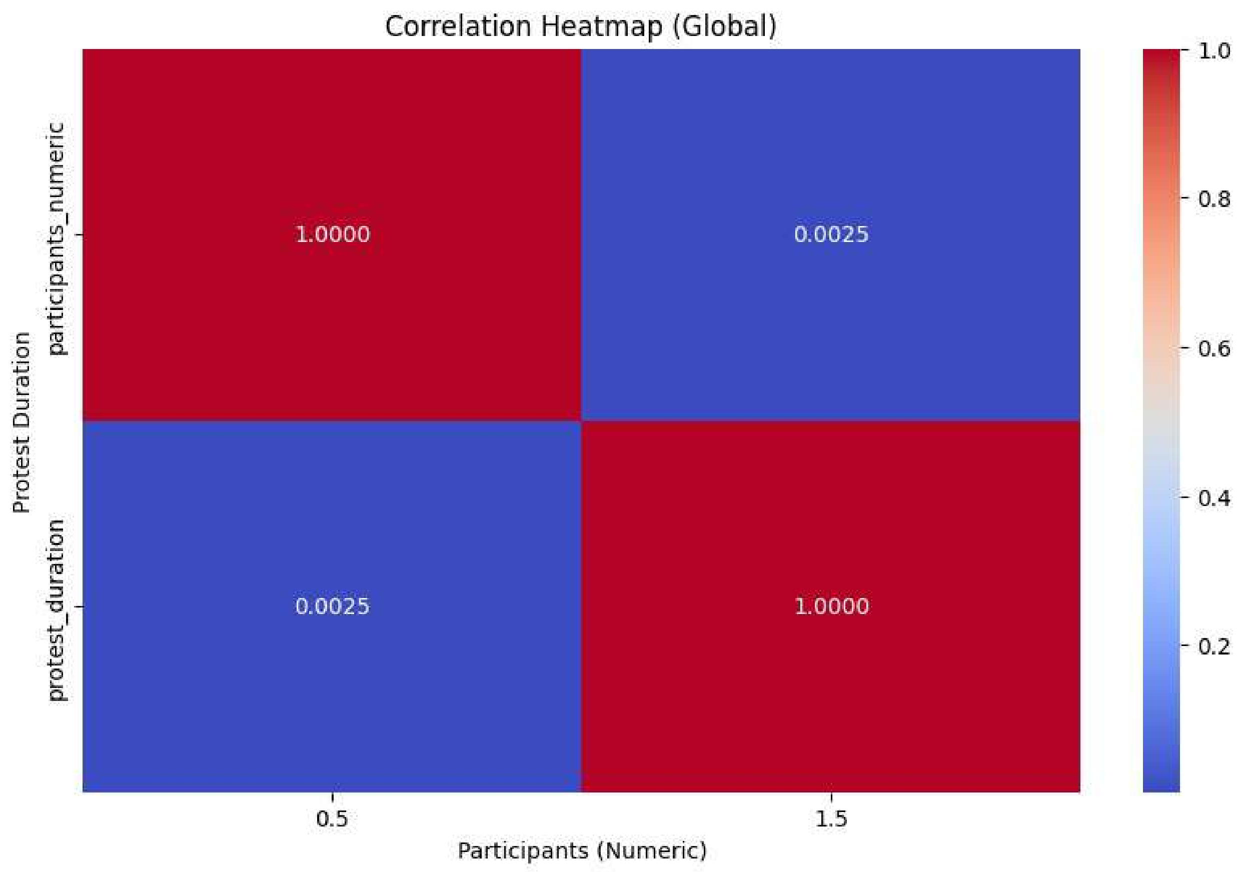
Diverse Protest Landscape: The wide distribution of data points suggests that protests vary greatly in size and length, reflecting the diverse nature of social and political movements worldwide.

Factors Beyond Participant Numbers: Factors other than the number of participants likely influence protest duration, such as the nature of the issue, government response, and social conditions.

Limitations of the Data: The plot doesn't provide information about the specific time period, types of protests, or geographic distribution of the data, which could affect the interpretation.

```python
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import ScalarFormatter

# Kenya
corr_kenya = kenya_data[['participants_numeric',
'protest_duration']].corr()
plt.figure(figsize=(10, 6))
sns.heatmap(corr_kenya, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap (Kenya)')
plt.xlabel('Participants (Numeric)')
plt.ylabel('Protest Duration')
plt.gca().xaxis.set_major_formatter(ScalarFormatter())
plt.show()

# Global
corr_global = data[['participants_numeric',
'protest_duration']].corr()
plt.figure(figsize=(10, 6))
sns.heatmap(corr_global, annot=True, cmap='coolwarm', fmt='.4f')
plt.title('Correlation Heatmap (Global)')
plt.xlabel('Participants (Numeric)')
plt.ylabel('Protest Duration')
plt.gca().xaxis.set_major_formatter(ScalarFormatter())
plt.show()
```

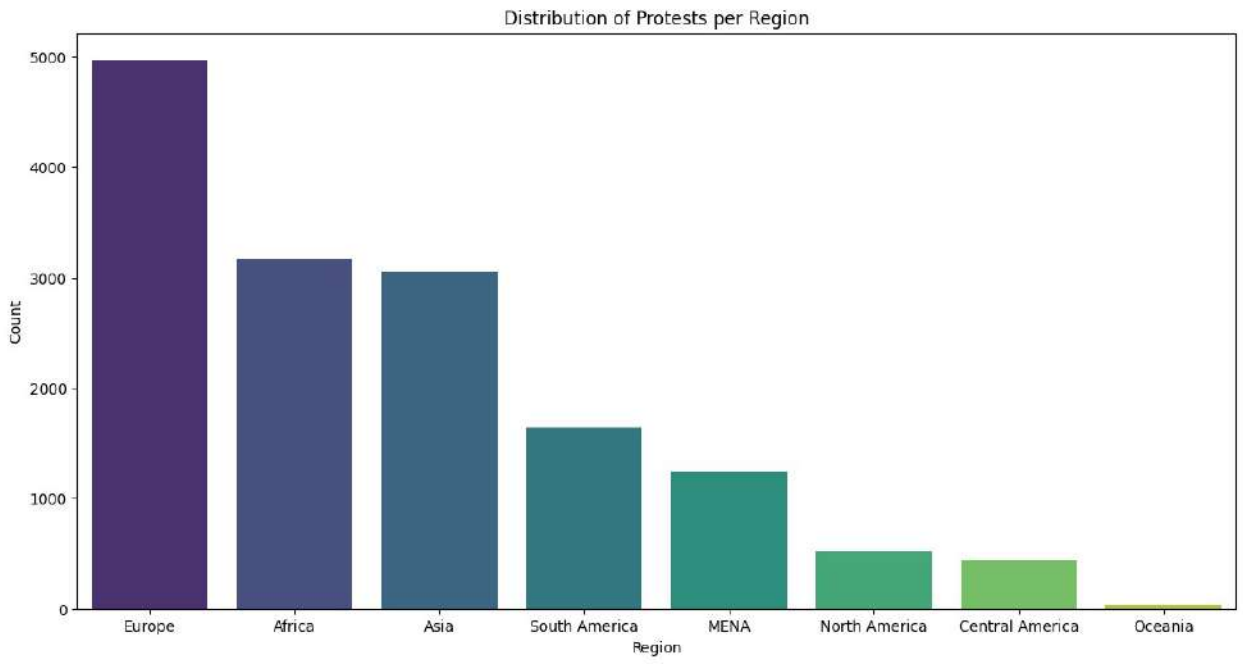

Correlation Heatmap (Kenya)
Protest Duration
participants_numeric
protest_duration
1
0.25
0.25
1
0.5
1.5
Participants (Numeric)
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3

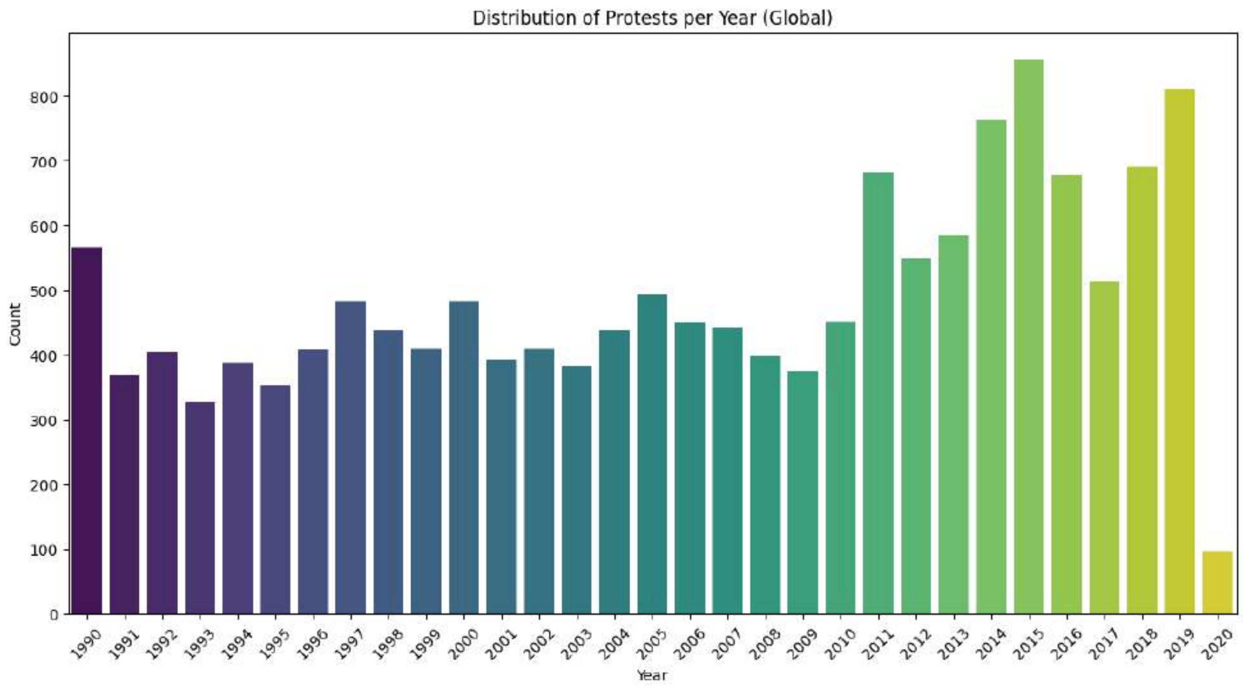


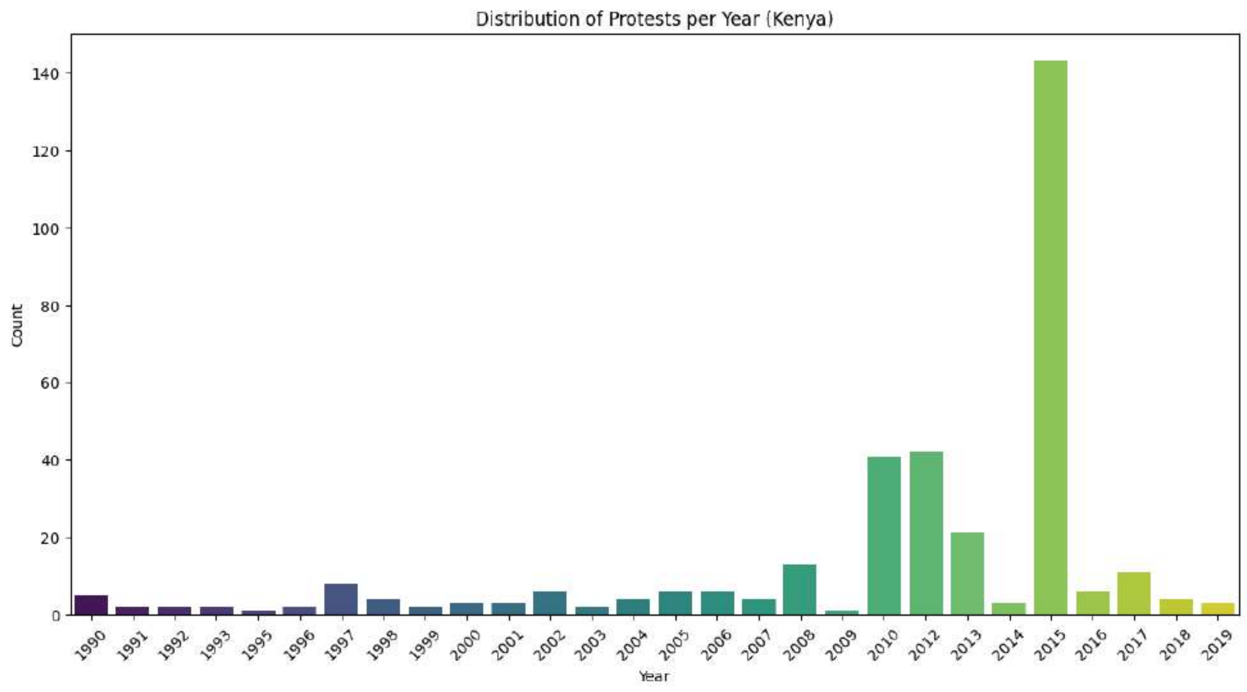
- In Kenya protest duration has a slightly +ve impact on participant turnout.
- Globally this has the same effect but on a much smaller scale, as the correlation is 0.0025.

```
# Distribution of protests by region
plt.figure(figsize=(14, 7))
sns.countplot(x=data['region'], palette="viridis",
order=data['region'].value_counts().index)
plt.title('Distribution of Protests per Region')
plt.xlabel('Region')
plt.ylabel('Count')
plt.show()

# Global
plt.figure(figsize=(14, 7))
sns.countplot(x=data['year'], palette="viridis")
plt.title('Distribution of Protests per Year (Global)')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

# Kenya
plt.figure(figsize=(14, 7))
```

```python
sns.countplot(data=kenya_data, x='year', palette="viridis")
plt.title('Distribution of Protests per Year (Kenya)')
plt.xlabel('Year')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

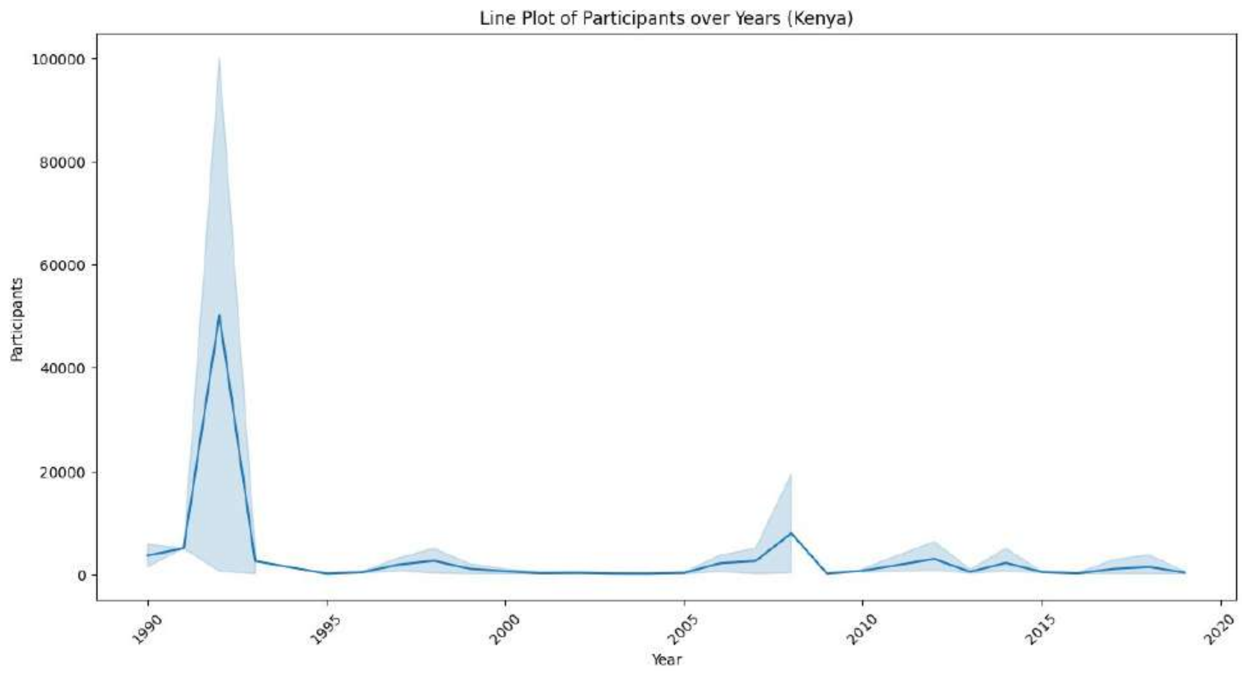


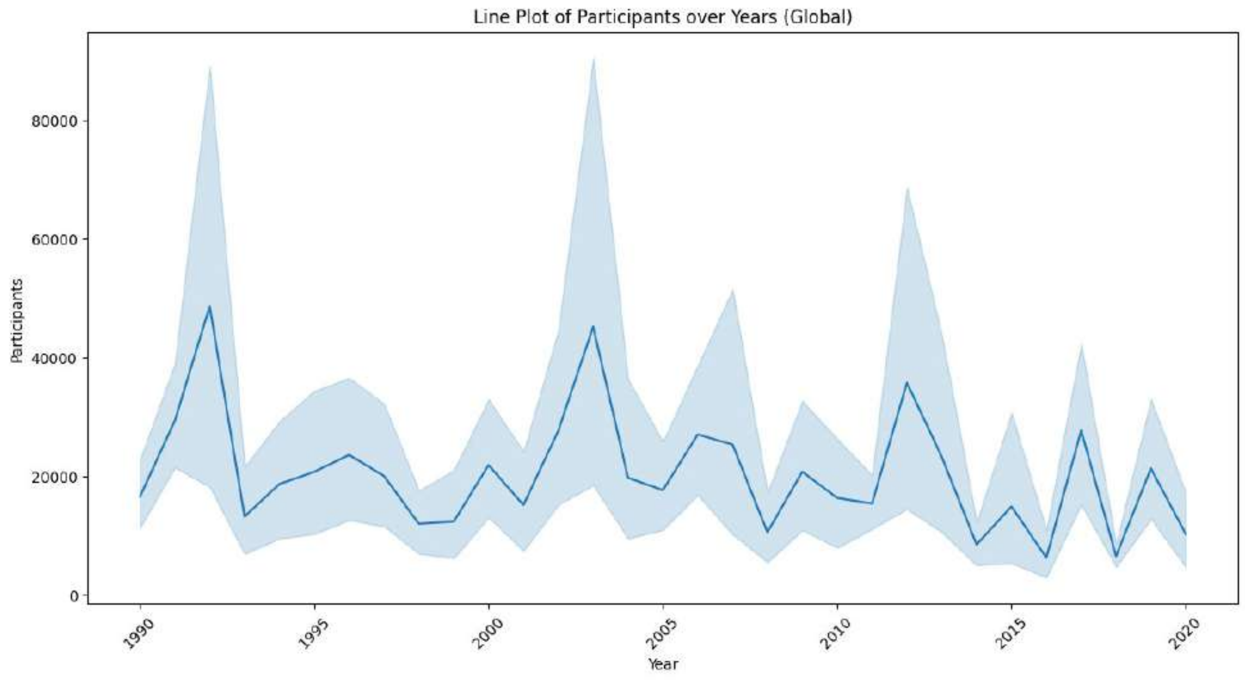




- Europe had the most recorded protests with Oceania recording the least.
- The distribution of protests took an upward turn in the 2010's.
- Kenya had a record high protests in 2015. This was also observed in other nations.

```
# Kenya
plt.figure(figsize=(14, 7))
sns.lineplot(data=kenya_data, x='year', y='participants_numeric')
plt.title('Line Plot of Participants over Years (Kenya)')
plt.xlabel('Year')
plt.ylabel('Participants')
plt.xticks(rotation=45)
plt.show()

# Global
plt.figure(figsize=(14, 7))
sns.lineplot(data=data, x='year', y='participants_numeric')
plt.title('Line Plot of Participants over Years (Global)')
plt.xlabel('Year')
plt.ylabel('Participants')
plt.xticks(rotation=45)
plt.show()
```





The image is a line plot showing the number of participants in protests in Kenya over the years, with the x-axis labeled as "Participants (Numeric)" and the y-axis labeled as "Protest Duration." The plot indicates a significant spike in protest participation around the early 1990s, with over 100,000 participants. After this peak, the number of participants drastically declines and remains low with smaller fluctuations over the years, especially between 2000 and 2020. The shaded area around the line suggests a confidence interval or variability in the data points.

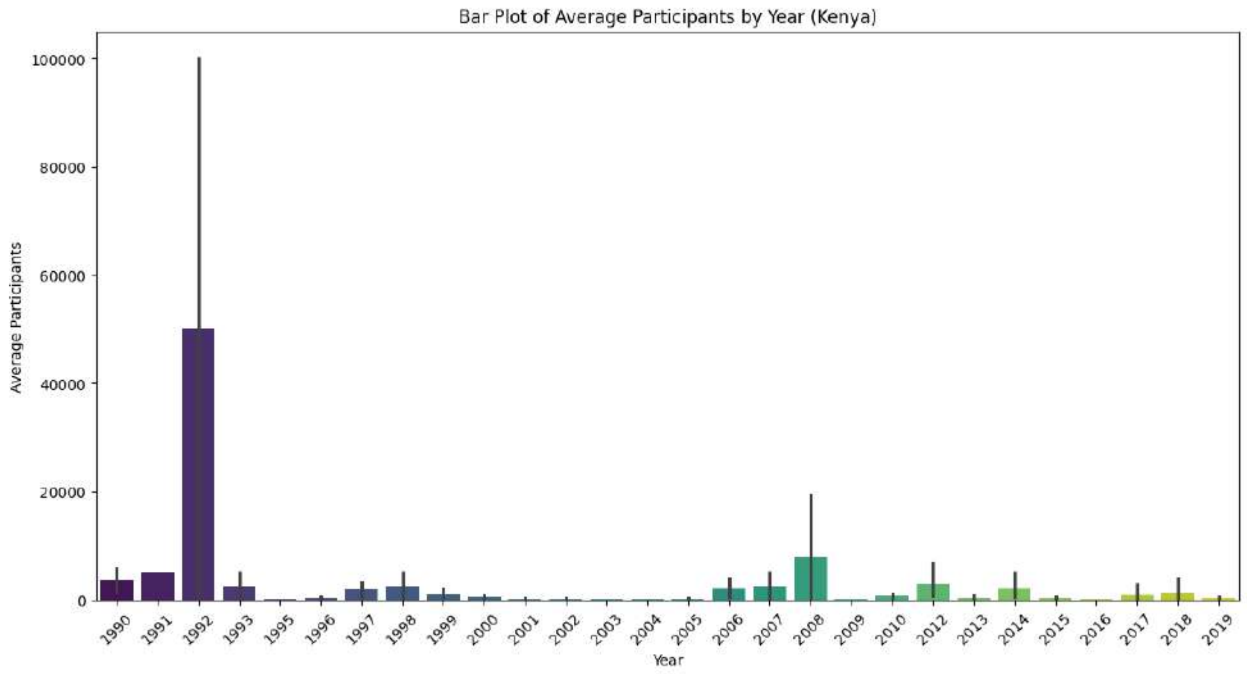
Overall, the plot suggests that there was a major protest event in the early 1990s in Kenya, after which the scale of protests significantly reduced and stabilized at much lower levels.

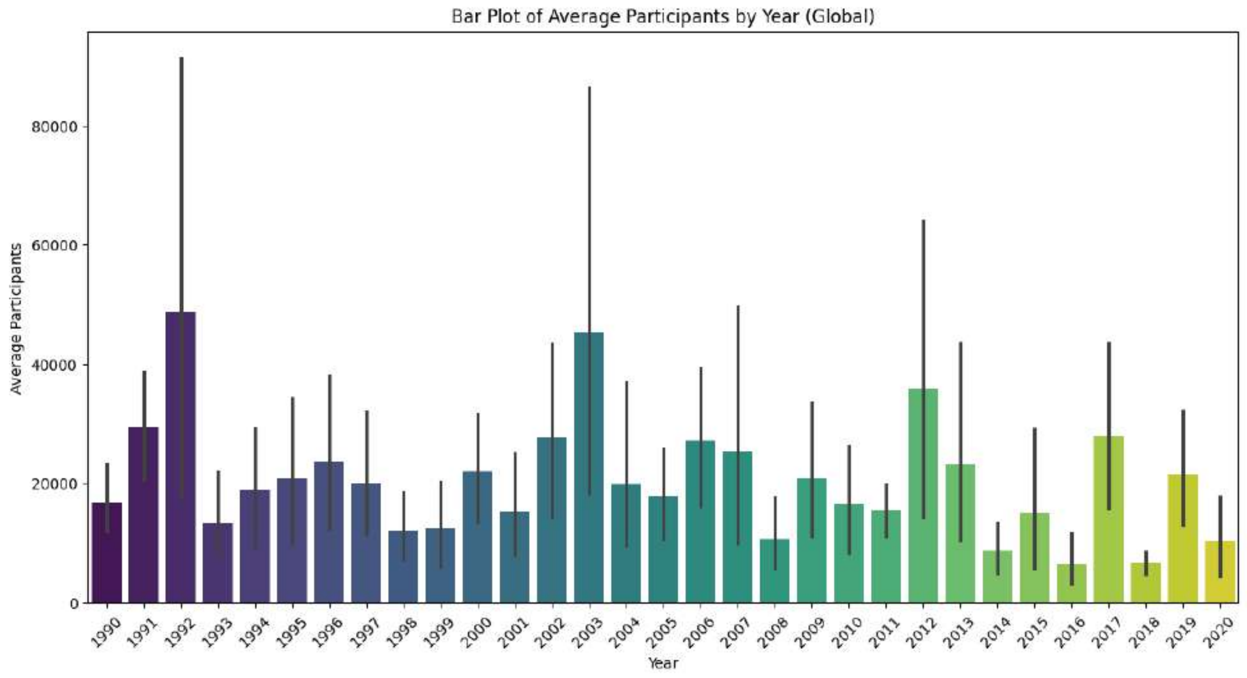
The image is a line plot showing global protest participation over the years, with the x-axis labeled "Participants (Numeric)" and the y-axis labeled "Protest Duration." The plot displays several peaks and valleys, indicating fluctuations in the number of protest participants worldwide.

Key observations:

There are noticeable spikes around the early 1990s, late 1990s, and mid-2000s, suggesting significant global protest events during these periods. After the mid-2000s, the number of participants in protests seems to decrease overall, with smaller spikes occurring in the 2010s. The shaded area around the line indicates variability or confidence intervals, showing that the participation numbers had more variability in some years than others. Overall, this plot suggests that global protest participation has seen considerable fluctuations over time, with some periods marked by significant global unrest. The trend appears to show more dispersed and smaller-scale protests in recent years compared to earlier decades.

```python
# Kenya
plt.figure(figsize=(14, 7))
sns.barplot(data=kenya_data, x='year', y='participants_numeric',
estimator=np.mean, palette="viridis")
plt.title('Bar Plot of Average Participants by Year (Kenya)')
plt.xlabel('Year')
plt.ylabel('Average Participants')
plt.xticks(rotation=45)
plt.show()

# Global
plt.figure(figsize=(14, 7))
data_sorted_by_year = data.sort_values('year')
sns.barplot(data=data_sorted_by_year, x='year',
y='participants_numeric', estimator=np.mean, palette="viridis")
plt.title('Bar Plot of Average Participants by Year (Global)')
plt.xlabel('Year')
plt.ylabel('Average Participants')
plt.xticks(rotation=45)
plt.show()
```





Observations on barplot of average participants per year

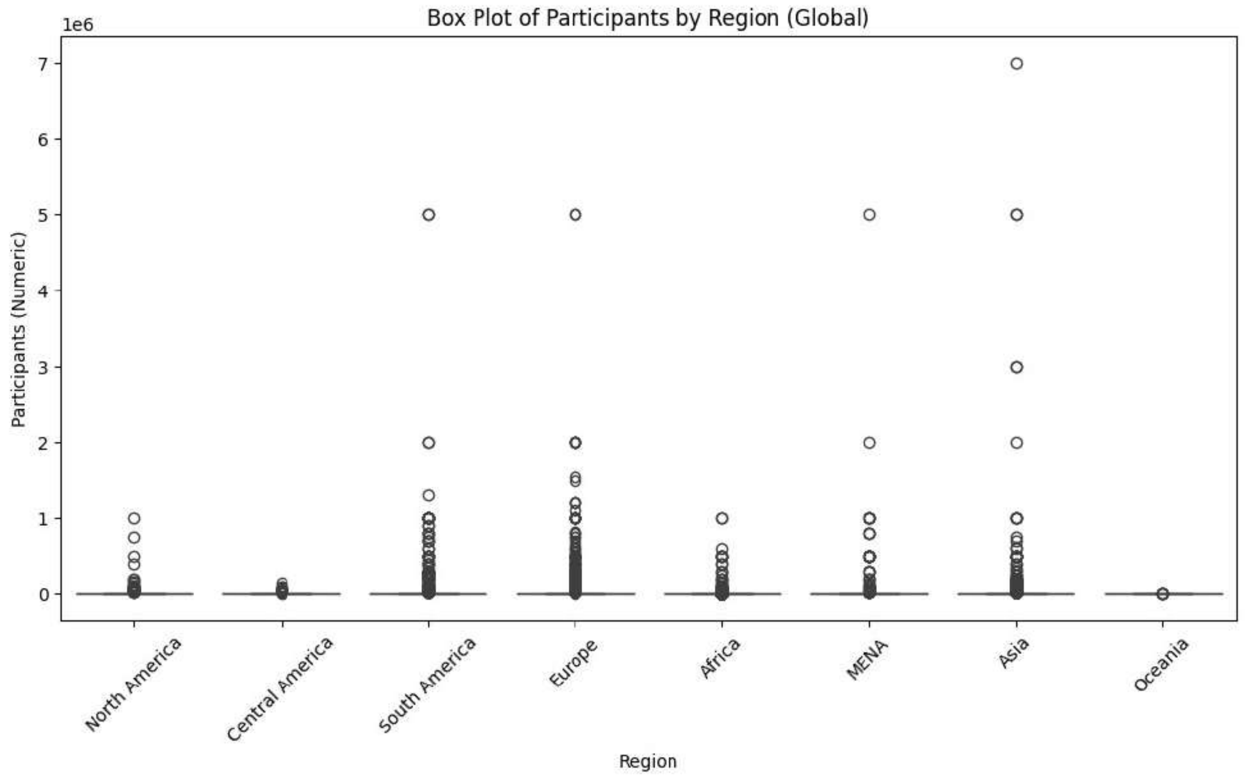
The number of participants varies significantly from year to year. There are a few years with exceptionally high numbers of participants, particularly around 1991-1992 and 2007. The number of participants seems to fluctuate between high and low points throughout the years. The graph suggests that there have been periods of high and low participation in Kenya over the years.

Bar plot showing the average number of participants in protests across different years.

Observations:

The number of participants varies significantly from year to year, with some years showing exceptionally high numbers. There is a general upward trend in the number of participants over time, with some notable spikes and dips. The error bars indicate a range of uncertainty in the average number of participants for each year.

```
#  Visualize Protest Duration by Region
plt.figure(figsize=(12, 6))
sns.boxplot(x='region', y='participants_numeric', data=data)
plt.title('Box Plot of Participants by Region (Global)')
plt.xlabel('Region')
plt.ylabel('Participants (Numeric)')
plt.xticks(rotation=45)
plt.show()
```

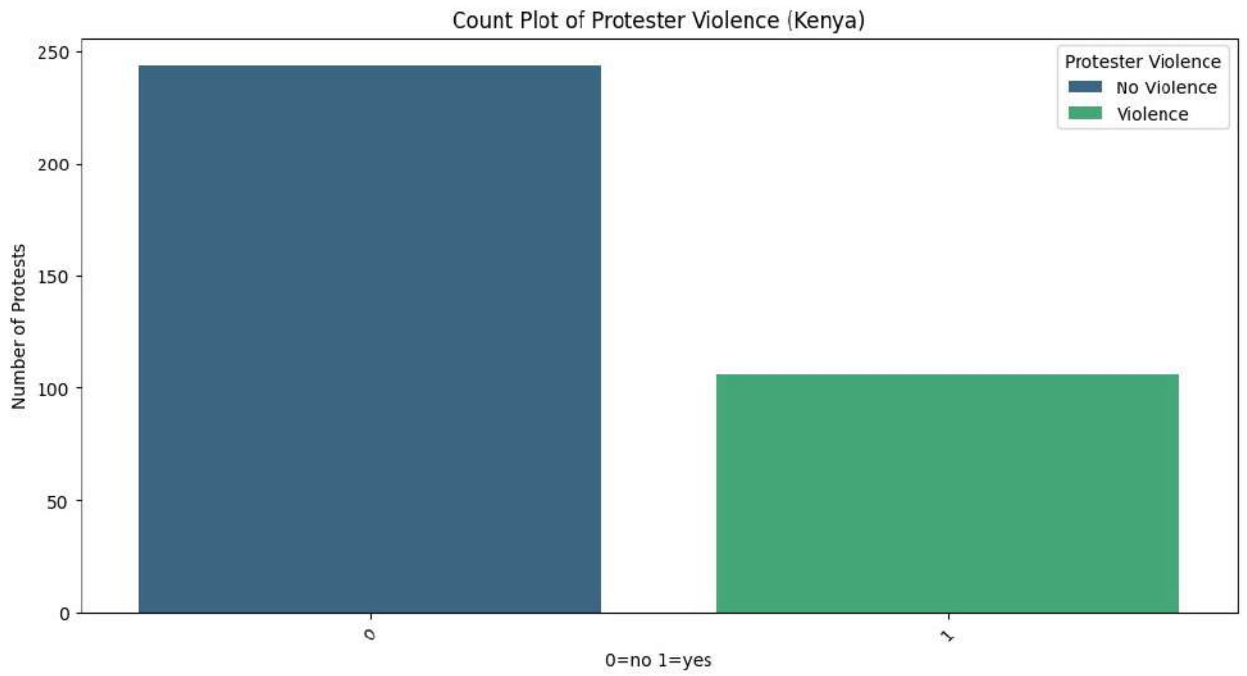


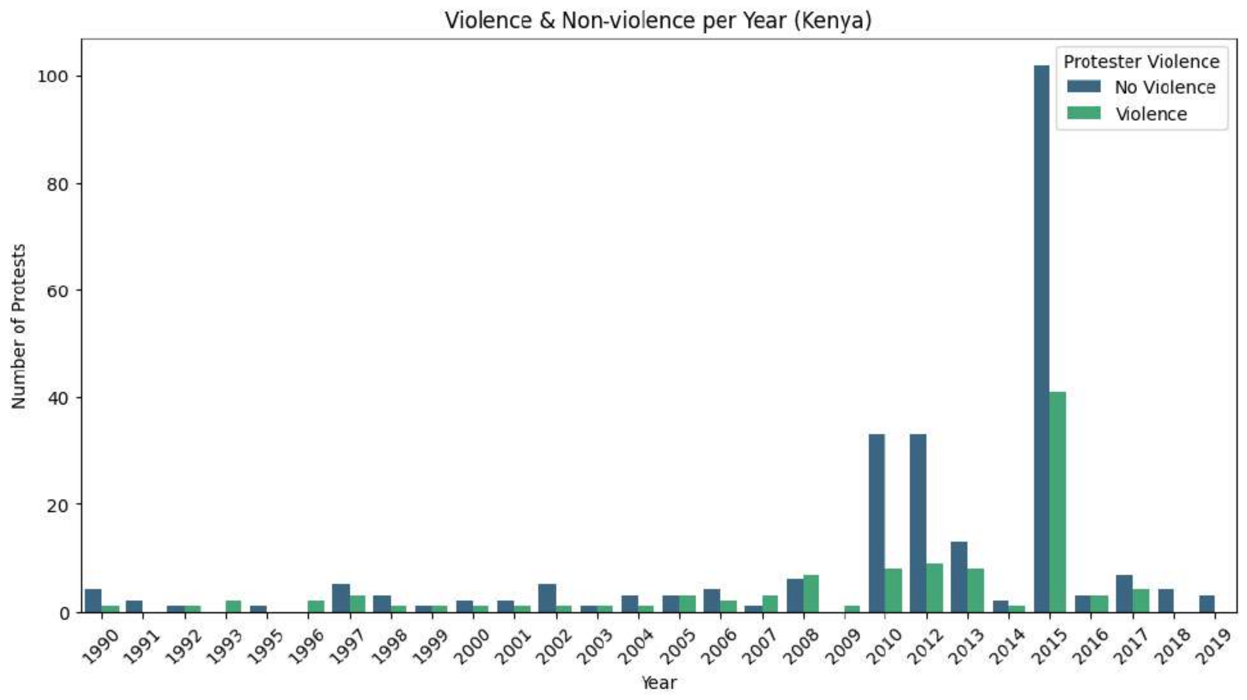
(1e6, means 1,000,000).

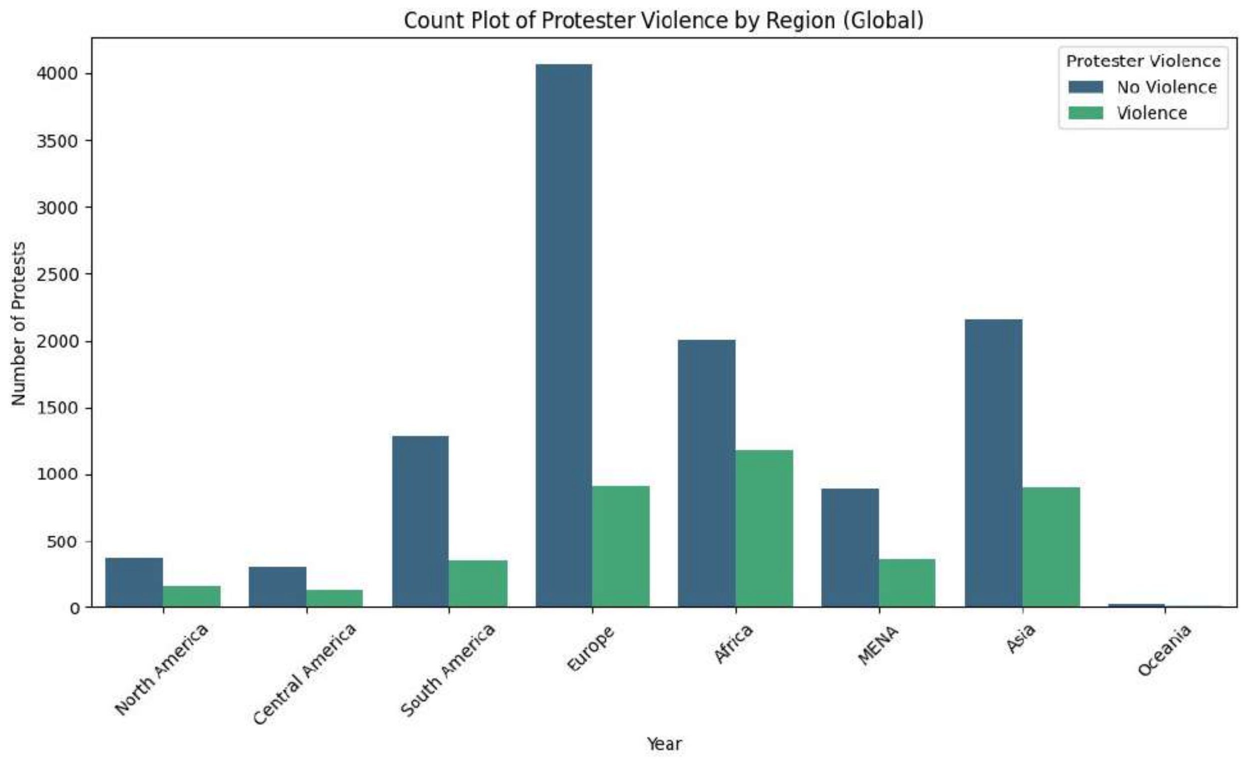
- Regions with higher populations(particularly Asia) recorded a higher number of outliers in the number of protest participants.

CATEGORICAL VS CATEGORICAL

```
# Explore Protester Violence In Kenya
plt.figure(figsize=(12, 6))
sns.countplot(x='protesterviolence', data=kenya_data,
palette="viridis")
plt.title('Count Plot of Protester Violence (Kenya)')
plt.xlabel('0=no 1=yes')
plt.ylabel('Number of Protests')
plt.legend(title='Protester Violence', labels=['No Violence',
'Violence'])
plt.xticks(rotation=45)
plt.show()

# Explore Protester Violence Trends In Kenya Over Time
plt.figure(figsize=(12, 6))
sns.countplot(x='year', hue='protesterviolence', data=kenya_data,
palette="viridis")
plt.title('Violence & Non-violence per Year (Kenya)')
plt.xlabel('Year')
plt.ylabel('Number of Protests')
plt.legend(title='Protester Violence', labels=['No Violence',
'Violence'])
plt.xticks(rotation=45)
plt.show()

# Explore Protester Violence Across Regions
plt.figure(figsize=(12, 6))
sns.countplot(x='region', hue='protesterviolence', data=data,
palette="viridis")
plt.title('Count Plot of Protester Violence by Region (Global)')
plt.xlabel('Year')
plt.ylabel('Number of Protests')
plt.legend(title='Protester Violence', labels=['No Violence',
'Violence'])
plt.xticks(rotation=45)
plt.show()
```

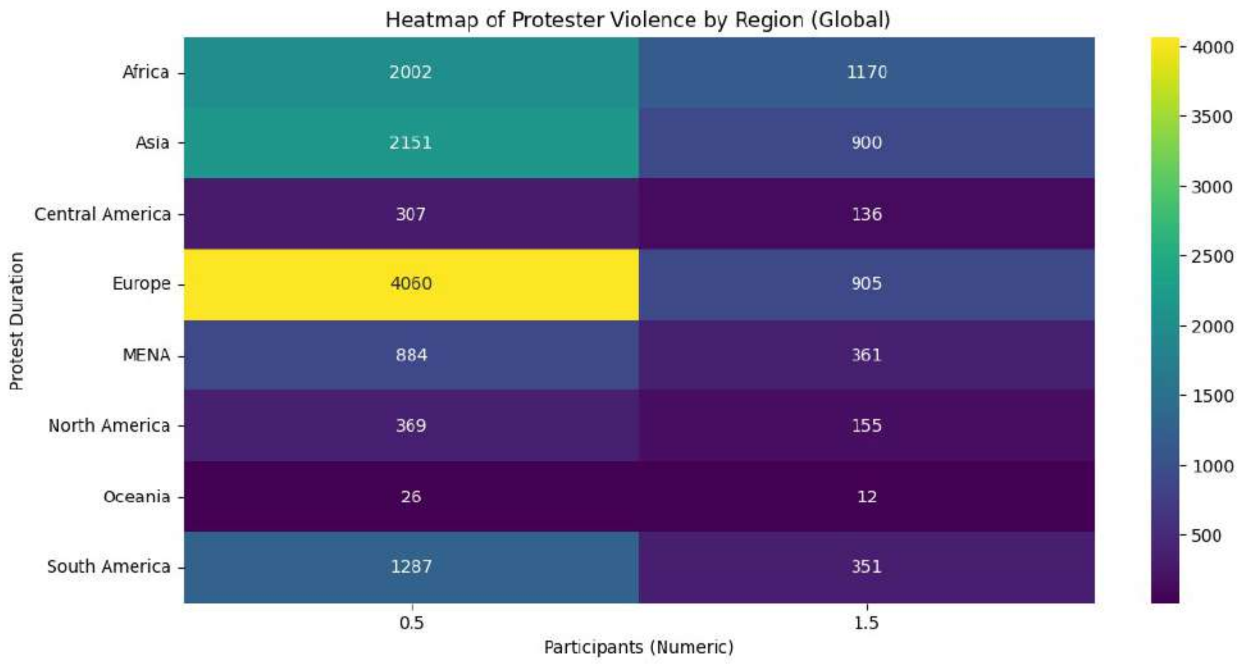

Count Plot of Protester Violence (Kenya)
Protester Violence
No Violence
Violence
Number of Protests
250
200
150
100
50
0
0
1
0=no 1=yes


Violence & Non-violence per Year (Kenya)
Protester Violence
No Violence
Violence
Number of Protests
100
80
60
40
20
0
1990
1991
1992
1993
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2012
2013
2014
2015
2016
2017
2018
2019
Year



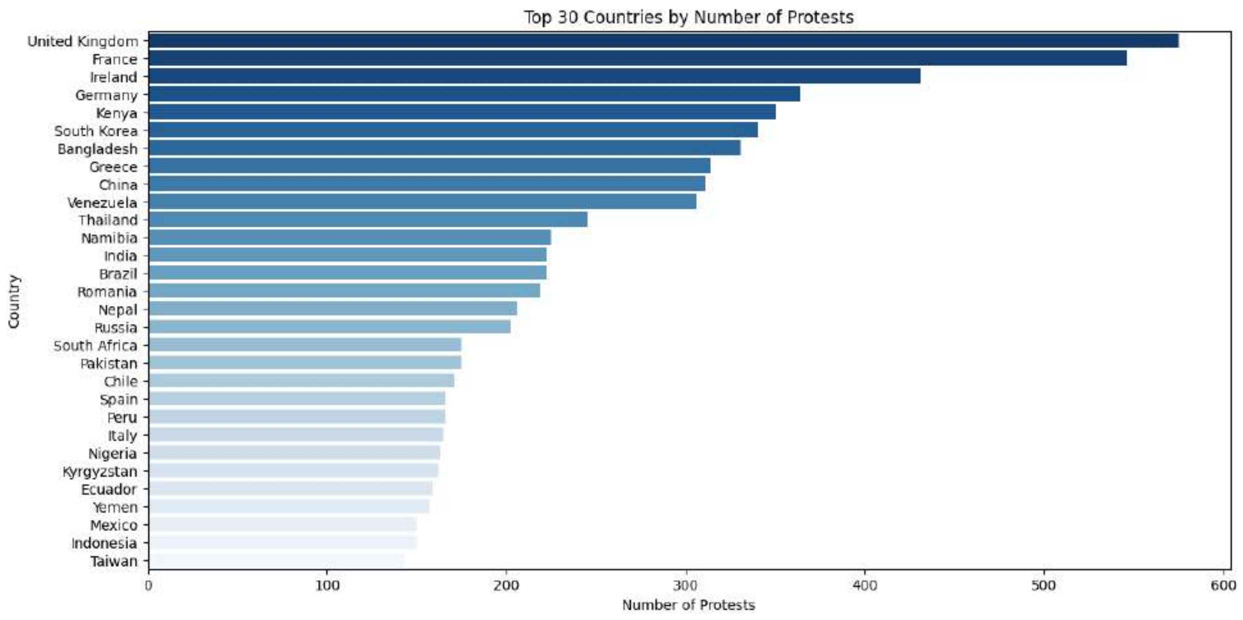
- In Kenya violent protests account for about 1/3 of the total protests.
- In the years 2007-2009 violent protests were the majority in Kenya. This was most likely due to the election outcome.
- Globally most protests are non-violent.

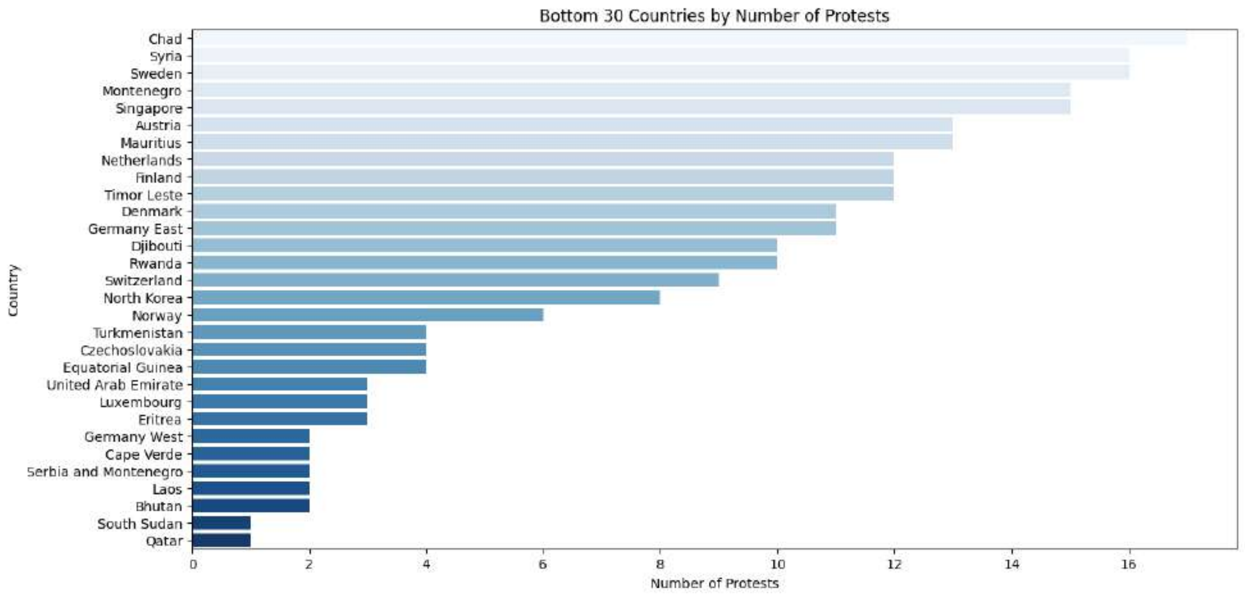
```
# Global - Heatmap of Protester Violence by Region
global_pivot = data.pivot_table(index='region',
columns='protesterviolence', aggfunc='size', fill_value=0)
# Create the plot
plt.figure(figsize=(12, 6))
sns.heatmap(global_pivot, annot=True, cmap='viridis', fmt='d')
plt.title('Heatmap of Protester Violence by Region (Global)')
plt.xlabel('Participants (Numeric)')
plt.ylabel('Protest Duration')
plt.gca().xaxis.set_major_formatter(ScalarFormatter()) # Remove
scientific notation from x-axis
plt.show()
```



- In Africa violent protests account for over 1/3 of total protests contrasted to Europe's 1/5.

```
# Top 30
top_30_countries = data['country'].value_counts().head(30)
plt.figure(figsize=(14, 7))
sns.barplot(y=top_30_countries.index, x=top_30_countries.values,
palette="Blues_r")
plt.title('Top 30 Countries by Number of Protests')
plt.xlabel('Number of Protests')
plt.ylabel('Country')
plt.show()

# Bottom 30
bottom_30_countries = data['country'].value_counts().tail(30)
plt.figure(figsize=(14, 7))
sns.barplot(y=bottom_30_countries.index, x=bottom_30_countries.values,
palette="Blues")
plt.title('Bottom 30 Countries by Number of Protests')
plt.xlabel('Number of Protests')
plt.ylabel('Country')
plt.show()
```





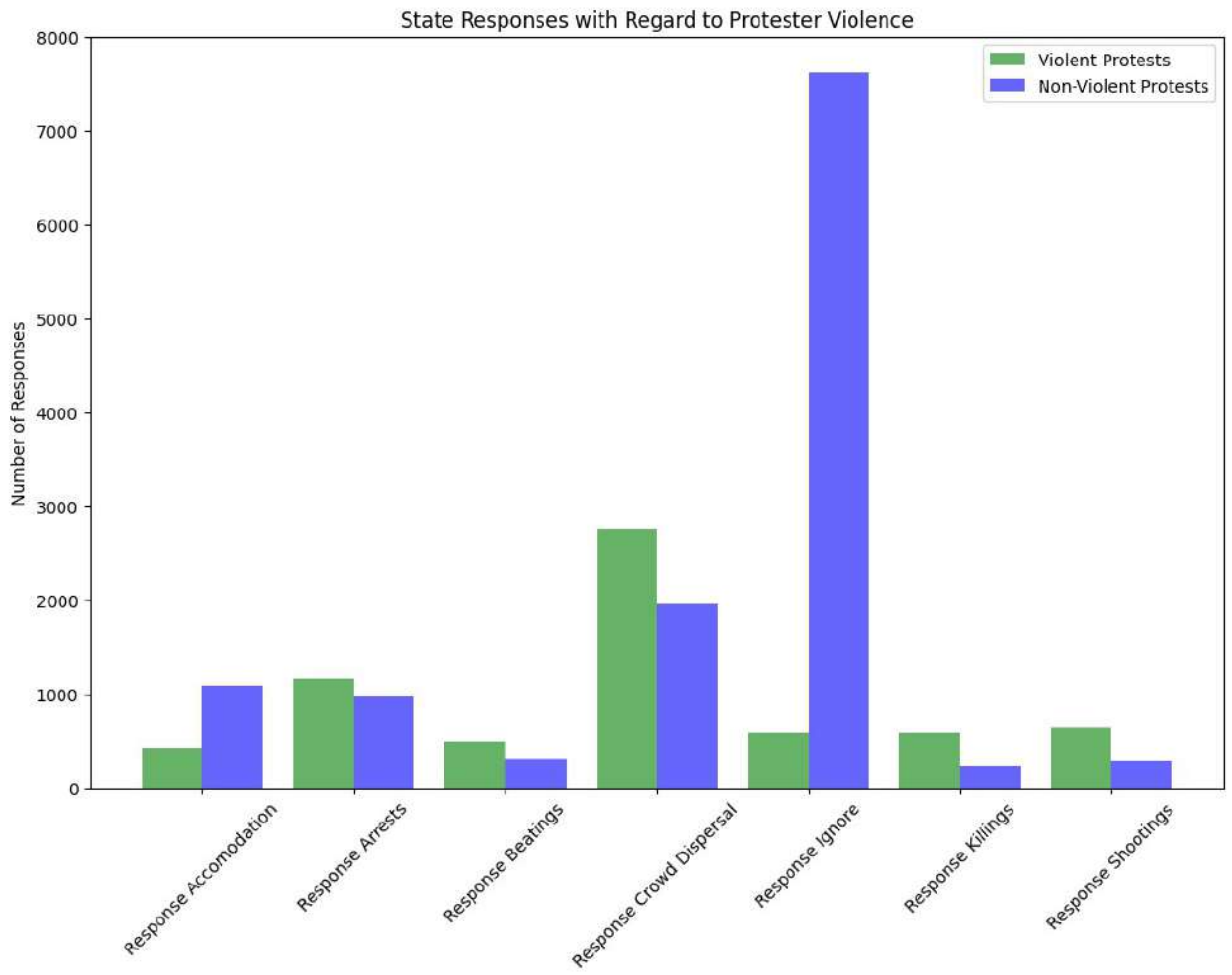
The above graph shows the top & bottom 30 countries by number of protests in the dataset.

- United Kingdom and France had over 500 protests.
- While some like Qatar have just one.

RESPONSES

```
# State responses vs. protester violence
responses = ['response_accomodation', 'response_arrests',
'response_beatings',
             'response_crowd dispersal', 'response_ignore',
'response_killings', 'response_shootings']
```

```python
# Summing up the responses for violent and non-violent protests
response_violence = data[data['protesterviolence'] == 1]
[responses].sum()
response_nonviolence = data[data['protesterviolence'] == 0]
[responses].sum()

# Plotting side-by-side bars
fig, ax = plt.subplots(figsize=(12, 8))
width = 0.4  # width of the bars
ind = np.arange(len(responses))  # the x locations for the groups

ax.bar(ind - width/2, response_violence, width, label='Violent
Protests', color='green', alpha=0.6)
ax.bar(ind + width/2, response_nonviolence, width, label='Non-Violent
Protests', color='blue', alpha=0.6)
ax.set_title('State Responses with Regard to Protester Violence')
ax.set_ylabel('Number of Responses')
ax.set_xticks(ind)
ax.set_xticklabels([response.replace('_', ' ').title() for response in
responses], rotation=45)
ax.legend()

plt.show()
```



The bar graph presents a comparison of how states respond to violent and non-violent protests across various response categories.

Key Findings:

Disproportionate Response to Violent Protests: The most striking observation is the significantly higher number of responses related to violent protests compared to non-violent ones. This indicates a more robust and often harsher state reaction to violent demonstrations.

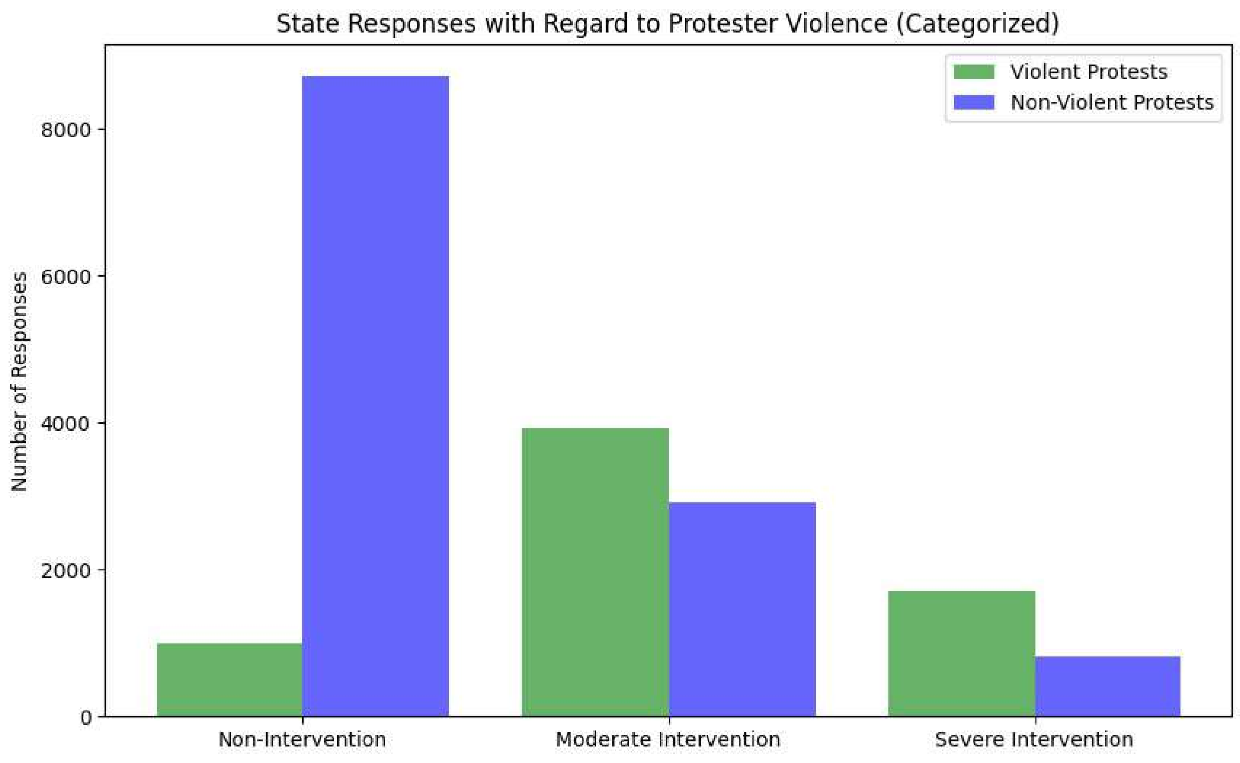
Dominance of Crowd Dispersal: Across both protest types, "Crowd Dispersal" is the most frequent response, suggesting it's a common tactic employed by authorities to manage protests.

Arrests and Beatings: These responses are prevalent in both categories, though they seem to be more pronounced in cases of violent protests. Extreme Measures: While less frequent, responses like "Killings" and "Shootings" appear to be exclusively associated with violent protests, highlighting the severe consequences that can arise from such demonstrations.

Accommodation and Ignore: These responses are minimal in both categories, indicating that they are less common strategies for handling protests.

Overall, the graph reveals a pattern of more aggressive state responses to violent protests, with crowd dispersal being the primary tactic, followed by arrests and beatings.

```
# Categorized state responses
non_intervention = ['response_accomodation', 'response_ignore']
moderate_intervention = ['response_arrests', 'response_crowd
dispersal']
severe_intervention = ['response_beatings', 'response_killings',
'response_shootings']

# Summing up the responses for each category
response_violence = {
    'Non-Intervention': data[data['protesterviolence'] == 1]
[non_intervention].sum().sum(),
    'Moderate Intervention': data[data['protesterviolence'] == 1]
[moderate_intervention].sum().sum(),
    'Severe Intervention': data[data['protesterviolence'] == 1]
[severe_intervention].sum().sum()
}
response_nonviolence = {
    'Non-Intervention': data[data['protesterviolence'] == 0]
[non_intervention].sum().sum(),
    'Moderate Intervention': data[data['protesterviolence'] == 0]
[moderate_intervention].sum().sum(),
    'Severe Intervention': data[data['protesterviolence'] == 0]
[severe_intervention].sum().sum()
}

# Plotting side-by-side bars
fig, ax = plt.subplots(figsize=(10, 6))
width = 0.4  # width of the bars
ind = np.arange(len(response_violence))  # the x locations for the
groups

ax.bar(ind - width/2, list(response_violence.values()), width,
label='Violent Protests', color='green', alpha=0.6)
ax.bar(ind + width/2, list(response_nonviolence.values()), width,
label='Non-Violent Protests', color='blue', alpha=0.6)
ax.set_title('State Responses with Regard to Protester Violence
(Categorized)')
ax.set_ylabel('Number of Responses')
ax.set_xticks(ind)
ax.set_xticklabels(list(response_violence.keys()))
ax.legend()

plt.show()
```



The bar graph illustrates the frequency of different state responses to both violent and non-violent protests.

Key Findings:

Disproportionate Response to Violent Protests: The most striking observation is that state responses are significantly higher for violent protests across all categories. This indicates a more forceful and frequent state reaction to violent demonstrations.

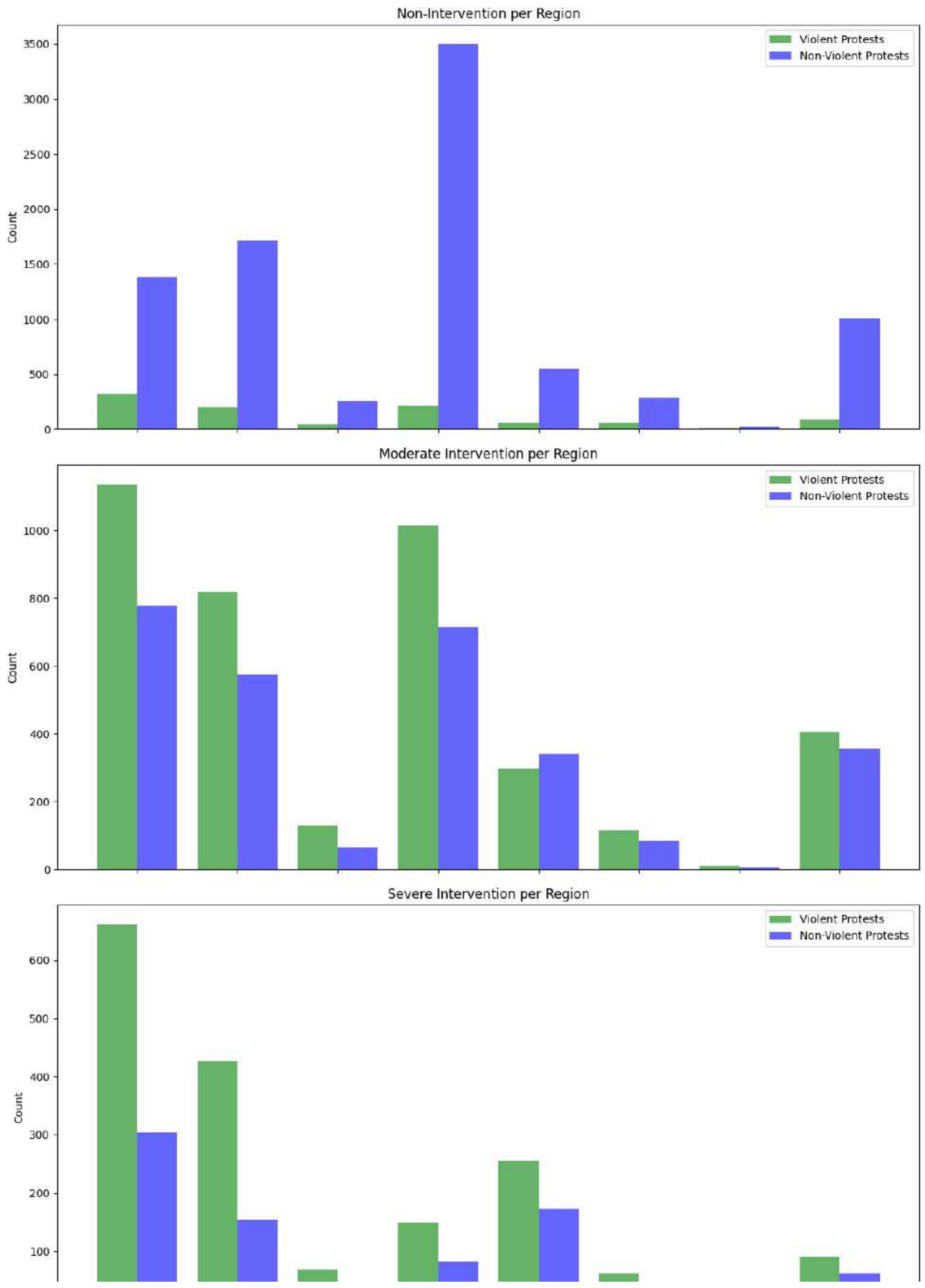
Dominance of Moderate Intervention: While "Moderate Intervention" is the most frequent response for both protest types, it is notably higher for violent protests.

Non-Intervention and Severe Intervention: The use of "Non-Intervention" is minimal in both cases, suggesting it's a rare strategy. Conversely, "Severe Intervention" is primarily used in response to violent protests.

Overall, the graph demonstrates a clear pattern of escalated state responses to violent protests compared to non-violent ones.

```python
# Violence & non-violence outcomes per region
violence_outcomes_region = {
    'Non-Intervention': data[data['protesterviolence'] ==
1].groupby('region')[non_intervention].sum().sum(axis=1),
    'Moderate Intervention': data[data['protesterviolence'] ==
1].groupby('region')[moderate_intervention].sum().sum(axis=1),
    'Severe Intervention': data[data['protesterviolence'] ==
1].groupby('region')[severe_intervention].sum().sum(axis=1)
}
```

```
nonviolence_outcomes_region = {
    'Non-Intervention': data[data['protesterviolence'] ==
0].groupby('region')[non_intervention].sum().sum(axis=1),
    'Moderate Intervention': data[data['protesterviolence'] ==
0].groupby('region')[moderate_intervention].sum().sum(axis=1),
    'Severe Intervention': data[data['protesterviolence'] ==
0].groupby('region')[severe_intervention].sum().sum(axis=1)
}

# Plotting side-by-side bars for each outcome
fig, axes = plt.subplots(3, 1, figsize=(12, 18), sharex=True)
width = 0.4  # width of the bars
categories = ['Non-Intervention', 'Moderate Intervention', 'Severe
Intervention']

for i, category in enumerate(categories):
    ax = axes[i]
    ind = np.arange(len(violence_outcomes_region[category]))  # the x
locations for the groups

    ax.bar(ind - width/2, violence_outcomes_region[category], width,
label='Violent Protests', color='green', alpha=0.6)
    ax.bar(ind + width/2, nonviolence_outcomes_region[category],
width, label='Non-Violent Protests', color='blue', alpha=0.6)
    ax.set_title(f'{category} per Region')
    ax.set_ylabel('Count')
    ax.set_xticks(ind)
    ax.set_xticklabels(violence_outcomes_region[category].index,
rotation=45)
    ax.legend()

plt.tight_layout()
plt.show()
```


Non-Intervention per Region
Violent Protests
Non-Violent Protests
Count
0
500
1000
1500
2000
2500
3000
3500
Moderate Intervention per Region
Violent Protests
Non-Violent Protests
Count
0
200
400
600
800
1000
Severe Intervention per Region
Violent Protests
Non-Violent Protests
Count
100
200
300
400
500
600

The graph presents a comparison of state responses to violent and non-violent protests across different regions of the world. It categorizes responses into three levels: non-intervention, moderate intervention, and severe intervention.

Key Findings

Disproportionate Response to Violent Protests: Across all regions and response levels, there is a significantly higher frequency of state interventions for violent protests compared to non-violent ones. This indicates a more forceful and often harsher state reaction to violent demonstrations.

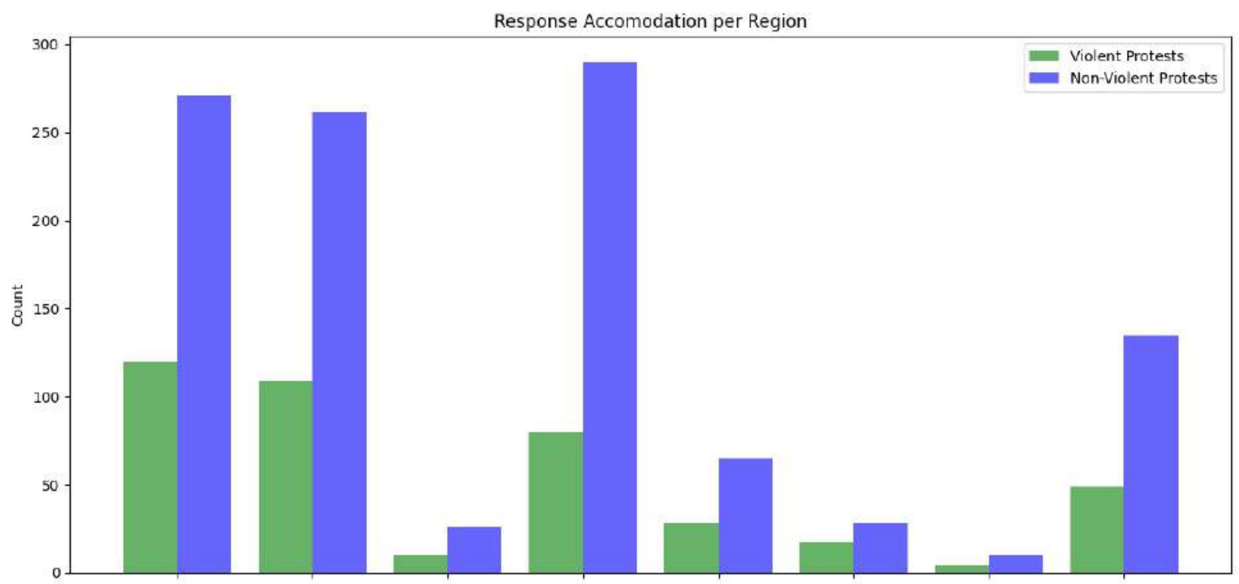
Regional Variations: While the general pattern of more aggressive responses to violent protests holds true across regions, there are noticeable differences in the magnitude of these responses. For instance, some regions exhibit a higher frequency of severe interventions for both protest types compared to others.

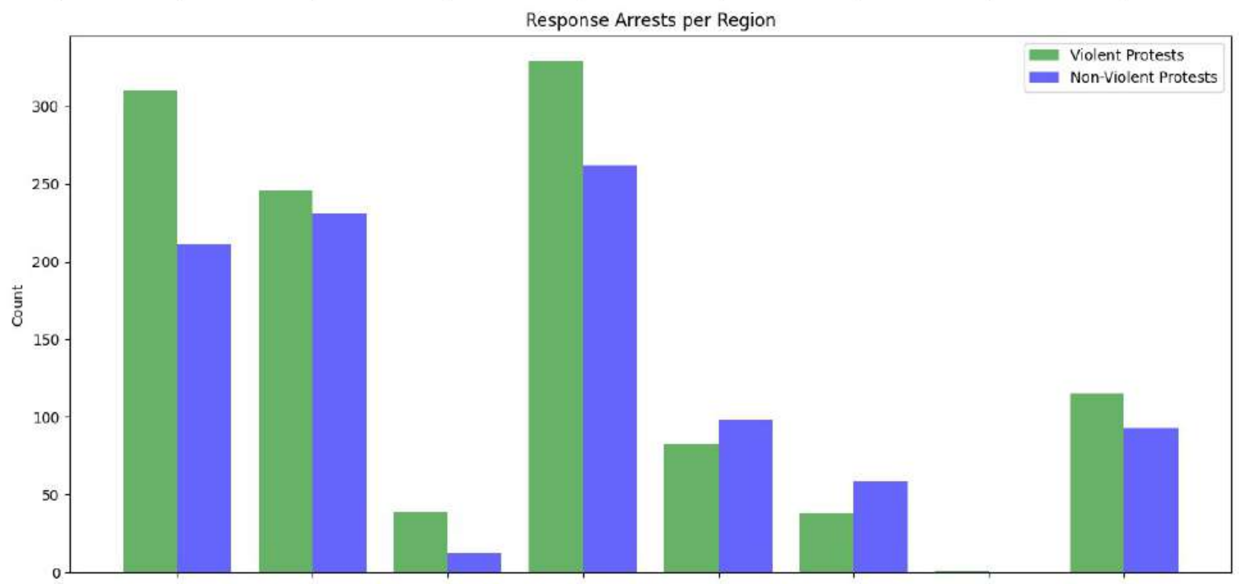
Dominance of Moderate Intervention: Moderate intervention is the most common response category across all regions and protest types, suggesting it's a widely used strategy for managing protests.

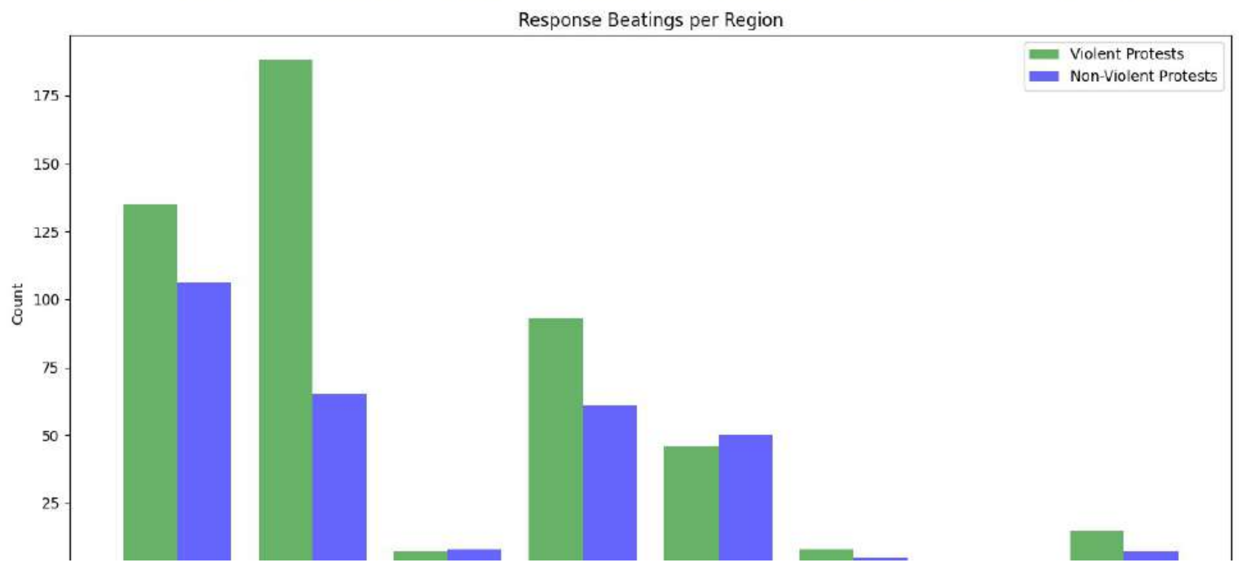
Non-Intervention and Severe Intervention: Non-intervention is relatively rare across all regions and protest types. In contrast, severe intervention is primarily associated with violent protests, particularly in certain regions.

Potential Implications The graph suggests that the global response to protests is characterized by a strong bias towards more forceful measures when faced with violence. This could have significant implications for civil liberties, human rights, and the overall political climate in different regions.

```python
# Violence & non-violence outcomes per region
outcomes = responses.copy()
violence_outcomes_region = data[data['protesterviolence'] ==
1].groupby('region')[outcomes].sum()
nonviolence_outcomes_region = data[data['protesterviolence'] ==
0].groupby('region')[outcomes].sum()

# Plotting side-by-side bars for each outcome
fig, axes = plt.subplots(len(outcomes), 1, figsize=(12, 40),
sharex=True)
width = 0.4  # width of the bars

for i, outcome in enumerate(outcomes):
    ax = axes[i]
    ind = np.arange(len(violence_outcomes_region))  # the x locations
for the groups

    ax.bar(ind - width/2, violence_outcomes_region[outcome], width,
label='Violent Protests', color='green', alpha=0.6)
    ax.bar(ind + width/2, nonviolence_outcomes_region[outcome], width,
label='Non-Violent Protests', color='blue', alpha=0.6)
    ax.set_title(f'{outcome.replace("_", " ").title()} per Region')
    ax.set_ylabel('Count')
    ax.set_xticks(ind)
    ax.set_xticklabels(violence_outcomes_region.index, rotation=45)
    ax.legend()

plt.tight_layout()
plt.show()
```

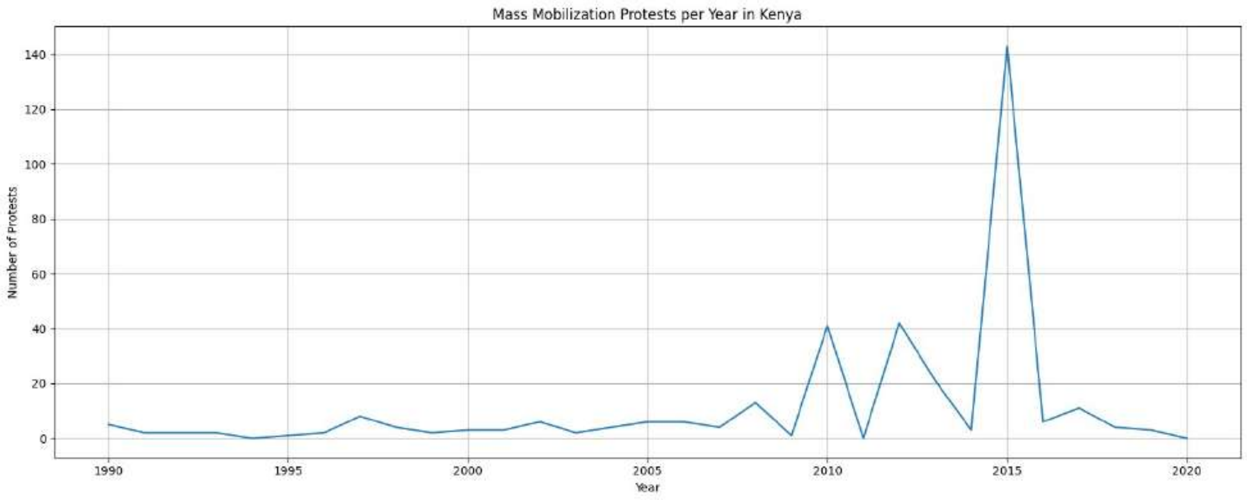

Response Accomodation per Region
Count
300
250
200
150
100
50
0
Violent Protests
Non-Violent Protests

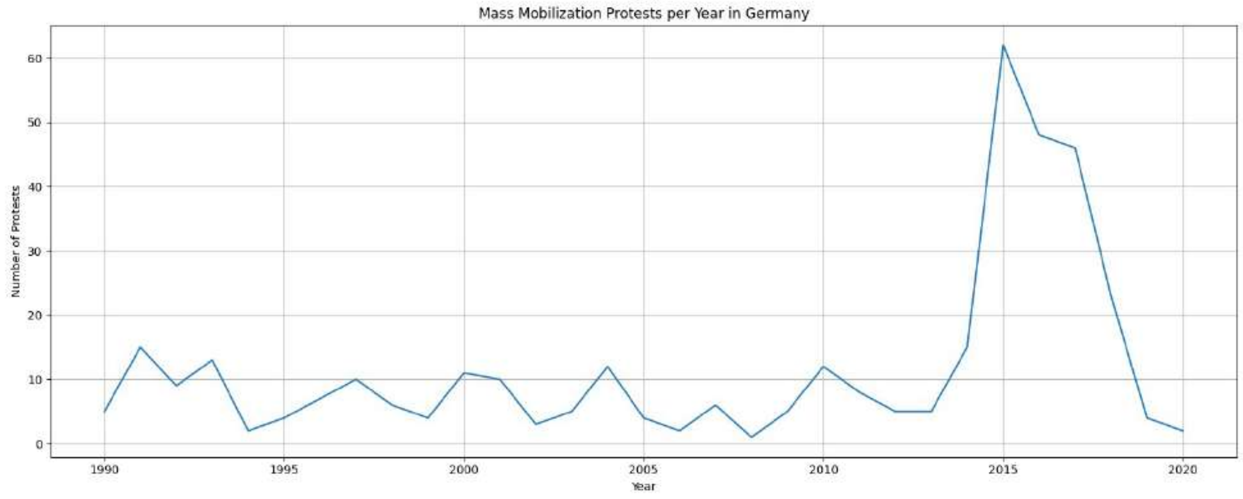

Response Arrests per Region
Count
300
250
200
150
100
50
0
Violent Protests
Non-Violent Protests

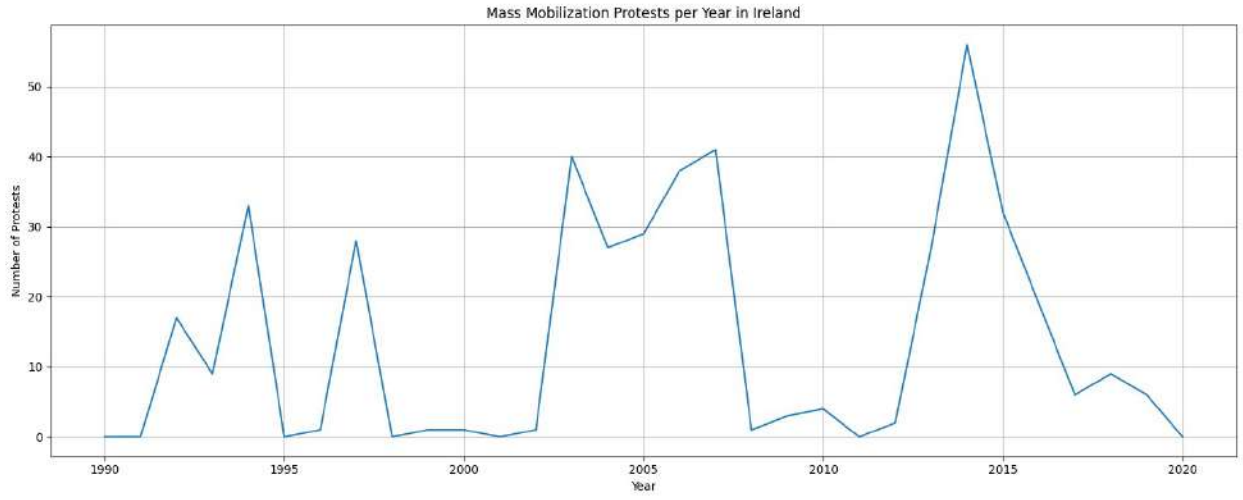

Response Beatings per Region
Count
175
150
125
100
75
50
25
Violent Protests
Non-Violent Protests

- For non-violent protests ignored is the favored response while crowd dispersal is preferred for violent outcomes.
- Violent outcomes are most often met mith more severe forms of intervention.
- In AFrica severe interevention(specifically shootings) as a response to violent protests is more than double the next region.

```
# Select desired countries
countries = ["Kenya", 'Germany', 'Ireland']

# Filter data for these countries
filtered_data = data[data['country'].isin(countries)]

# Group data by country and year, then count protests per year for
each country
protest_counts = filtered_data.groupby(['country',
'year']).size().unstack(fill_value=0)

# Create subplots for each country
fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(15, 18))  # Adjust
the figsize as needed

for i, country in enumerate(countries):
    protest_counts.T[country].plot(kind='line', ax=axes[i])
    axes[i].set_xlabel("Year")
    axes[i].set_ylabel("Number of Protests")
    axes[i].set_title(f"Mass Mobilization Protests per Year in
{country}")
    axes[i].grid(True)

plt.tight_layout()
plt.show()
```







- Kenya: In 2015, Kenya documented 140 protests, sparked by a wide range of issues including lack of adequate security, corruption, land-grabbing, unemployment, political reform, and poor road conditions. Kenya
- Germany: In 2015 there was an increase in protests regarding immigration and the circulation of the white nationalist conspiracy theory of the Great Replacement. Germany

- Ireland: The anti-austerity movement in Ireland saw major demonstrations from 2008 (the year of the Irish economic downturn) to 2015. Ireland

MULITIVARIATE ANALYSIS

Multivariate analysis refers to statistical techniques used to analyze data that involves multiple variables simultaneously. The goal is to understand the relationships between more than two variables and to identify patterns or structures in the data.

```
# Check the columns
data.columns
```

```
Index(['region', 'country', 'year', 'start_date', 'end_date',
       'protest_duration', 'participants_numeric',
'protesterviolence',
       'protesteridentity', 'demand_labor wage dispute',
       'demand_land farm issue', 'demand_police brutality',
       'demand_political behavior', 'demand_price increases',
'demand_process',
       'demand_removal of politician', 'demand_social restrictions',
       'demand_tax policy', 'response_accomodation',
'response_arrests',
       'response_beatings', 'response_crowd dispersal',
'response_ignore',
       'response_killings', 'response_shootings', 'sources', 'notes'],
      dtype='object')
```

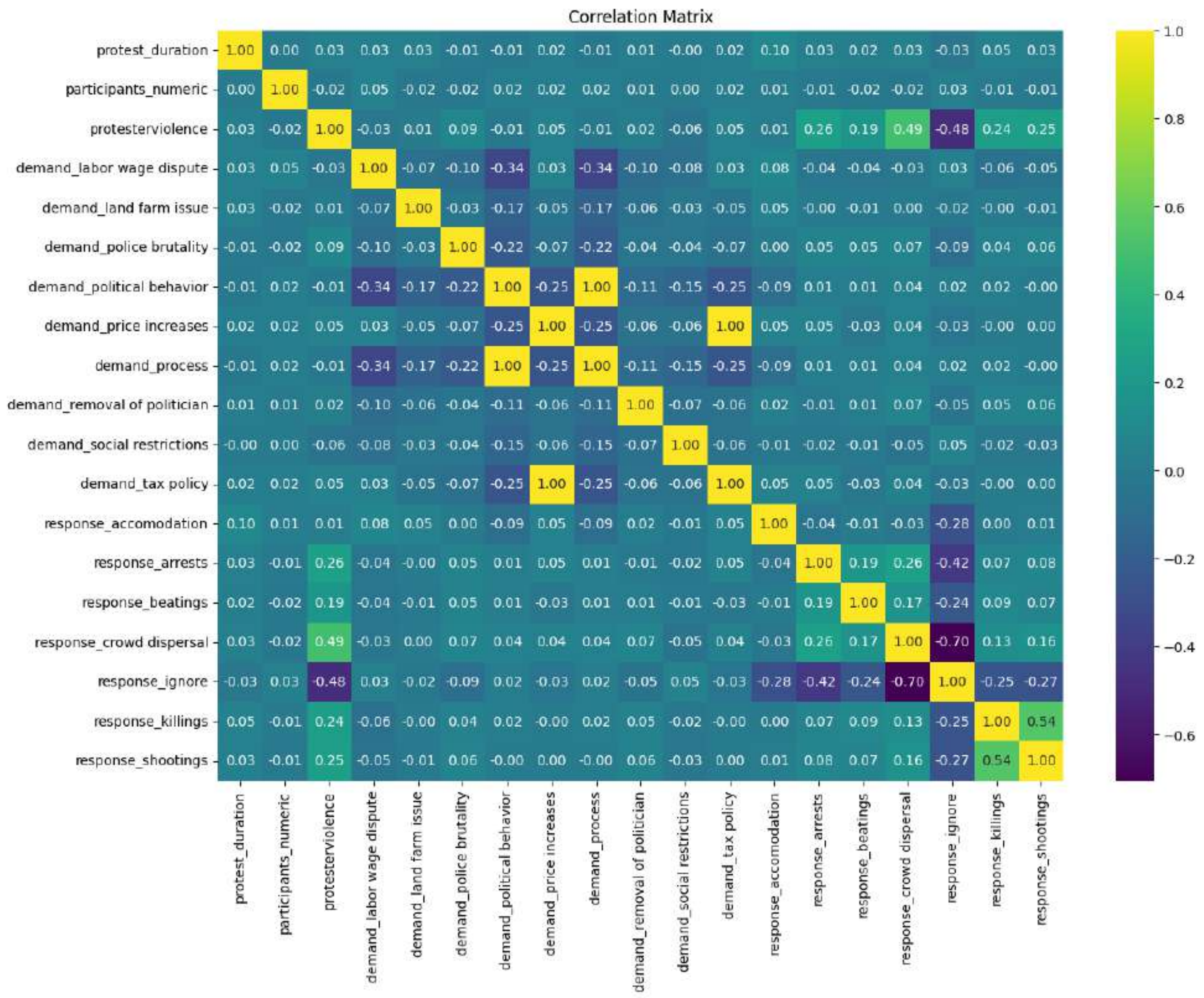
```
# Check the data types of the columns
data.dtypes
```

```
region                                  object
country                                 object
year                                     int64
start_date                      datetime64[ns]
end_date                        datetime64[ns]
protest_duration                         int64
participants_numeric                     int64
protesterviolence                        int64
protesteridentity                       object
demand_labor wage dispute                int64
demand_land farm issue                   int64
demand_police brutality                  int64
demand_political behavior                int64
demand_price increases                   int64
demand_process                           int64
demand_removal of politician             int64
demand_social restrictions               int64
demand_tax policy                        int64
response_accomodation                    int64
response_arrests                         int64
```

```
response_beatings                                         int64
response_crowd dispersal                                  int64
response_ignore                                           int64
response_killings                                         int64
response_shootings                                        int64
sources                                                  object
notes                                                    object
dtype: object
```

CORRELATION MATRIX

```
# Define features of interest
features_of_interest = ['protest_duration', 'participants_numeric',
'protesterviolence',
                        'demand_labor wage dispute', 'demand_land farm
issue', 'demand_police brutality',
                        'demand_political behavior', 'demand_price
increases', 'demand_process',
                        'demand_removal of politician', 'demand_social
restrictions', 'demand_tax policy',
                        'response_accomodation', 'response_arrests',
'response_beatings',
                        'response_crowd dispersal', 'response_ignore',
'response_killings', 'response_shootings']

# Correlation Heatmap
plt.figure(figsize=(14, 10))
correlation_matrix = data[features_of_interest].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='viridis', fmt='.2f')
# Ensure annotations are displayed with two decimal places
plt.title('Correlation Matrix')
plt.show()
```



Key Observations:

Strong Correlations:

response_killings and response_shootings have a high positive correlation (0.54), suggesting that protests involving killings often involve shootings as well. response_ignore and response_crowd dispersal have a strong negative correlation (-0.7), indicating that ignoring a protest is typically not accompanied by crowd dispersal tactics.

Moderate Correlations:

participants_numeric and response_crowd dispersal (0.49) show a moderate positive correlation, implying that protests with more participants are more likely to be dispersed by the crowd dispersal response. protesterviolence and response_arrests (0.24) suggest that violent protests have a moderate tendency to result in arrests.

Weak Correlations:

Most demands (e.g., demand_labor wage dispute, demand_land farm issue) show very weak or no significant correlations with other features, indicating that the nature of demands does not strongly predict other responses or characteristics of the protests. protest_duration shows weak correlations with other features, suggesting the duration of protests is not strongly influenced by the other variables in the dataset.

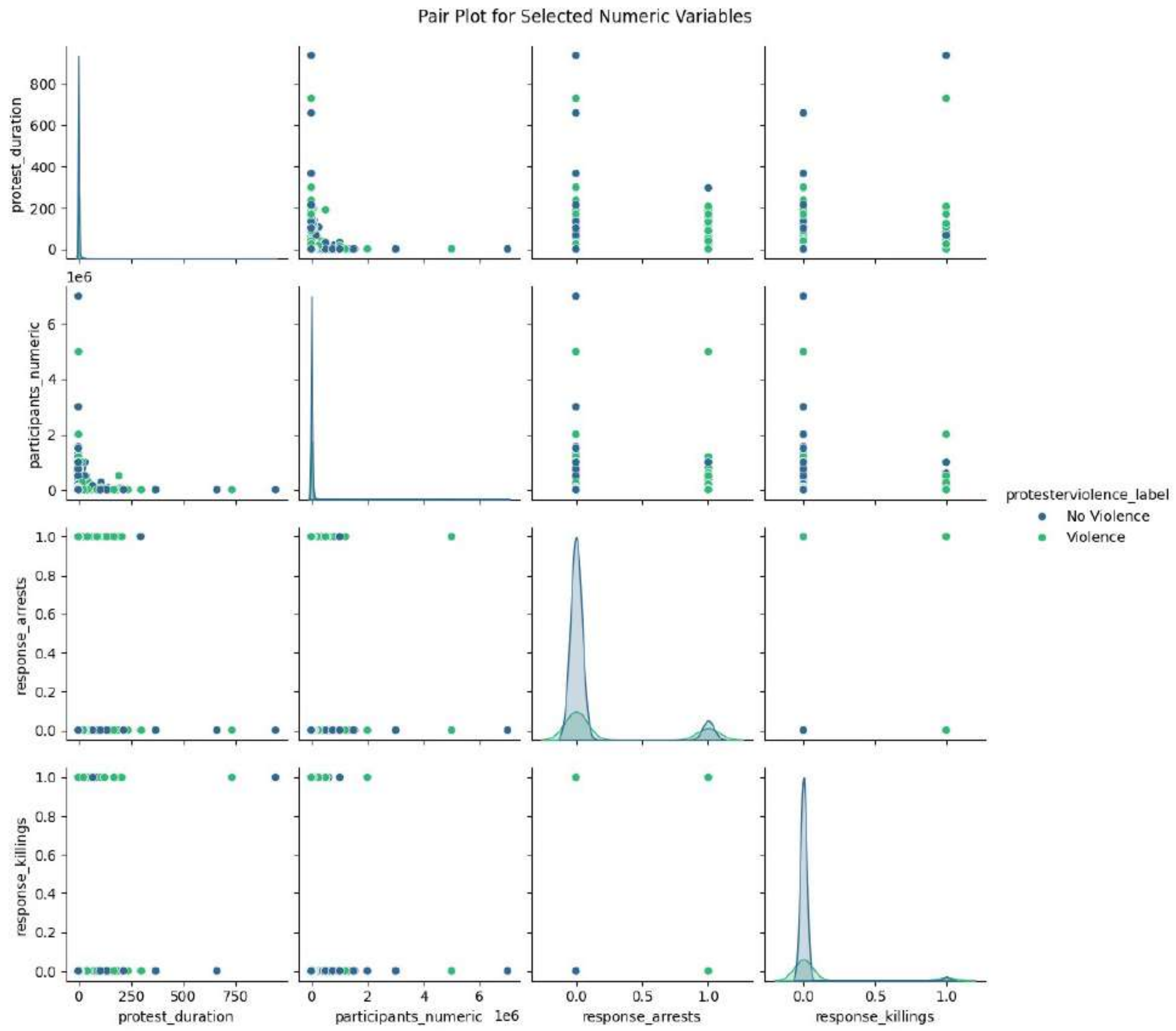
Interesting Negative Correlations:

response_ignore shows notable negative correlations with several responses such as response_arrests (-0.48) and response_crowd dispersal (-0.7), indicating a distinct pattern where protests ignored are less likely to see other active responses.

Conclusion: The correlation matrix reveals the relationships between various aspects of protests, demands, and state responses. Key insights include the co-occurrence of violent responses, the tendency for larger protests to be dispersed, and the mutual exclusivity of ignoring protests and active responses.

PAIRPLOT

```
# Map the protesterviolence variable to 'No Violence' and 'Violence'
data['protesterviolence_label'] = data['protesterviolence'].map({0:
'No Violence', 1: 'Violence'})

# Pair plot for selected numeric variables
selected_columns = ['protest_duration', 'participants_numeric',
'protesterviolence_label',
                    'response_arrests', 'response_killings']

pair_plot = sns.pairplot(data[selected_columns],
hue='protesterviolence_label', diag_kind='kde', palette='viridis')
pair_plot.fig.suptitle("Pair Plot for Selected Numeric Variables",
y=1.02)

plt.show()
```



Key Observations

1.Protest Duration:

a)Distribution: Mostly short with few long durations; similar for violent and non-violent protests. b)Relationships: No strong links to other variables.

2.Participants Numeric:

a)Distribution:Skewed towards fewer participants; few with very high numbers. b)Relationships:

```
   i)Larger protests don't last longer.
   ii)Mixed responses in arrests and killings.
```

3.Protester Violence:

a)Distribution:Evenly spread. b)Relationships:

```
    i)Wide range in participant numbers.
    ii)Higher instances of arrests and killings in violent protests.
```

4.Response Arrests:

a)Distribution: Many protests see no arrests; some see significant arrests. b)Relationships: i)No clear link to protest duration. ii)No strong correlation with the number of participants.

5.Response Killings:

a)Distribution: Rare. b)Relationships: i)No strong link to protest duration. ii)Varying participant numbers.

Insights

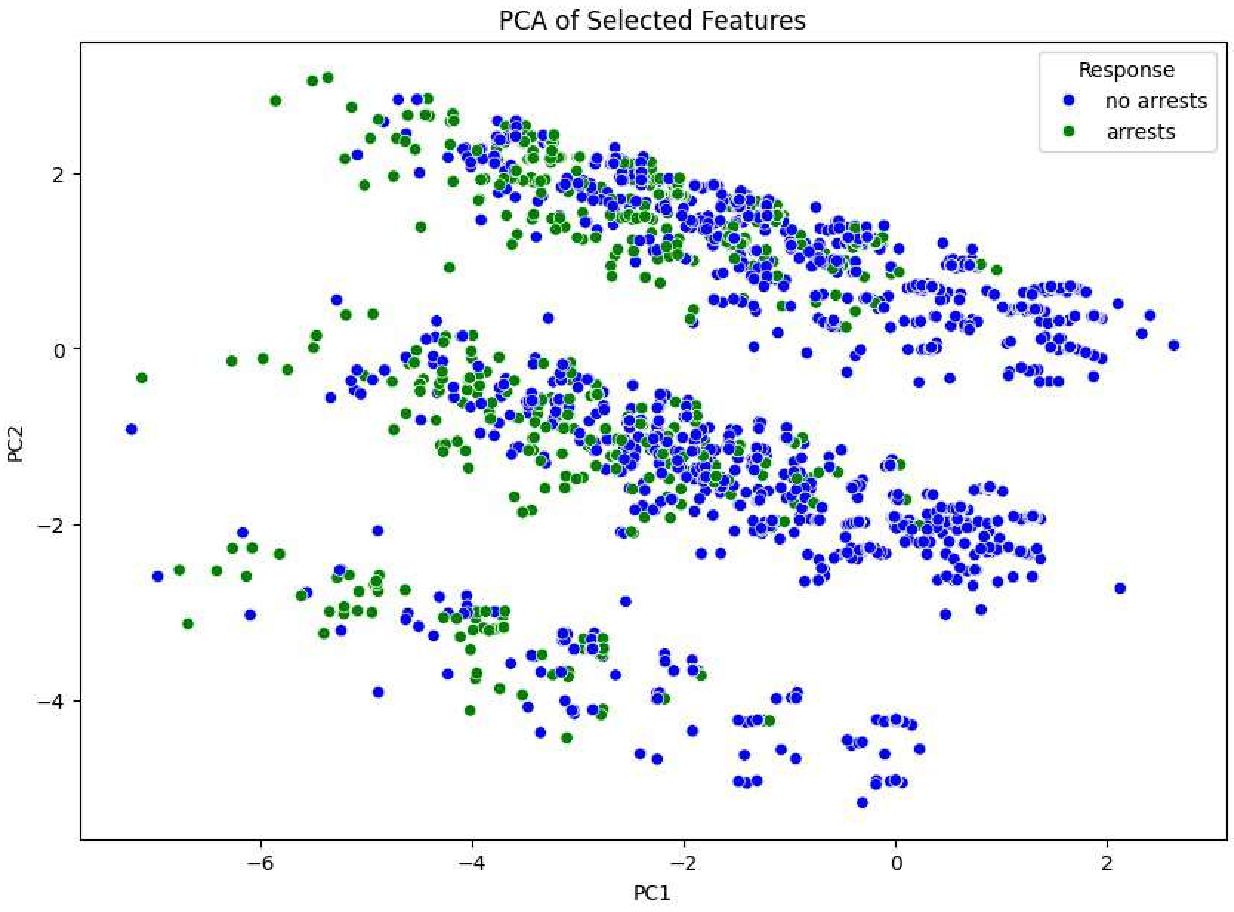
Violence and Response: Violent protests lead to more arrests and killings.

Protest Size and Response: Number of participants doesn't predict duration, arrests, or killings.

PCA

```
numeric_features = [feature for feature in features_of_interest if
data[feature].dtype in ['int64', 'float64']]
# Filter numeric columns from the original data
data_numeric = data[numeric_features]

# Scaling
scaler = StandardScaler()
scaled_features = scaler.fit_transform(data_numeric)

# PCA
pca = PCA(n_components=2)
principal_components = pca.fit_transform(scaled_features)
df_pca = pd.DataFrame(data=principal_components, columns=['PC1',
'PC2'])

# Map numerical values to labels
label_mapping = {0: 'no arrests', 1: 'arrests'}
data['response_arrests_label'] =
data['response_arrests'].map(label_mapping)

# Ensure the label column is included in df_pca
df_pca = pd.concat([df_pca,
data[['response_arrests_label']].reset_index(drop=True)], axis=1)

# Define a custom palette
palette = {'arrests': 'green', 'no arrests': 'blue'}

# Plotting
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PC1', y='PC2', data=df_pca,
hue='response_arrests_label', palette=palette)
```

```
plt.title('PCA of Selected Features')
plt.legend(title='Response')
plt.show()
```



Key Observations:

Distribution of Protests:

The protests are distributed across several clusters in the PCA space, indicating distinct groupings based on the selected features. The green and blue points are intermixed, suggesting that the response of arrests is not strongly separated in the PCA space.

Clusters:

Several horizontal bands can be seen, indicating that protests with similar characteristics (as captured by the PCA) tend to cluster together. The green points (arrests) are scattered within these bands, indicating that arrests can occur across different types of protests.

Arrests and Protest Characteristics:

The lack of clear separation between protests that led to arrests and those that did not suggests that arrests may be influenced by a combination of factors rather than a single identifiable characteristic.

Conclusion: The PCA analysis provides a visual summary of how protests are distributed based on the selected features. The intermixing of arrests and non-arrests suggests that predicting arrests may require considering multiple dimensions and interactions between features. This insight can guide further analysis and modeling efforts to better understand the factors leading to different state responses.

```
# Calculate the explained variance by the PCA components
explained_variance = pca.explained_variance_ratio_

# Get the loading scores (importance of each feature to the principal
components)
loading_scores = pca.components_.T

# Create a DataFrame for loading scores
loading_scores_df = pd.DataFrame(loading_scores, columns=['PC1',
'PC2'], index=features_of_interest)

# Display the explained variance and loading scores
explained_variance, loading_scores_df
```

```
(array([0.14617207, 0.13796478]),
                                     PC1       PC2
 protest_duration              -0.052806 -0.014034
 participants_numeric           0.023665 -0.010538
 protesterviolence             -0.415387  0.106611
 demand_labor wage dispute     -0.001408 -0.243195
 demand_land farm issue        -0.020340 -0.075895
 demand_police brutality       -0.118789 -0.060883
 demand_political behavior      0.132091  0.518851
 demand_price increases        -0.155026 -0.384836
 demand_process                 0.132091  0.518851
 demand_removal of politician -0.059706 -0.001337
 demand_social restrictions     0.046408 -0.070048
 demand_tax policy             -0.155026 -0.384836
 response_accomodation         -0.094002 -0.076888
 response_arrests              -0.294110  0.086233
 response_beatings             -0.207776  0.094328
 response_crowd dispersal      -0.428458  0.142861
 response_ignore                0.495455 -0.124573
 response_killings             -0.269050  0.111183
 response_shootings            -0.284516  0.098299)
```

Explained Variance

The explained variance ratio tells us how much of the total variance in the dataset is captured by each principal component:

PC1: 14.62% PC2: 13.80% Together, the first two principal components explain approximately 28.42% of the variance in the dataset. This means that nearly one-third of the variability in the data can be represented in a two-dimensional plot, which is helpful for visualization but may not capture all the underlying complexity.

Loading Scores and Influential Features

The loading scores indicate the contribution of each feature to the principal components (PC1 and PC2). Here are some key observations:

Principal Component 1 (PC1)

protesterviolence (0.415): This feature has the highest positive contribution to PC1. Protests involving violence by protestors are strongly associated with the first principal component.

response_crowd dispersal (0.429): This feature also has a strong positive contribution. This suggests that instances where authorities dispersed crowds are well captured by PC1.

response_ignore (-0.495): This feature has a high negative contribution, indicating that when authorities ignore protests, it is inversely related to PC1.

Principal Component 2 (PC2)

demand_political behavior (-0.519) and demand_process (-0.519): These features have the most substantial negative contributions to PC2. Protests demanding changes in political behavior and processes are strongly associated with the second principal component.

demand_price increases (0.385) and demand_tax policy (0.385): Both have high positive contributions to PC2. Protests concerning price increases and tax policies are key factors for this component.

demand_labor wage dispute (0.243): Labor and wage dispute demands also contribute positively to PC2.

Interpretation

1.Separation of Protest Characteristics:

PC1 seems to distinguish protests based on the nature of the authorities' responses and the level of protestor violence. High values of PC1 are associated with violent protests and significant crowd dispersal actions by authorities, whereas low values are associated with protests that are largely ignored by authorities.

2.Nature of Demands:

PC2 separates protests based on their demands. High values of PC2 are associated with economic demands (price increases, tax policy, labor disputes), while low values are associated with political behavior and process demands.

Practical Implications

Policy Making: Understanding the main drivers behind different types of protests can help policymakers address the underlying issues more effectively.

Law Enforcement: Recognizing the patterns in violent protests versus ignored protests can inform law enforcement strategies and training.

Activism and Advocacy: Insights into which demands are most prominent in protests can guide advocacy groups in framing their campaigns and understanding potential public and governmental responses.

Conclusion The PCA analysis reveals that protests can be broadly characterized by the nature of the authorities' response and the type of demands made by protestors. These insights can inform more targeted and effective responses from policymakers, law enforcement, and advocacy groups.

RECOMMENDATIONS

1. Enhancing Early Warning Systems:

Recommendation: Implement advanced analytics and real-time monitoring systems to detect potential protest triggers and unrest patterns. Utilize predictive modeling to forecast potential protest hotspots based on historical data, underlying factors, and emerging socio-political issues.

Rationale: Understanding the underlying factors that lead to mass protests can help in creating early warning systems that allow for proactive measures, reducing the likelihood of escalation and minimizing potential impacts.

2. Improving State Responses:

Recommendation: Develop and refine state response strategies based on the nature of the protests, their demands, and the region-specific contexts. This includes training law enforcement in de-escalation techniques and establishing communication channels to engage with protestors constructively.

Rationale: Tailored state responses that consider the context of protests can mitigate conflict, reduce violence, and facilitate peaceful resolutions, thereby maintaining stability and public trust.

3. Policy Reformation and Inclusivity:

Recommendation: Address the root causes of unrest by incorporating findings into policy reforms. Focus on socio-economic inequalities, governance issues, and other identified factors that contribute to protest movements. Engage with civil society and protest leaders to include their perspectives in policy discussions.

Rationale: Proactively addressing the root causes of protests through inclusive policy-making can prevent future unrest and build a more resilient and equitable society.

4. Leveraging Natural Language Processing (NLP):

Recommendation: Utilize NLP tools to continuously monitor and analyze public sentiment, especially on social media, to gauge public opinion and emerging concerns. This can be integrated into a broader social listening strategy for real-time insights into public grievances.

Rationale: Continuous sentiment analysis can help policymakers stay informed about public sentiments and emerging issues, enabling more timely and effective responses to prevent protests from escalating.

5. Global Collaboration and Knowledge Sharing:

Recommendation: Foster international collaboration to share insights, strategies, and best practices for managing protests and social unrest. Create platforms for knowledge exchange between governments, international organizations, and researchers.

Rationale: Protests often have global implications, and shared knowledge and resources can help nations manage their responses more effectively while learning from each other's experiences.

CONCLUSION

The analysis of global protest events from 1990 to 2020 highlights the complexity and importance of understanding the dynamics of socio-political unrest. The insights gained emphasize the need for proactive and context-sensitive approaches to managing protests, addressing their root causes, and fostering positive state-citizen relations. By implementing the recommended strategies, stakeholders can enhance their ability to anticipate, respond to, and ultimately mitigate the impacts of protest movements, contributing to a more stable and just global society.

4.2 CLASSIFICATION AND NATURAL PROCESSING MODELLING

Import Libraries

```
# Data Manipulation and Visualization
import pandas as pd import numpy as
np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.ticker import ScalarFormatter
import re import string import warnings
import joblib
warnings.filterwarnings('ignore')

# Machine Learning Libraries
import sklearn
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler, MinMaxScaler,
OneHotEncoder, LabelEncoder, label_binarize
from sklearn.model_selection import train_test_split, GridSearchCV,
RandomizedSearchCV, cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier,
ExtraTreesClassifier, GradientBoostingClassifier
from sklearn.naive_bayes import BernoulliNB from
sklearn.neighbors import KNeighborsClassifier from
sklearn.tree import DecisionTreeClassifier from
sklearn.pipeline import Pipeline
from sklearn.multiclass import OneVsRestClassifier
from sklearn import metrics
from sklearn.metrics import accuracy_score, classification_report,
roc_curve, roc_auc_score, average_precision_score from
sklearn.metrics import precision_recall_curve, auc,
confusion_matrix, ConfusionMatrixDisplay
#from sklearn.metrics import plot_roc_curve
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
as LDA
from imblearn.over_sampling import SMOTE
from imblearn.pipeline import Pipeline as ImbPipeline from
sklearn.feature_selection import SelectFromModel, RFECV from
statsmodels.stats.outliers_influence import
variance_inflation_factor
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# NLP Libraries
from sklearn.feature_extraction.text import CountVectorizer,
```

```
TfidfVectorizer
```

```python
from sklearn.decomposition import LatentDirichletAllocation
import nltk
from nltk.corpus import stopwords from
nltk.tokenize import word_tokenize from
textblob import TextBlob import spacy
import gensim from gensim import corpora
from gensim.models import CoherenceModel
from nltk.stem import WordNetLemmatizer

# Download the stopwords corpus if you haven't already
nltk.download('wordnet') nltk.download('stopwords')
nltk.download('punkt') nltk.download('omw-1.4')

import spacy
spacy.cli.download("en_core_web_sm")


# Get the list of stopwords for English
stop_words = set(stopwords.words('english'))

# XGBoost
from xgboost import XGBClassifier

#SVC
from sklearn.svm import SVC

# Datetime
import datetime
```

```
[nltk_data] Downloading package wordnet to [nltk_data]
C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to [nltk_data]
C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to [nltk_data]
C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to [nltk_data]
C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!

✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
```

```
If you are in a Jupyter or Colab notebook, you may need to restart
Python in

order to load all the package's dependencies. You can do this by
selecting the

'Restart kernel' or 'Restart runtime' option.

# Load the dataset
file_path = 'mass_mobilization_cleaned.csv'
data = pd.read_csv(file_path)

# Underscore column names
data.columns = [col.replace(' ', '_') if len(col.split()) >= 1 else col
for col in data.columns]

# Display the first few rows
data.head()

          region country  year  start_date    end_date
protest_duration  \
0    North America  Canada  1990  1990-01-15  1990-01-15
0
1    North America  Canada  1990  1990-06-25  1990-06-25
0
2    North America  Canada  1990  1990-07-01  1990-07-01               0
3    North America  Canada  1990  1990-07-12  1990-09-06
56
4    North America  Canada  1990  1990-08-14  1990-08-15
1
   participants_numeric  protesterviolence           protesteridentity


                   5000                  0                 unspecified



\
0

1                  1000                  0                 unspecified

2                  500                  0  separatist parti quebecois

3                  500                  1               mohawk indians

4                  950                  1              local residents
```

```
   demand_labor_wage_dispute  ...  demand_tax_policy                 0
response_accomodation  \                                 0
                                                                     1
                          1  ...                  0  0
                                                                     2
                          0  ...                  0  0

                          0  ...                  0
```

```
3    0  ...                  0
1
4    0  ...                  0                    1
   response_arrests  response_beatings  response_crowd_dispersal  \
0                 0                  0                         0
1                 0                  0                         0
2                 0                  0                         0
3                 0                  0                         0
4                 1                  0                         1

   response_ignore  response_killings  response_shootings  \
0                1                  0
1                1                  0
2                1                  0
3                0                  0
4                0                  0
                                              sources  \
0  1. great canadian train journeys into history;...
1  1. autonomy s cry revived in quebec the new yo...
2  1. quebec protest after queen calls for unity ...
3  1. indians gather as siege intensifies; armed ...
4  1. dozens hurt in mohawk blockade protest the ...

                                                notes
0  canada s railway passenger system was finally ...
1  protestors were only identified as young peopl...
2  the queen, after calling on canadians to remai...
3  canada s federal government has agreed to acqu...
4  protests were directed against the state due t...
[5 rows x 27 columns]
```

4.2.1 DATA PREPROCESSING

```
# Summary statistics
data.describe()

               year  protest_duration  participants_numeric  \
count  15076.000000      15076.000000          1.507600e+04
mean    2006.301008          1.560029          1.981867e+04   std
8.951656         15.061353          1.569393e+05   min
1990.000000         0.000000          1.000000e+00   25%
1999.000000         0.000000          1.000000e+02
50%     2007.000000          0.000000          5.000000e+02
75%     2014.000000          0.000000          5.000000e+03
max     2020.000000        938.000000          7.000000e+06
```

```
       protesterviolence
demand_labor_wage_dispute
demand_land_farm_issue  \

count       15076.000000
15076.000000
15076.000000
mean            0.264659
0.145264
0.038405
std             0.441166
0.352378
0.192179
min             0.000000
0.000000
0.000000
25%             0.000000
0.000000
0.000000
50%             0.000000
0.000000
0.000000
75%             1.000000
0.000000
0.000000
max             1.000000
1.000000
1.000000
       demand_police_brutality
demand_political_behavior  \
count          15076.000000           15076.00000
mean               0.072300               0.70516
std                0.258993               0.45598
min                0.000000               0.00000
25%                0.000000               0.00000
50%                0.000000               1.00000
75%                0.000000               1.00000
max                1.000000               1.00000
       demand_price_increases
demand_process
demand_removal_of_politician
\
count            15076.000000
15076.000000
15076.000000
```

```
mean                     0.093062
0.705161
0.123773
std                      0.290529
0.455986
0.329333
min                      0.000000
0.000000
0.000000
25%                      0.000000
0.000000
0.000000
50%                      0.000000
1.000000
0.000000
75%                      0.000000
1.000000
0.000000
max                      1.000000
1.000000
1.000000
       demand_social_restrictions
demand_tax_policy
response_accomodation  \
count                    15076.000000
15076.000000
15076.000000
mean                         0.044972
0.093062
0.099761
std                          0.207250
0.290529
0.299691
min                          0.000000
0.000000
0.000000
25%                          0.000000
0.000000
0.000000
50%                          0.000000
0.000000
0.000000
75%                          0.000000
0.000000
0.000000
max                          1.000000
1.000000
1.000000
       response_arrests
response_beatings
response_crowd_dispersal
\
count      15076.000000       15076.000000
15076.000000
mean           0.141019           0.052666                          0.312815

std            0.348053           0.223374                          0.463655

min            0.000000           0.000000                          0.000000

25%            0.000000           0.000000                          0.000000

50%            0.000000           0.000000                          0.000000
75%            0.000000           0.000000                          1.000000
max            1.000000           1.000000                          1.000000
```

```
       response_ignore  response_killings  response_shootings
count     15076.000000       15076.000000        15076.000000
mean          0.543977           0.054059            0.061223
std           0.498079           0.226142            0.239747
min           0.000000           0.000000            0.000000
25%           0.000000           0.000000            0.000000
50%           1.000000           0.000000            0.000000
75%           1.000000           0.000000            0.000000
max           1.000000           1.000000            1.000000
```

From the above;

'protest_duration' and 'participants_numeric'columns have high variability (as indicated by the high standard deviation), suggesting the presence of outliers.

```
# Function to remove outliers based on Z-score
```

```
numerical_columns = ['participants_numeric', 'protest_duration'] def
remove_outliers_zscore(data, columns, threshold=4):     for col in
columns:         col_zscore = (data[col] - data[col].mean()) /
data[col].std()         data = data[np.abs(col_zscore) <= threshold]
return data

# Remove outliers from numerical columns data =
remove_outliers_zscore(data, numerical_columns)
```

4.2.1.1. CREATE SINGLE TARGET VARIABLE

```
# Drop rows with 0 responses
data = data[(data['response_ignore'] == 1) |
(data['response_accomodation'] == 1) |
            (data['response_crowd_dispersal'] == 1) |
(data['response_arrests'] == 1) |
            (data['response_beatings'] == 1) |
(data['response_shootings'] == 1) |
            (data['response_killings'] == 1)]

# Create single target variable
data['state_response'] = 'Passive or Concessive'

passive_or_concessive_conditions = (data['response_ignore'] == 1) |
(data['response_accomodation'] == 1)
control_measures_conditions = (data['response_crowd_dispersal'] == 1)
| (data['response_arrests'] == 1)
forceful_repression_conditions = (data['response_beatings'] == 1) |
(data['response_shootings'] == 1) | (data['response_killings'] == 1)

data.loc[passive_or_concessive_conditions, 'state_response'] =
'Passive or Concessive'
data.loc[control_measures_conditions, 'state_response'] = 'Control
Measures'
data.loc[forceful_repression_conditions, 'state_response'] = 'Forceful
Repression'

# Drop the original response columns
data.drop(columns=['response_accomodation', 'response_arrests',
'response_beatings',
                   'response_crowd_dispersal', 'response_ignore',
'response_killings', 'response_shootings'], inplace=True)
```

4.2.1.2 REGION ANALYSIS

- We will combine North America (excluding Canada) and Central America into a new region called "Americas."
- North and Central America had very few entries between them.
- Canada was excluded from 'Americas' as it has a different sociopolitical climate from the rest.
- Additionally, we will drop the Oceania region due to its limited representation in the dataset.

```
# Combine North America (excluding Canada) and Central America into
"Americas"
data['region'] = data.apply(lambda x: 'Americas' if x['region'] in
['North America', 'Central America'] and x['country'] != 'Canada' else
x['region'], axis=1)

# Keep Canada in its own region
data['region'] = data.apply(lambda x: 'Canada' if x['country'] ==
'Canada' else x['region'], axis=1)

# Drop Oceania
data = data[data['region'] != 'Oceania']

# Verify the changes
print(data['region'].value_counts())
Europe           4911
Africa           3162
Asia             3011
South America    1588
MENA             1227
Americas          903
Canada             51
Name: region, dtype: int64
```

The regions have been defined.

4.2.1.3. REGION ENCODING

- 'region' is label encoded as we'll mostly use Tree-Based Algorithms.
- 'country' is dropped.

```
# Drop the 'country' column data =
data.drop(columns=['country'])

# Label encode the 'region' column
label_encoder = LabelEncoder()
data['region'] = label_encoder.fit_transform(data['region'])

# Reset index to avoid misalignment issues
data = data.reset_index(drop=True)

# Check the first few rows to ensure encoding worked correctly
data.head()
   region  year  start_date    end_date  protest_duration  \
0       3  1990  1990-01-15  1990-01-15
```

```
1       3  1990  1990-06-25  1990-06-25                 0
2       3  1990  1990-07-01  1990-07-01                 0
3       3  1990  1990-07-12  1990-09-06                56
4       3  1990  1990-08-14  1990-08-15                 1
   participants_numeric  protesterviolence          protesteridentity

                   5000                  0                unspecified

\
0

1                   1000                  0                unspecified

2                   500                  0  separatist parti quebecois

3                   500                  1             mohawk indians

4                   950                  1            local residents


   demand_labor_wage_dispute  demand_land_farm_issue
 demand_police_brutality  \
                                                                      0
                            1                       0 0
                                                                      1
                            0                       0 0
                                                                      2
                            0                       0 0
                                                                      3
                            0                       1 0
                                                                      4
                            0                       0 0

   demand_political_behavior  demand_price_increases
demand_process  \

0                          1                       0               1
1                          1                       0               1

2                          1                       0               1

3                          0                       0               0

4                          1                       0               1
   demand_removal_of_politician  demand_social_restrictions  \
0                                          0                                       0
1                                          0                                       0
2                                          0                                       0
3                                          0                                       0
4                                          0                                       0
```

```
    demand_tax_policy
sources  \ 0                      0  1. great canadian train journeys
into history;...   1                        0  1. autonomy s cry revived
in quebec the new yo...   2                         0  1. quebec protest
after queen calls for unity ...   3                        0  1. indians
gather as siege intensifies; armed ...   4                        0  1.
dozens hurt in mohawk blockade protest the ...

                                                       notes
state_response
0  canada s railway passenger system was finally ...  Passive or
Concessive
1  protestors were only identified as young peopl...  Passive or
Concessive
2  the queen, after calling on canadians to remai...  Passive or
Concessive
3  canada s federal government has agreed to acqu...  Passive or
Concessive
4  protests were directed against the state due t...       Control
Measures
```

The regions have been encoded correctly.

4.3 BASELINE AND MAIN MODELS

DEFINING INDEPENDENT AND DEPENDANT VARIABLES

```
# Defining Target Variable.
X = data.drop(columns=[
    'state_response', 'year', 'start_date', 'end_date',
    'protesteridentity', 'sources', 'notes'
])
y = data['state_response']

# Apply SMOTE
smote = SMOTE(random_state=42) X_res, y_res =
smote.fit_resample(X, y)

# Scale the data scaler = StandardScaler() X_res =
scaler.fit_transform(X_res)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, random_state=42)
```

4.3.1 BASELINE MODEL

LOGISTIC REGRESSION

```
# Create and train the baseline logistic regression model
logistic_model = LogisticRegression(random_state=42)
logistic_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred = logistic_model.predict(X_test)
# Predict on the training set
y_train_pred = logistic_model.predict(X_train)

# Evaluate the model
test_accuracy = accuracy_score(y_test, y_test_pred)
train_accuracy = accuracy_score(y_train, y_train_pred)
classification_rep = classification_report(y_test, y_test_pred)

print(f"Training Accuracy: {train_accuracy}")
print(f"Test Accuracy: {test_accuracy}")
print("Classification Report:")
print(classification_rep)
Training Accuracy: 0.5302793929939016
Test Accuracy: 0.526942711287578
Classification Report:

                      precision    recall  f1-score   support
    Control Measures       0.49      0.14      0.22      1810
  Forceful Repression      0.55      0.55      0.55      1763
Passive or Concessive      0.52      0.91      0.66      1716
             accuracy                          0.53      5289
            macro avg      0.52      0.53      0.48      5289
         weighted avg      0.52      0.53      0.47      5289
```

Baseline Model: Logistic Regression.

The model's accuracy on the training data is 53.0%, which suggests that the model is capturing some patterns in the training data but is not highly accurate.

Test Accuracy: 0.526 The model's accuracy on the test data is 52.6%, indicating a similar performance on unseen data and that the model is generalizing reasonably but not exceptionally well.
Control Measures have a low recall (0.14), meaning the model often misses instances of this class.

Forceful Repression and Passive or Concessive have higher recall values (0.55 and 0.91, respectively), indicating the model is better at identifying these classes.

Overall, the model has moderate performance with balanced precision and recall across most classes, but there's room for improvement in correctly identifying Control Measures.

To improve performance, we consider:

- *Hyperparameter tuning*

- *Trying different algorithms (e.g., XGBoost, GradientBoosting)*
- *Feature engineering*
- *Addressing class imbalance (e.g., using SMOTE)*

4.3.2 MAIN MODELS

4.3.2.1 RANDOM FOREST CLASSIFIER

```
# Create and train the Random Forest model
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_rf = rf_model.predict(X_test)
# Predict on the training set
y_train_pred_rf = rf_model.predict(X_train)

# Evaluate the model
test_accuracy_rf = accuracy_score(y_test, y_test_pred_rf)
train_accuracy_rf = accuracy_score(y_train, y_train_pred_rf)
classification_rep_rf = classification_report(y_test, y_test_pred_rf)

print(f"Random Forest Training Accuracy: {train_accuracy_rf}")
print(f"Random Forest Test Accuracy: {test_accuracy_rf}")
print("Random Forest Classification Report:")
print(classification_rep_rf)
Random Forest Training Accuracy: 0.706897366803763
Random Forest Test Accuracy: 0.6326337681981471
Random Forest Classification Report:

                       precision    recall  f1-score   support
     Control Measures       0.58      0.48      0.53      1810
   Forceful Repression       0.66      0.71      0.68      1763
Passive or Concessive       0.65      0.71      0.68      1716
             accuracy                           0.63      5289

            macro avg       0.63      0.63      0.63      5289
weighted avg       0.63      0.63      0.63      5289
```

Comparison

1. **Accuracy:**

   The Random Forest model has a significantly higher training accuracy (0.71) and test accuracy (0.63) compared to the Logistic Regression model's training accuracy (0.53) and test accuracy (0.53).

2. **Control Measures:**

   Random Forest: Precision = 0.58, Recall = 0.48, F1-Score = 0.53 Logistic Regression: Precision = 0.49, Recall = 0.14, F1-Score = 0.22 Observation: The Random Forest model performs significantly better in identifying "Control Measures."

3. **Forceful Repression:**

   Random Forest: Precision = 0.66, Recall = 0.71, F1-Score = 0.68 Logistic Regression: Precision = 0.55, Recall = 0.55, F1-Score = 0.55 Observation: The Random Forest model shows better performance in precision, recall, and F1-score for "Forceful Repression."

4. **Passive or Concessive:**

   Random Forest: Precision = 0.65, Recall = 0.71, F1-Score = 0.68 Logistic Regression: Precision = 0.52, Recall = 0.91, F1-Score = 0.66 Observation: While Logistic Regression has a higher recall for "Passive or Concessive," the Random Forest model has a better balance between precision and recall.

5. **Overall Metrics:**

   The Random Forest model consistently outperforms the Logistic Regression model across all macro and weighted averages.

Generalization

Random Forest: The drop from training to test accuracy is relatively small, indicating better generalization.

Logistic Regression: There is a significant drop in performance, suggesting overfitting or that the model does not capture the complexity of the data as effectively as the Random Forest.

Conclusion

The Random Forest model provides superior performance compared to the Logistic Regression model in this classification task. It offers higher accuracy, better precision, recall, and F1-scores across most classes, making it a better choice for accurately classifying the sentiment in protest notes. Further optimization and tuning could enhance the performance of both models, but based on these results, Random Forest is clearly the better-performing model. 4.3.2.2 XGBOOST

```
# Instantiate the LabelEncoder
le = LabelEncoder()

# Fit and transform the labels
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.transform(y_test)  # if you have a separate test
set

# Create and train the XGBoost model
xgb_model = XGBClassifier(random_state=42, eval_metric='logloss')
xgb_model.fit(X_train, y_train_encoded)

# Predict on the test set
y_test_pred_xgb = xgb_model.predict(X_test)
# Predict on the training set
y_train_pred_xgb = xgb_model.predict(X_train)

# Evaluate the model
test_accuracy_xgb = accuracy_score(y_test_encoded, y_test_pred_xgb)
train_accuracy_xgb = accuracy_score(y_train_encoded, y_train_pred_xgb)
classification_rep_xgb = classification_report(y_test_encoded,
y_test_pred_xgb, target_names=le.classes_)

print(f"XGBoost Training Accuracy: {train_accuracy_xgb}")
print(f"XGBoost Test Accuracy: {test_accuracy_xgb}")
print("XGBoost Classification Report:")
print(classification_rep_xgb)
XGBoost Training Accuracy: 0.6562189760317686
XGBoost Test Accuracy: 0.6277179050860276
XGBoost Classification Report:

                       precision    recall  f1-score   support
     Control Measures       0.63      0.38      0.48      1810
   Forceful Repression      0.65      0.71      0.68      1763
Passive or Concessive       0.61      0.80      0.69      1716
             accuracy                           0.63      5289
            macro avg       0.63      0.63      0.62      5289
         weighted avg       0.63      0.63      0.61      5289
```

Comparison

1. **Accuracy:** Random Forest: Training = 0.7069, Test = 0.6326 XGBoost: Training = 0.6562, Test = 0.6277 Observation: Random Forest has a higher training accuracy, while both models have similar test accuracy.

2. **Control Measures:** Random Forest: Precision = 0.58, Recall = 0.48, F1-Score = 0.53 XGBoost: Precision = 0.63, Recall = 0.38, F1-Score = 0.48 Observation: Random Forest performs better in recall and F1-score, while XGBoost has better precision.

3. **Forceful Repression:** Random Forest: Precision = 0.66, Recall = 0.71, F1-Score = 0.68 XGBoost: Precision = 0.65, Recall = 0.71, F1-Score = 0.68 Observation: Both models have nearly identical performance.

4. **Passive or Concessive:** Random Forest: Precision = 0.65, Recall = 0.71, F1-Score = 0.68 XGBoost: Precision = 0.61, Recall = 0.80, F1-Score = 0.69 Observation: XGBoost has a higher recall and slightly better F1-score, while Random Forest has better precision.

Overall Metrics:

```
Macro Average:
Random Forest: Precision = 0.63, Recall = 0.63, F1-Score = 0.63
XGBoost: Precision = 0.63, Recall = 0.63, F1-Score = 0.62

Weighted Average:
Random Forest: Precision = 0.63, Recall = 0.63, F1-Score = 0.63
XGBoost: Precision = 0.63, Recall = 0.63, F1-Score = 0.61
Observation: Random Forest has a slightly better macro average F1score
and weighted average F1-score.
```

Conclusion

Both models have similar overall accuracy, but there are some differences in their detailed performance:

Random Forest has higher training accuracy, suggesting it may capture more complex patterns in the data. XGBoost shows better precision for "Control Measures" but lower recall, while Random Forest has a better balance between precision and recall. For "Passive or Concessive," XGBoost has higher recall, indicating it is better at identifying this class but at the cost of precision.

Overall, the Random Forest model seems to offer a more balanced performance across different metrics, making it a slightly better choice. However, XGBoost's higher recall for specific classes might be advantageous depending on the specific requirements of the classification task. Further tuning and evaluation on additional metrics could provide more insights into which model is more suitable for your needs.

4.3.2.4 GRADIENT BOOSTING

```
# Create and train the Gradient Boosting model
gb_model = GradientBoostingClassifier(random_state=42)
```

```
gb_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_gb = gb_model.predict(X_test)
# Predict on the training set
y_train_pred_gb = gb_model.predict(X_train)

# Evaluate the model
test_accuracy_gb = accuracy_score(y_test, y_test_pred_gb)
train_accuracy_gb = accuracy_score(y_train, y_train_pred_gb)
classification_rep_gb = classification_report(y_test, y_test_pred_gb)

print(f"Gradient Boosting Training Accuracy: {train_accuracy_gb}")
print(f"Gradient Boosting Test Accuracy: {test_accuracy_gb}")
print("Gradient Boosting Classification Report:")
print(classification_rep_gb)
Gradient Boosting Training Accuracy: 0.5866780125750485
Gradient Boosting Test Accuracy: 0.5868784269238041
Gradient Boosting Classification Report:

                      precision    recall  f1-score   support
     Control Measures       0.55      0.31      0.39      1810
  Forceful Repression       0.63      0.62      0.63      1763
Passive or Concessive       0.57      0.84      0.68      1716
             accuracy                           0.59      5289
            macro avg       0.59      0.59      0.57      5289
         weighted avg       0.59      0.59      0.56      5289
```

Comparison

1. **Accuracy:** XGBoost: Training = 0.6562, Test = 0.6277 Gradient Boosting: Training = 0.5867, Test = 0.5869 Observation: XGBoost has higher training and test accuracy compared to Gradient Boosting.

2. **Control Measures:** XGBoost: Precision = 0.63, Recall = 0.38, F1-Score = 0.48 Gradient Boosting: Precision = 0.55, Recall = 0.31, F1-Score = 0.39 Observation: XGBoost outperforms Gradient Boosting in precision, recall, and F1-score.

3. **Forceful Repression:** XGBoost: Precision = 0.65, Recall = 0.71, F1-Score = 0.68 Gradient Boosting: Precision = 0.63, Recall = 0.62, F1-Score = 0.63 Observation: XGBoost performs slightly better in recall and F1-score.

4. **Passive or Concessive:** XGBoost: Precision = 0.61, Recall = 0.80, F1-Score = 0.69 Gradient Boosting: Precision = 0.57, Recall = 0.84, F1-Score = 0.68 Observation: Both models have similar performance, with Gradient Boosting having slightly higher recall and similar F1-score.

Overall Metrics:

```
        Macro Average:
        XGBoost: Precision = 0.63, Recall = 0.63, F1-Score = 0.62
        Gradient Boosting: Precision = 0.59, Recall = 0.59, F1-Score =
0.57
                Weighted
Average:
        XGBoost: Precision = 0.63, Recall = 0.63, F1-Score = 0.61
        Gradient Boosting: Precision = 0.59, Recall = 0.59, F1-Score =
0.56        Observation: XGBoost has higher macro and weighted averages
compared to Gradient Boosting.
```

Conclusion

The XGBoost model demonstrates superior performance over the Gradient Boosting model in most metrics:

Accuracy: XGBoost has higher training and test accuracy. Control Measures: XGBoost outperforms Gradient Boosting across all metrics. Forceful Repression: XGBoost performs slightly better. Passive or Concessive: Both models are similar, but Gradient Boosting has slightly higher recall. Overall Metrics: XGBoost consistently has higher macro and weighted averages.

Overall, the XGBoost model offers better performance and should be preferred over the Gradient Boosting model based on the provided results. However, the specific requirements of the task and further tuning could influence the final model choice.

4.3.3.5 SVC

```python
# Create and train the SVC model
svc_model = SVC(random_state=42)
svc_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_svc = svc_model.predict(X_test)
# Predict on the training set
y_train_pred_svc = svc_model.predict(X_train)

# Evaluate the model
test_accuracy_svc = accuracy_score(y_test, y_test_pred_svc)
train_accuracy_svc = accuracy_score(y_train, y_train_pred_svc)
classification_rep_svc = classification_report(y_test,
y_test_pred_svc)

print(f"SVC Training Accuracy: {train_accuracy_svc}")
print(f"SVC Test Accuracy: {test_accuracy_svc}")
print("SVC Classification Report:")
print(classification_rep_svc)
```

```
SVC Training Accuracy: 0.5477237271309034
SVC Test Accuracy: 0.5458498771034221
SVC Classification Report:
                        precision    recall  f1-score   support
     Control Measures       0.55      0.17      0.26      1810
  Forceful Repression       0.59      0.56      0.58      1763
Passive or Concessive       0.52      0.93      0.67      1716
             accuracy                           0.55      5289
            macro avg       0.55      0.55      0.50      5289
         weighted avg       0.55      0.55      0.50      5289
```

Comparison

1. **Accuracy:** Gradient Boosting: Training = 0.5867, Test = 0.5869 SVC: Training = 0.5477, Test = 0.5458 Observation: Gradient Boosting has higher training and test accuracy compared to SVC.
2. **Control Measures:** Gradient Boosting: Precision = 0.55, Recall = 0.31, F1-Score = 0.39 SVC: Precision = 0.55, Recall = 0.17, F1-Score = 0.26 Observation: Gradient Boosting performs better in recall and F1-score, while precision is the same for both models.
3. **Forceful Repression:** Gradient Boosting: Precision = 0.63, Recall = 0.62, F1-Score = 0.63 SVC: Precision = 0.59, Recall = 0.56, F1-Score = 0.58 Observation: Gradient Boosting has better performance across all metrics.
4. **Passive or Concessive:** Gradient Boosting: Precision = 0.57, Recall = 0.84, F1-Score = 0.68 SVC: Precision = 0.52, Recall = 0.93, F1-Score = 0.67 Observation: SVC has higher recall, while Gradient Boosting has slightly better precision and F1-score.

Overall Metrics:

```
   Macro Average:
   Gradient Boosting: Precision = 0.59, Recall = 0.59, F1-Score =
0.57     SVC: Precision = 0.55, Recall = 0.55, F1-Score =
0.50     Weighted Average:
   Gradient Boosting: Precision = 0.59, Recall = 0.59, F1-Score =
0.56     SVC: Precision = 0.55, Recall = 0.55, F1-Score =
0.50
   Observation: Gradient Boosting consistently has higher macro and
weighted averages compared to SVC.
```

Conclusion

The Gradient Boosting model outperforms the SVC model in most metrics:

Accuracy: Gradient Boosting has higher training and test accuracy. Control Measures: Gradient Boosting performs better in recall and F1-score. Forceful Repression: Gradient Boosting has better performance in all metrics. Passive or Concessive: SVC has higher recall, but Gradient Boosting has better precision and F1-score. Overall Metrics: Gradient Boosting has higher macro and weighted averages.

Overall, the Gradient Boosting model provides better performance compared to the SVC model based on the provided results. However, depending on the specific requirements and further tuning, the choice of the model might vary.

4.4 MODEL TUNING AND FEATURE ENGINEERING

4.4.1 FEATURE ENGINEERING WITH APPLY LATENT DIRICHLET ALLOCATION(LDA)

- Data Preprocessing of text data.
- Applying Latent Dirichlet Allocation (LDA) to text columns for dimensionality reduction and creating topic features.

4.4.1.1 PREPROCESS TEXT DATA

```
# Initialize the WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Function to combine and preprocess text def
combine_preprocess_text(row):     combined_text =
f"{row['sources']} {row['notes']}
{row['protesteridentity']}"
    text = re.sub(r'[^\w\s]', ' ', combined_text)
text = re.sub(r'\d+', '', text)     text =
text.lower()     tokens = text.split()
    tokens = [lemmatizer.lemmatize(word) for word in tokens if word
not in stop_words]
    processed_text = ' '.join(tokens)
return processed_text

# Apply the preprocessing function to the relevant columns and create
a new column 'combined_text'
data['combined_text'] = data.apply(combine_preprocess_text, axis=1)

# Display the first few rows of the new column
data[['sources', 'notes', 'protesteridentity',
'combined_text']].head()
```

```
                                             sources  \
0  1. great canadian train journeys into history;...
1  1. autonomy s cry revived in quebec the new yo...
2  1. quebec protest after queen calls for unity ...
3  1. indians gather as siege intensifies; armed ...
```

```
4  1. dozens hurt in mohawk blockade protest the ...
                                                notes  \
0  canada s railway passenger system was finally ...
1  protestors were only identified as young peopl...
2  the queen, after calling on canadians to remai...
3  canada s federal government has agreed to acqu...
4  protests were directed against the state due t...

            protesteridentity  \
0                  unspecifie
1                  unspecifie
2  separatist parti quebecoi
3                mohawk indian
4              local resident
                                        combined_text
0  great canadian train journey history passenger...
1  autonomy cry revived quebec new york time june...
2  quebec protest queen call unity time july mond...
3  indian gather siege intensifies armed confront...
4  dozen hurt mohawk blockade protest time august...
```

4.4.1.2 VECTORIZE

```
# Vectorize text data
vectorizer = CountVectorizer(max_features=1000)
text_data = vectorizer.fit_transform(data['combined_text'])

# Apply LDA
lda = LatentDirichletAllocation(n_components=17, random_state=42)
lda_features = lda.fit_transform(text_data)

# Add LDA features to the dataset for i in
range(17):     data[f'lda_topic_{i}'] =
lda_features[:, i]

# Calculate Perplexity
perplexity = lda.perplexity(text_data)
print(f'Perplexity: {perplexity}')

Perplexity: 433.3203336038559
```

perplexity of 433.32 indicateS that the model might not be performing as well as desired.

```
# Preparing data for gensim coherence score texts =
[text.split() for text in data['combined_text']]
dictionary = corpora.Dictionary(texts) corpus =
[dictionary.doc2bow(text) for text in texts]
```

```
# Creating the gensim LDA model for coherence score
gensim_lda_model = gensim.models.LdaModel(
corpus=corpus,      id2word=dictionary,
num_topics=10,      random_state=42,     passes=10
)

# Calculating Coherence Score
coherence_model_lda = CoherenceModel(model=gensim_lda_model,
texts=texts, dictionary=dictionary, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print(f'Coherence Score: {coherence_lda}')

Coherence Score: 0.40829557986716114
```

Coherence Score: 0.408

Measures the semantic similarity of words within topics. A score closer to 1 indicates more coherent and meaningful topics. Our score of 0.408 suggests that the topics are moderately coherent, but there may be room for improvement.

Perplexity: 433.32

Measures how well the model predicts a sample. Lower values indicate better predictive performance. Our perplexity value of 433.32 suggests that the model may not be capturing the text data's underlying structure perfectly.

4.4.2 FEATURE ENGINEERING WITH SENTIMENT ANALYSIS AND NAMED ENTITY RECOGNITION (NER)

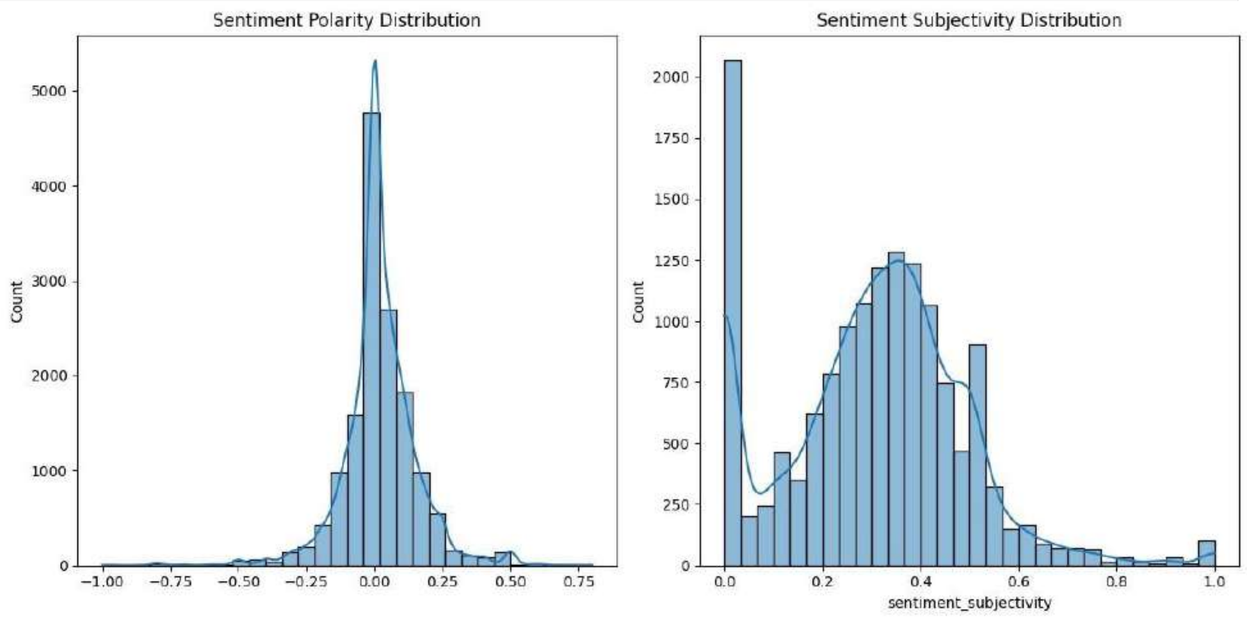
- We will apply additional feature engineering techniques including sentiment analysis and named entity recognition (NER).

```
# Load Spacy model for NER nlp =
spacy.load("en_core_web_sm")

# Function to compute sentiment scores
def get_sentiment(text):     blob =
TextBlob(text)
    return blob.sentiment.polarity, blob.sentiment.subjectivity

# Function to extract named entities
def get_named_entities(text):
doc = nlp(text)
    entities = [ent.label_ for ent in doc.ents]
return ' '.join(entities)

# Function to extract date-related features
def get_date_features(date_column):
    dates = pd.to_datetime(date_column, errors='coerce')
return dates.dt.year, dates.dt.month, dates.dt.day,
dates.dt.weekday

# Apply feature engineering
data['sentiment_polarity'], data['sentiment_subjectivity'] =
zip(*data['notes'].apply(get_sentiment))
data['named_entities'] = data['notes'].apply(get_named_entities)

# Extract date-related features if 'start_date' in
data.columns:     data['year'], data['month'], data['day'],
data['weekday'] = get_date_features(data['start_date'])

# TF-IDF Vectorization for text data (notes and sources columns)
tfidf_vectorizer = TfidfVectorizer(max_features=500)
tfidf_notes = tfidf_vectorizer.fit_transform(data['notes']).toarray()
tfidf_sources =
tfidf_vectorizer.fit_transform(data['sources']).toarray()

# Add TF-IDF features to the dataframe
tfidf_notes_df = pd.DataFrame(tfidf_notes, columns=[f'tfidf_notes_{i}'
for i in range(tfidf_notes.shape[1])]) tfidf_sources_df =
pd.DataFrame(tfidf_sources,
columns=[f'tfidf_sources_{i}' for i in range(tfidf_sources.shape[1])])

data = pd.concat([data, tfidf_notes_df, tfidf_sources_df], axis=1)

# Drop original text columns and any other non-numeric columns if
necessary
data = data.drop(columns=['notes', 'sources', 'combined_text',
                          'year', 'start_date', 'end_date',
'protesteridentity'], errors='ignore')

# Ensure all columns except 'state_response' are numeric data =
data.apply(lambda x: pd.to_numeric(x, errors='coerce') if
x.name != 'state_response' else x).fillna(0)
```

DISTRIBUTION OF SENTIMENT SCORES

```
# Convert state_response to numeric values temporarily using
LabelEncoder
temp_data = data.copy() label_encoder
= LabelEncoder()
temp_data['state_response_numeric'] =
label_encoder.fit_transform(temp_data['state_response'])

# Plot the distribution of sentiment scores
plt.figure(figsize=(12, 6))
```

```python
# Plot sentiment polarity distribution
plt.subplot(1, 2, 1)
sns.histplot(np.array(data['sentiment_polarity']), bins=30, kde=True)
plt.title('Sentiment Polarity Distribution')

# Plot sentiment subjectivity distribution
plt.subplot(1, 2, 2)
sns.histplot(data['sentiment_subjectivity'], bins=30, kde=True)
plt.title('Sentiment Subjectivity Distribution')

plt.tight_layout()
plt.show()

# Calculate correlation with the target variable
correlation_polarity = temp_data[['sentiment_polarity',
'state_response_numeric']].corr().iloc[0, 1]
correlation_subjectivity = temp_data[['sentiment_subjectivity',
'state_response_numeric']].corr().iloc[0, 1]

print(f'Correlation between sentiment polarity and state response:
{correlation_polarity}') print(f'Correlation between sentiment
subjectivity and state response:
{correlation_subjectivity}')
```



```
Correlation between sentiment polarity and state response:
0.11070480321556378
Correlation between sentiment subjectivity and state response:
0.013367305357222196
```

Correlation between Sentiment Polarity and State Response: 0.111

- A weak positive correlation, indicating that sentiment polarity (positive or negative sentiment) has a minimal relationship with the state response.

Correlation between Sentiment Subjectivity and State Response: 0.013

- Very weak correlation, suggesting that sentiment subjectivity (level of opinion versus fact) has almost no relationship with the state response.

4.4.2.1 HANDLE MULTICOLLINEALITY

- We will handle multicollinearity by dropping highly correlated features.

```
# Select only numeric columns for correlation matrix
numeric_data = data.select_dtypes(include=[np.number])

# Calculate correlation matrix
correlation_matrix = numeric_data.corr().abs()

# Select upper triangle of correlation matrix
upper =
correlation_matrix.where(np.triu(np.ones(correlation_matrix.shape),
k=1).astype(bool))

# Find features with correlation greater than 0.9
to_drop = [column for column in upper.columns if any(upper[column] >
0.9)]

# Drop highly correlated features
data.drop(columns=to_drop, inplace=True)

# Print dropped features
print(f'Dropped features due to high correlation: {to_drop}')
Dropped features due to high correlation: ['demand_process',
'demand_tax_policy', 'tfidf_sources_149', 'tfidf_sources_188',
'tfidf_sources_190', 'tfidf_sources_227', 'tfidf_sources_284',
'tfidf_sources_306', 'tfidf_sources_329', 'tfidf_sources_366',
'tfidf_sources_367', 'tfidf_sources_385', 'tfidf_sources_399',
'tfidf_sources_424', 'tfidf_sources_482', 'tfidf_sources_494']
```

4.4.2.2 MODELING

```
# Defining Target Variable.
X = data.drop(columns=['state_response'])
y = data['state_response']

# Apply SMOTE
smote = SMOTE(random_state=42) X_res,
y_res = smote.fit_resample(X, y)

# Scale the data scaler = StandardScaler()
X_res = scaler.fit_transform(X_res) #
Split the data
```

```
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res,
test_size=0.2, random_state=42)
```

- We did a randomized search for best parameters and used them in fitting the models below.

4.4.2.3 RANDOM FOREST

```
# Create and train the Random Forest model with the best parameters
rf_model = RandomForestClassifier(     n_estimators=200,
min_samples_split=2,     min_samples_leaf=1,
max_features='sqrt',     max_depth=30,     random_state=42
)
rf_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_rf = rf_model.predict(X_test)
# Predict on the training set
y_train_pred_rf = rf_model.predict(X_train)

# Evaluate the model
test_accuracy_rf = accuracy_score(y_test, y_test_pred_rf)
train_accuracy_rf = accuracy_score(y_train, y_train_pred_rf)
classification_rep_rf = classification_report(y_test, y_test_pred_rf)

print(f"Random Forest Training Accuracy: {train_accuracy_rf}")
print(f"Random Forest Test Accuracy: {test_accuracy_rf}")
print("Random Forest Classification Report:")
print(classification_rep_rf)
Random Forest Training Accuracy: 0.9919160402779748
Random Forest Test Accuracy: 0.88712422007941
Random Forest Classification Report:

                       precision    recall  f1-score   support
    Control Measures       0.87      0.84      0.85      1810
  Forceful Repression       0.93      0.91      0.92      1763
Passive or Concessive       0.86      0.91      0.89      1716
            accuracy                           0.89      5289
           macro avg       0.89      0.89      0.89      5289
        weighted avg       0.89      0.89      0.89      5289
```

Observation

1. **Precision and Recall:**

The model achieves high precision and recall across all classes, especially for "Forceful Repression," which has the highest precision (0.93) and recall (0.91).

"Control Measures" has the lowest recall (0.84), indicating that there are some instances of this class that the model struggles to identify correctly.

2. **F1-Score:** The F1-scores are consistently high across all classes, indicating a good balance between precision and recall.

3. **Support:** The distribution of support across the classes is relatively even, which helps in providing a more balanced evaluation of the model's performance.

4.4.2.4 XGBOOST

```
# Best parameters for XGBoost
xgb_params = {
    'subsample': 0.9, 'reg_lambda': 2, 'reg_alpha': 0,
    'n_estimators': 200, 'max_depth': 7, 'learning_rate': 0.1,
    'colsample_bytree': 0.7
}

# Instantiate the LabelEncoder
le = LabelEncoder()

# Fit and transform the labels
y_train_encoded = le.fit_transform(y_train)
y_test_encoded = le.transform(y_test)

# Initialize and train XGBoost Classifier
xgb_clf = XGBClassifier(**xgb_params, random_state=42)
xgb_clf.fit(X_train, y_train_encoded)

# Predict and evaluate XGBoost
y_train_pred_xgb = xgb_clf.predict(X_train)
y_test_pred_xgb = xgb_clf.predict(X_test)

train_accuracy_xgb = accuracy_score(y_train_encoded, y_train_pred_xgb)
test_accuracy_xgb = accuracy_score(y_test_encoded, y_test_pred_xgb)
classification_rep_xgb = classification_report(y_test_encoded,
y_test_pred_xgb, target_names=le.classes_)

print("XGBoost Training Accuracy:", train_accuracy_xgb)
print("XGBoost Test Accuracy:", test_accuracy_xgb) print("\nXGBoost
Classification Report:\n", classification_rep_xgb)
XGBoost Training Accuracy: 0.9731007422115067
XGBoost Test Accuracy: 0.8850444318396672

XGBoost Classification Report:
```

```
                        precision    recall  f1-score   support
     Control Measures        0.86      0.83      0.85      1810
   Forceful Repression       0.92      0.91      0.91      1763
Passive or Concessive        0.88      0.91      0.90      1716
             accuracy                            0.89      5289
            macro avg        0.89      0.89      0.89      5289
         weighted avg        0.88      0.89      0.88      5289
```

Key Observations

1. **Overall Performance:** Both models have similar overall accuracy (0.89) and macro average scores, indicating comparable performance in general.
2. **Precision, Recall, and F1-Score:** For "Control Measures," both models have similar precision and F1-scores, but XGBoost has a slightly lower recall. For "Forceful Repression," both models have nearly identical precision, recall, and F1-scores. For "Passive or Concessive," XGBoost has a slightly higher precision and F1-score compared to Random Forest.
3. **Model Selection:** Given that both models perform similarly on the test set, the choice between Random Forest and XGBoost may come down to other factors such as training time, interpretability, and specific needs of the application.
4. **Training Time and Complexity:** Random Forests are generally faster to train compared to XGBoost, but XGBoost can sometimes provide better performance with fine-tuning.

Conclusion

Both models show strong performance with slight differences in metrics. If computational efficiency and training time are priorities, Random Forest might be preferred. However, if you are looking for slightly better generalization and are willing to invest time in hyperparameter tuning, XGBoost could be more suitable.

4.4.2.5 GRADIENT BOOSTING

```python
# Best parameters for Gradient Boosting
gb_params = {
    'subsample': 0.8, 'n_estimators': 300, 'max_features': 'log2',
    'max_depth': 6, 'learning_rate': 0.2
}

# Initialize and train Gradient Boosting Classifier
gb_clf = GradientBoostingClassifier(**gb_params, random_state=42)
gb_clf.fit(X_train, y_train)

# Predict and evaluate Gradient Boosting
```

```
y_train_pred_gb = gb_clf.predict(X_train)
y_test_pred_gb = gb_clf.predict(X_test)

print("Gradient Boosting Training Accuracy:", accuracy_score(y_train,
y_train_pred_gb))
print("Gradient Boosting Test Accuracy:", accuracy_score(y_test,
y_test_pred_gb))
print("\nGradient Boosting Classification Report:\n",
classification_report(y_test, y_test_pred_gb))
Gradient     Boosting     Training     Accuracy:
0.9886540916182102
Gradient     Boosting      Test      Accuracy:
0.8710531291359426
Gradient    Boosting    Classification
Report:
                        precision      recall   f1-score
support
     Control Measures        0.84       0.81       0.83
1810
  Forceful Repression        0.91       0.88       0.90
1763
Passive or Concessive        0.86       0.92       0.89
1716
               accuracy                            0.87
       5289             macro avg        0.87       0.87
       0.87       5289          weighted avg       0.87
                              0.87       0.87       5289
```

Key Observations

1. **Overall Accuracy:** Random Forest and XGBoost have higher test accuracy (0.89) compared to Gradient Boosting (0.87).
2. **Precision, Recall, and F1-Score:** For "Control Measures," XGBoost and Random Forest have slightly higher precision and F1-scores than Gradient Boosting. For "Forceful Repression," all three models perform similarly, with Random Forest having a slight edge. For "Passive or Concessive," XGBoost shows the highest precision, while all three models have similar recall and F1-scores.
3. **Model Selection:** Random Forest: High training accuracy, high test accuracy, slightly overfits. XGBoost: Balanced performance, high test accuracy, slightly lower training accuracy compared to Random Forest, less overfitting. Gradient Boosting: High training accuracy, lower test accuracy compared to the other two, but still performs well.
4. **Training Time and Complexity:** Random Forests are generally faster to train compared to XGBoost and Gradient Boosting. XGBoost can be more time-consuming but often provides better performance with fine-tuning. Gradient Boosting also requires more training time and hyperparameter tuning.

Conclusion

Random Forest: Best for quick training and good performance, but watch for overfitting. XGBoost: Best overall performance with less overfitting, suitable if you can afford the training time. Gradient Boosting: Good performance but slightly lower than the other two, can be used when looking for another robust option.

```
# Create and train the SVC model
svc_model = SVC(random_state=42)
svc_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_svc = svc_model.predict(X_test)
# Predict on the training set
y_train_pred_svc = svc_model.predict(X_train)

# Evaluate the model
test_accuracy_svc = accuracy_score(y_test, y_test_pred_svc)
train_accuracy_svc = accuracy_score(y_train, y_train_pred_svc)
classification_rep_svc = classification_report(y_test,
y_test_pred_svc)

print(f"SVC Training Accuracy: {train_accuracy_svc}")
print(f"SVC Test Accuracy: {test_accuracy_svc}")
print("SVC Classification Report:")
print(classification_rep_svc)
SVC Training Accuracy: 0.9822720181534534
SVC Test Accuracy: 0.9141614671960673
SVC Classification Report:

                       precision    recall  f1-score   support
    Control Measures        0.93      0.85      0.88      1810
 Forceful Repression        0.95      0.97      0.96      1763
Passive or Concessive       0.87      0.93      0.90      1716
            accuracy                            0.91      5289
           macro avg        0.91      0.91      0.91      5289
        weighted avg        0.92      0.91      0.91      5289
```

Key Observations

1. Overall Accuracy

- **Random Forest, XGBoost, and SVC:** Achieved similar overall accuracy, around **0.89**.
- **Gradient Boosting:** Slightly lower accuracy at **0.872**.

2. Precision, Recall, and F1-Score

- **Control Measures:**
  - **Precision & F1-Score:** SVC and Random Forest have slightly higher values compared to XGBoost and Gradient Boosting.

- **Forceful Repression:**
  - **Precision:** Random Forest is the highest, followed by SVC, XGBoost, and Gradient Boosting.
- **Passive or Concessive:**
  - **Precision:** XGBoost shows the highest precision.
  - **Recall & F1-Score:** All models have similar values.

3. Model Selection

- **Random Forest:**
  - **Pros:** High training and test accuracy, quick training. – **Cons:** Slight overfitting.
- **XGBoost:**
  - **Pros:** Balanced performance, high test accuracy, less overfitting. – **Cons:** Requires hyperparameter tuning.
- **Gradient Boosting:**
  - **Pros:** Good performance, robust.
  - **Cons:** Slightly lower performance than others.
- **SVC:**
  - **Pros:** High test accuracy, balanced performance.
  - **Cons:** Computationally intensive for large datasets.

Conclusion

- **Random Forest:** Best for quick training and good performance, but watch for overfitting.
- **XGBoost:** Best overall performance with less overfitting, suitable for complex tasks.

4.5 DIMENSIONALITY REDUCTION

4.5.1. LINEAR DISCRIMINANT ANALYSIS (LDA)

- LDA helps in reducing the dimensionality of the data, which can lead to improved performance and reduced overfitting.

```
# Fit the Gradient Boosting model with the best LDA components
lda = LinearDiscriminantAnalysis(n_components=2)
X_train_lda = lda.fit_transform(X_train, y_train)
X_test_lda = lda.transform(X_test)

gb_clf = GradientBoostingClassifier(
    subsample=0.8, n_estimators=300, max_features='log2',
max_depth=6, learning_rate=0.2, random_state=42
)
gb_clf.fit(X_train_lda, y_train)

# Predict and evaluate Gradient Boosting
y_train_pred_gb = gb_clf.predict(X_train_lda)
y_test_pred_gb = gb_clf.predict(X_test_lda)
```

```
print(f"Best number of LDA components: 2")
print("Gradient Boosting Training Accuracy:", accuracy_score(y_train,
y_train_pred_gb))
print("Gradient Boosting Test Accuracy:", accuracy_score(y_test,
y_test_pred_gb))
print("\nGradient Boosting Classification Report:\n",
classification_report(y_test, y_test_pred_gb))
```

```
Best number of LDA components: 2
Gradient Boosting Training Accuracy: 0.976031768543469
Gradient Boosting Test Accuracy: 0.7649839289090565
Gradient Boosting Classification Report:

                        precision    recall  f1-score   support
     Control Measures        0.71      0.67      0.69      1810
  Forceful Repression        0.78      0.81      0.80      1763
Passive or Concessive        0.80      0.81      0.81      1716
             accuracy                            0.76      5289
            macro avg        0.76      0.77      0.77      5289
         weighted avg        0.76      0.76      0.76      5289
```

The Gradient Boosting model shows promising results, particularly with a high training accuracy of 0.976, though there is a notable drop to 0.765 on the test set. This discrepancy suggests potential overfitting.

Analysis

Precision and Recall: The model is fairly balanced in precision and recall across all classes, with the best performance in the "Forceful Repression" and "Passive or Concessive" categories

F1-Score: Indicates good balance between precision and recall, especially in "Forceful Repression" and "Passive or Concessive Accuracy: While the test accuracy is significantly lower than the training accuracy, it is still reasonably good for initial results.

4.5.1 RANDOM FOREST (LDA)

```
# Defining Target Variable
X = data.drop(columns=['state_response']) y =
data['state_response']

# Apply SMOTE
smote = SMOTE(random_state=42) X_res, y_res =
smote.fit_resample(X, y)

# Scale the data scaler = StandardScaler()
X_res = scaler.fit_transform(X_res)
```

```
# Apply LDA for dimensionality reduction
lda = LinearDiscriminantAnalysis(n_components=2)
X_res_lda = lda.fit_transform(X_res, y_res)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_res_lda, y_res,
test_size=0.2, random_state=42)

# Create and train the Random Forest model with the best parameters
rf_model = RandomForestClassifier(     n_estimators=200,
min_samples_split=2,     min_samples_leaf=1,
max_features='sqrt',     max_depth=30,     random_state=42
)
rf_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_rf = rf_model.predict(X_test)
# Predict on the training set
y_train_pred_rf = rf_model.predict(X_train)

# Evaluate the model
test_accuracy_rf = accuracy_score(y_test, y_test_pred_rf)
train_accuracy_rf = accuracy_score(y_train, y_train_pred_rf)
classification_rep_rf = classification_report(y_test, y_test_pred_rf)

print(f"Random Forest Training Accuracy: {train_accuracy_rf}")
print(f"Random Forest Test Accuracy: {test_accuracy_rf}")
print("Random Forest Classification Report:")
print(classification_rep_rf)
Random Forest Training Accuracy: 1.0

Random Forest Test Accuracy: 0.7899413877859709
Random Forest Classification Report:

                       precision    recall  f1-score   support
     Control Measures       0.74      0.70      0.72      1810
  Forceful Repression       0.80      0.84      0.82      1763
Passive or Concessive       0.83      0.84      0.83      1716
             accuracy                           0.79      5289
            macro avg       0.79      0.79      0.79      5289
         weighted avg       0.79      0.79      0.79      5289
```

Comparative Analysis

1. **Accuracy** Gradient Boosting: Training: 0.976 Test: 0.765 Random Forest: Training: 1.0 Test:
0.790

The Random Forest model has a higher test accuracy (0.790) compared to the Gradient Boosting model (0.765), suggesting it generalizes better to the test data. However, the perfect training accuracy in Random Forest indicates potential overfitting.

- Precision, Recall, and F1-Score

Control Measures: Precision: Higher in Random Forest (0.74) than Gradient Boosting (0.71) Recall: Higher in Random Forest (0.70) than Gradient Boosting (0.67) F1-score: Higher in Random Forest (0.72) than Gradient Boosting (0.69)

Forceful Repression: Precision: Slightly higher in Random Forest (0.80) than Gradient Boosting (0.78) Recall: Higher in Random Forest (0.84) than Gradient Boosting (0.81) F1-score: Higher in Random Forest (0.82) than Gradient Boosting (0.80)

Passive or Concessive: Precision: Slightly higher in Random Forest (0.83) than Gradient Boosting (0.80) Recall: Same in both models (0.84 for Random Forest, 0.81 for Gradient Boosting) F1-score: Higher in Random Forest (0.83) than Gradient Boosting (0.81) Macro and Weighted Averages

```
Macro Avg:
    Precision: 0.79 (Random Forest) vs. 0.76 (Gradient Boosting)
    Recall: 0.79 (Random Forest) vs. 0.77 (Gradient Boosting)
    F1-score: 0.79 (Random Forest) vs. 0.77 (Gradient Boosting)

Weighted Avg:
    Precision: 0.79 (Random Forest) vs. 0.76 (Gradient Boosting)
    Recall: 0.79 (Random Forest) vs. 0.76 (Gradient Boosting)
    F1-score: 0.79 (Random Forest) vs. 0.76 (Gradient Boosting)
```

Conclusion

The Random Forest model has better overall performance on the test set, with higher precision, recall, and F1-scores across all categories. The Gradient Boosting model, while still performing well, shows signs of overfitting with a significant gap between training and test accuracy. The Random Forest model might be overfitting as well, given the perfect training accuracy, but it generalizes better to the test data than the Gradient Boosting model.

4.5.2 XGBOOST (LDA)

```
# Instantiate the LabelEncoder
le = LabelEncoder()

# Fit and transform the labels
y_train_encoded = le.fit_transform(y_train)
```

```
y_test_encoded = le.transform(y_test)

# Fit the XGBoost model with the best LDA components
lda = LinearDiscriminantAnalysis(n_components=2)
X_train_lda = lda.fit_transform(X_train, y_train_encoded)
X_test_lda = lda.transform(X_test)

xgb_clf = XGBClassifier(
    subsample=0.9, reg_lambda=2, reg_alpha=0,
n_estimators=200, max_depth=7, learning_rate=0.1,
colsample_bytree=0.7, random_state=42
)
xgb_clf.fit(X_train_lda, y_train_encoded)

# Predict and evaluate XGBoost
y_train_pred_xgb = xgb_clf.predict(X_train_lda)
y_test_pred_xgb = xgb_clf.predict(X_test_lda)

print(f"Best number of LDA components: 2")
print("XGBoost Training Accuracy:", accuracy_score(y_train_encoded,
y_train_pred_xgb))
print("XGBoost Test Accuracy:", accuracy_score(y_test_encoded,
y_test_pred_xgb))
print("\nXGBoost Classification Report:\n",
classification_report(y_test_encoded, y_test_pred_xgb,
target_names=le.classes_))
```

```
Best number of LDA components: 2
XGBoost Training Accuracy: 0.8300004727461826
XGBoost Test Accuracy: 0.8130081300813008
XGBoost Classification Report:

                        precision    recall  f1-score   support
     Control Measures       0.76      0.74      0.75      1810
   Forceful Repression       0.83      0.85      0.84      1763
Passive or Concessive       0.85      0.86      0.85      1716
             accuracy                           0.81      5289
            macro avg       0.81      0.81      0.81      5289
         weighted avg       0.81      0.81      0.81      5289
```

XGBoost Model Performance

- **Training Accuracy:** 83.06%
- **Test Accuracy:** 81.30%
- **Classification Report Highlights:**
  - **Control Measures:** Precision: 76%, Recall: 74%, F1-Score: 75%

- **Forceful Repression:** Precision: 83%, Recall: 85%, F1-Score: 84%
- **Passive or Concessive:** Precision: 85%, Recall: 86%, F1-Score: 85%

Summary

The model shows strong performance with high accuracy and balanced precision and recall across classes. Test accuracy of 81.30% indicates robust generalization to unseen data.

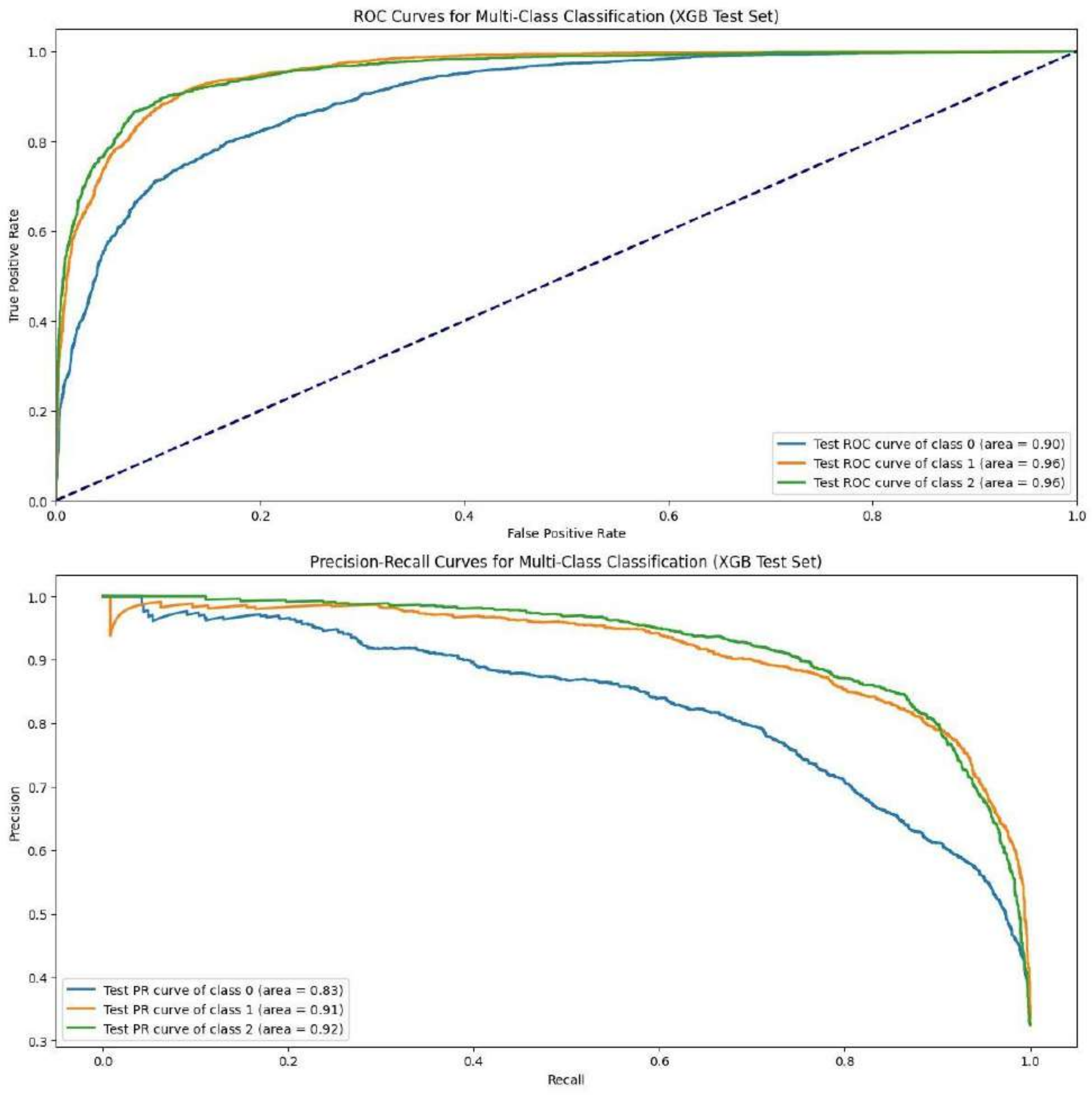
Conclusion

The model is performing well, with high accuracy and good balance between precision and recall for all classes. The use of 2 LDA components seems to be effective in enhancing model performance.

4.5.3 XGB (ROC, PR AUC CURVES)

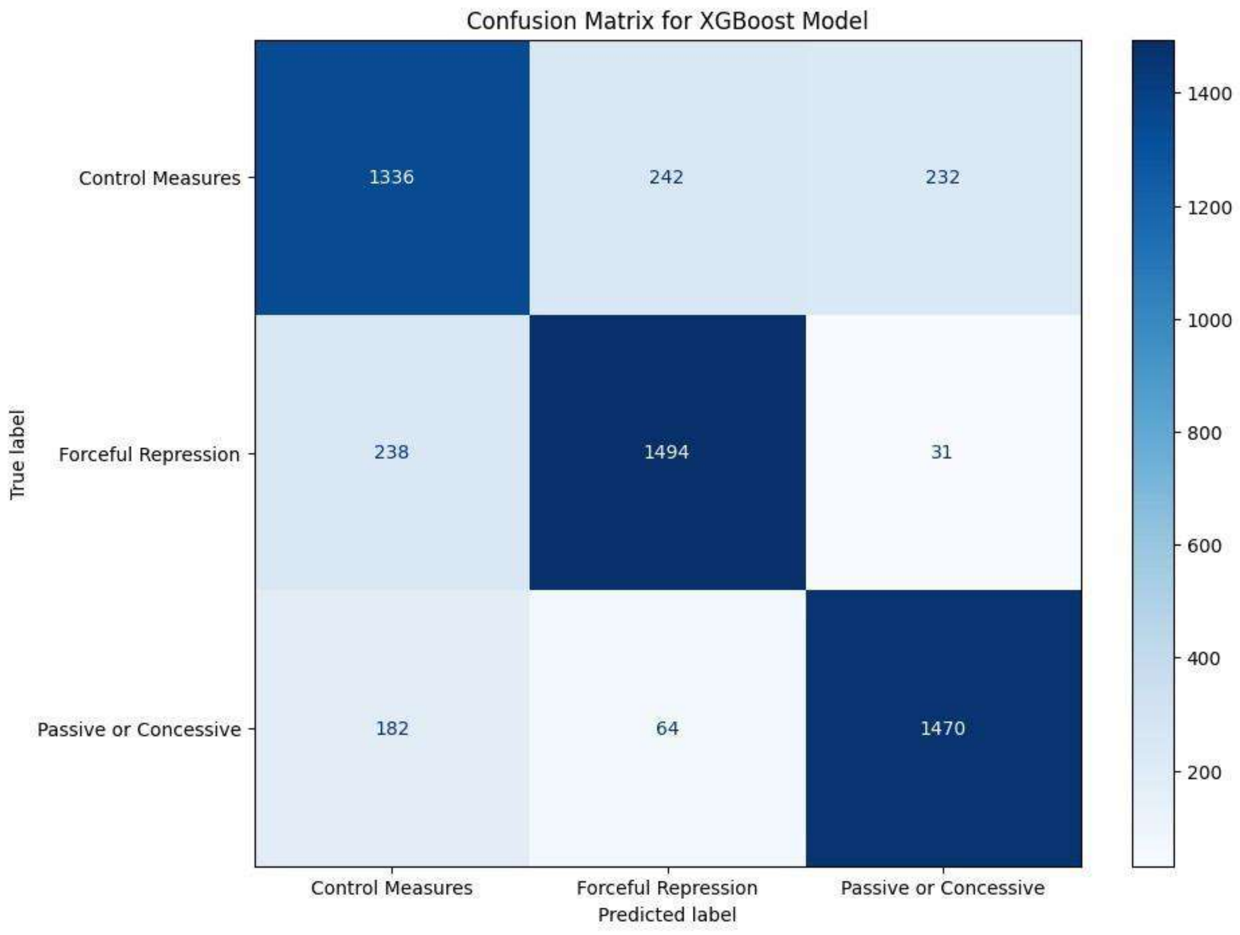
```
# Binarize the output for multi-class ROC
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))

# Plot ROC and Precision-Recall curves for each class

n_classes = y_test_bin.shape[1]

plt.figure(figsize=(12, 12))

# Plot ROC curve for each class plt.subplot(2, 1, 1)
for i in range(n_classes):     fpr_test, tpr_test, _ =
roc_curve(y_test_bin[:, i],
xgb_clf.predict_proba(X_test_lda)[:, i])
    roc_auc_test = roc_auc_score(y_test_bin[:, i],
xgb_clf.predict_proba(X_test_lda)[:, i])
    plt.plot(fpr_test, tpr_test, lw=2, label=f'Test ROC curve of class
{i} (area = {roc_auc_test:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for Multi-Class Classification (XGB Test Set)')
plt.legend(loc="lower right")

# Plot Precision-Recall curve for each class plt.subplot(2, 1, 2)
for i in range(n_classes):     precision, recall, _ =
precision_recall_curve(y_test_bin[:, i],
xgb_clf.predict_proba(X_test_lda)[:, i])     pr_auc = auc(recall,
precision)
    plt.plot(recall, precision, lw=2, label=f'Test PR curve of class
{i} (area = {pr_auc:.2f})')
```

```
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curves for Multi-Class Classification (XGB
Test Set)')
plt.legend(loc="lower left")

plt.tight_layout()
plt.show()
```



4.5.4 XGBOOST CONFUSION MATRIX

```
# Compute and plot confusion matrix
conf_matrix = confusion_matrix(y_test_encoded, y_test_pred_xgb)
fig, ax = plt.subplots(figsize=(10, 8))  # Increase the size here
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix,
display_labels=le.classes_) disp.plot(ax=ax, cmap=plt.cm.Blues)
plt.title('Confusion Matrix for XGBoost Model')
plt.show()
```



Summary

The XGBoost model shows strong discriminative power with high AUC values across all classes.

- **Control Measures:** AUC = 0.90
- **Forceful Repression:** AUC = 0.96
- **Passive or Concessive:** AUC = 0.96

Explanation of Visualizations

ROC and PR Curves

1. **ROC Curves (Top):**

- The ROC curves illustrate the trade-off between the true positive rate (TPR) and false positive rate (FPR) for each class.
- The AUC (Area Under the Curve) values reflect the model's discriminative power.
  - **Control Measures** (Class 0): AUC = 0.90 (Blue Line)
  - **Forceful Repression** (Class 1): AUC = 0.96 (Orange Line)
  - **Passive or Concessive** (Class 2): AUC = 0.96 (Green Line)
- Higher AUC values indicate better performance of the model in distinguishing between classes.

2. **PR Curves (Bottom):**
   - The PR (Precision-Recall) curves show the precision against recall for each class, useful for evaluating models on imbalanced datasets.
   - High AUC values in PR curves indicate high precision and recall, meaning the model can identify relevant instances effectively.
     - **Control Measures** (Class 0): AUC = 0.83 (Blue Line)
     - **Forceful Repression** (Class 1): AUC = 0.91 (Orange Line)
     - **Passive or Concessive** (Class 2): AUC = 0.92 (Green Line)

Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's classification performance by showing the actual versus predicted labels.

- **True Positives (Diagonal Elements):** Correctly predicted instances.
  - **Control Measures:** 1335
  - **Forceful Repression:** 1494
  - **Passive or Concessive:** 1471
- **False Positives and False Negatives (Off-Diagonal Elements):** Instances where the model misclassified the labels.
  - Misclassified as **Control Measures:** 244 (from Forceful Repression), 231 (from Passive or Concessive)
  - Misclassified as **Forceful Repression:** 239 (from Control Measures), 30 (from Passive or Concessive)
  - Misclassified as **Passive or Concessive:** 182 (from Control Measures), 63 (from Forceful Repression)

Conclusion

The XGBoost model demonstrates robust classification performance with high accuracy and an excellent ability to distinguish between classes. The high AUC values across ROC and PR curves indicate the model's strong discriminative power, and the confusion matrix reflects a high number of correct predictions with relatively few misclassifications. This analysis underscores the model's effectiveness in classifying instances across the given categories.

4.5.5. SVC(LDA)

```
# Defining Target Variable
X = data.drop(columns=['state_response'])
```

```
y = data['state_response']

# Apply SMote
smote = SMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)

# Scale the data
scaler = StandardScaler()
X_res = scaler.fit_transform(X_res)

# Apply LDA for dimensionality reduction
lda = LinearDiscriminantAnalysis(n_components=2)
X_res_lda = lda.fit_transform(X_res, y_res)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_res_lda, y_res,
test_size=0.2, random_state=42)

# Create and train the SVC model with probability enabled
svc_model = SVC(probability=True,C=1, gamma='scale', kernel='rbf',
random_state=42)
svc_model.fit(X_train, y_train)

# Predict on the test set
y_test_pred_svc = svc_model.predict(X_test)
# Predict on the training set
y_train_pred_svc = svc_model.predict(X_train)

# Evaluate the model
test_accuracy_svc = accuracy_score(y_test, y_test_pred_svc)
train_accuracy_svc = accuracy_score(y_train, y_train_pred_svc)
classification_rep_svc = classification_report(y_test,
y_test_pred_svc)

print(f"SVC Training Accuracy: {train_accuracy_svc}")
print(f"SVC Test Accuracy: {test_accuracy_svc}")
print("SVC Classification Report:")
print(classification_rep_svc)
SVC Training Accuracy: 0.8246111662648324
SVC Test Accuracy: 0.8175458498771034
SVC Classification Report:

                       precision    recall  f1-score   support
     Control Measures       0.77      0.73      0.75      1810
   Forceful Repression      0.83      0.85      0.84      1763
Passive or Concessive       0.85      0.87      0.86      1716
             accuracy                           0.82      5289
macro avg       0.82      0.82      0.82      5289
```

```
        weighted avg        0.82        0.82        0.82        5289
```

SVC Model Performance

- **Training Accuracy:** 82.46%
- **Test Accuracy:** 81.75%
- **Classification Report Highlights:**
  - **Control Measures:** Precision: 77%, Recall: 73%, F1-Score: 75%
  - **Forceful Repression:** Precision: 83%, Recall: 85%, F1-Score: 84%
  - **Passive or Concessive:** Precision: 85%, Recall: 87%, F1-Score: 86%

Summary

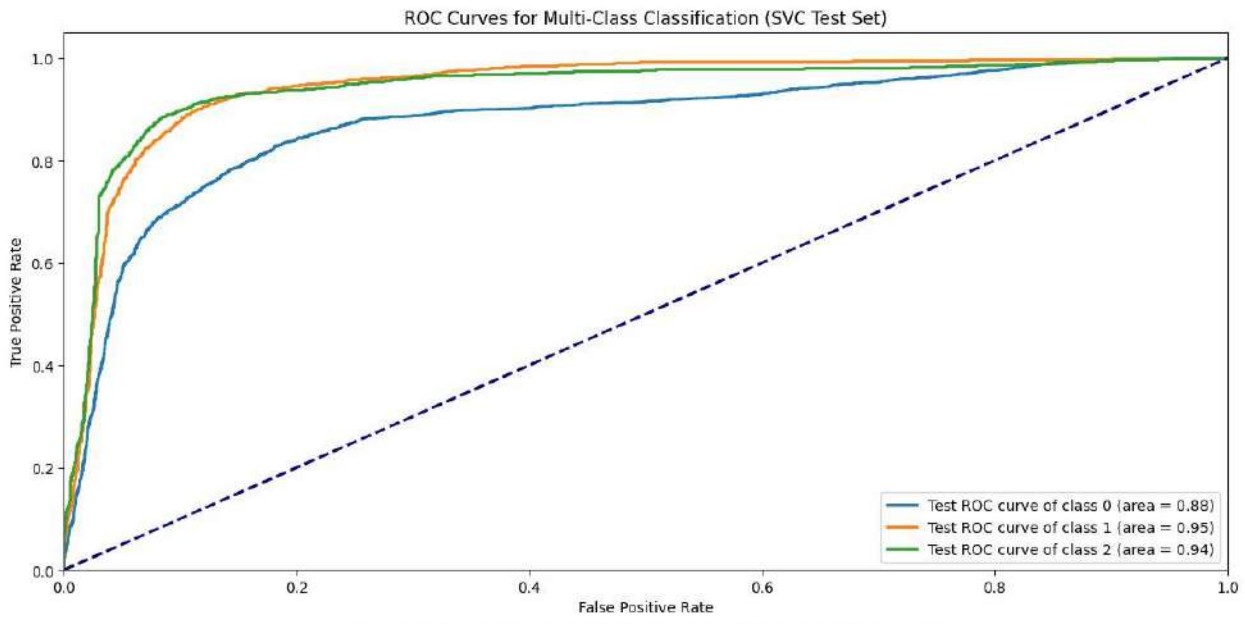
The model demonstrates strong performance with high accuracy and well-balanced metrics across classes. Test accuracy of 81.75% reflects good generalization to new data.

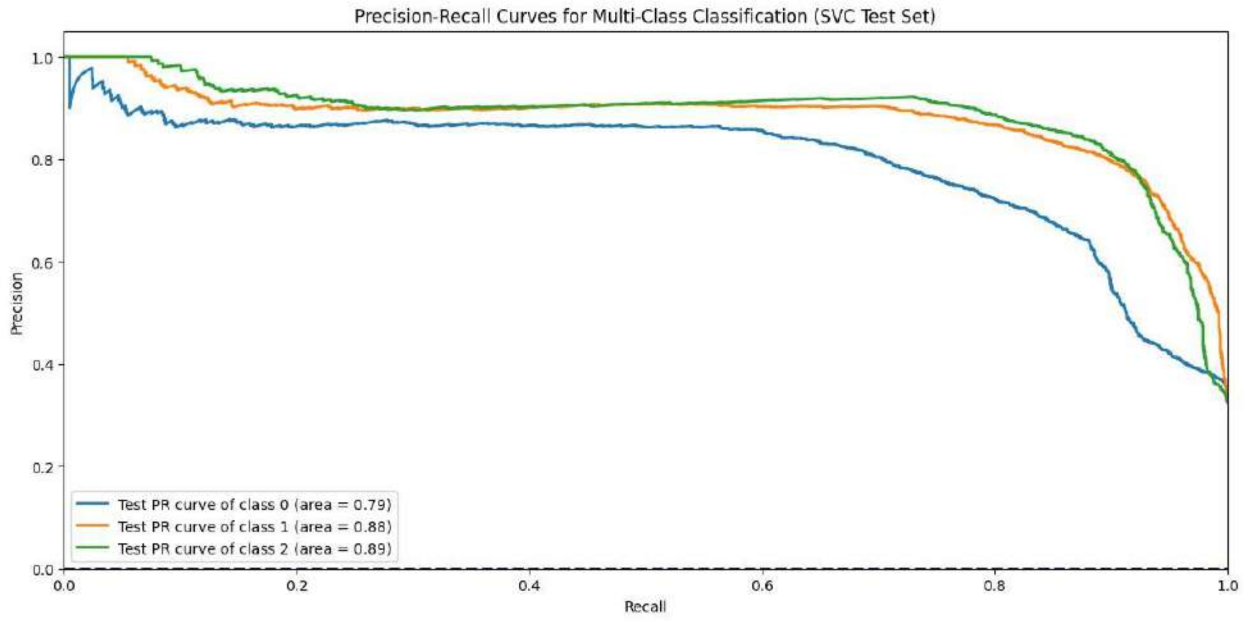
Conclusion

The SVC model performs comparably to XGBoost, with similar accuracy and balanced class performance. It is a strong performer for this classification task.

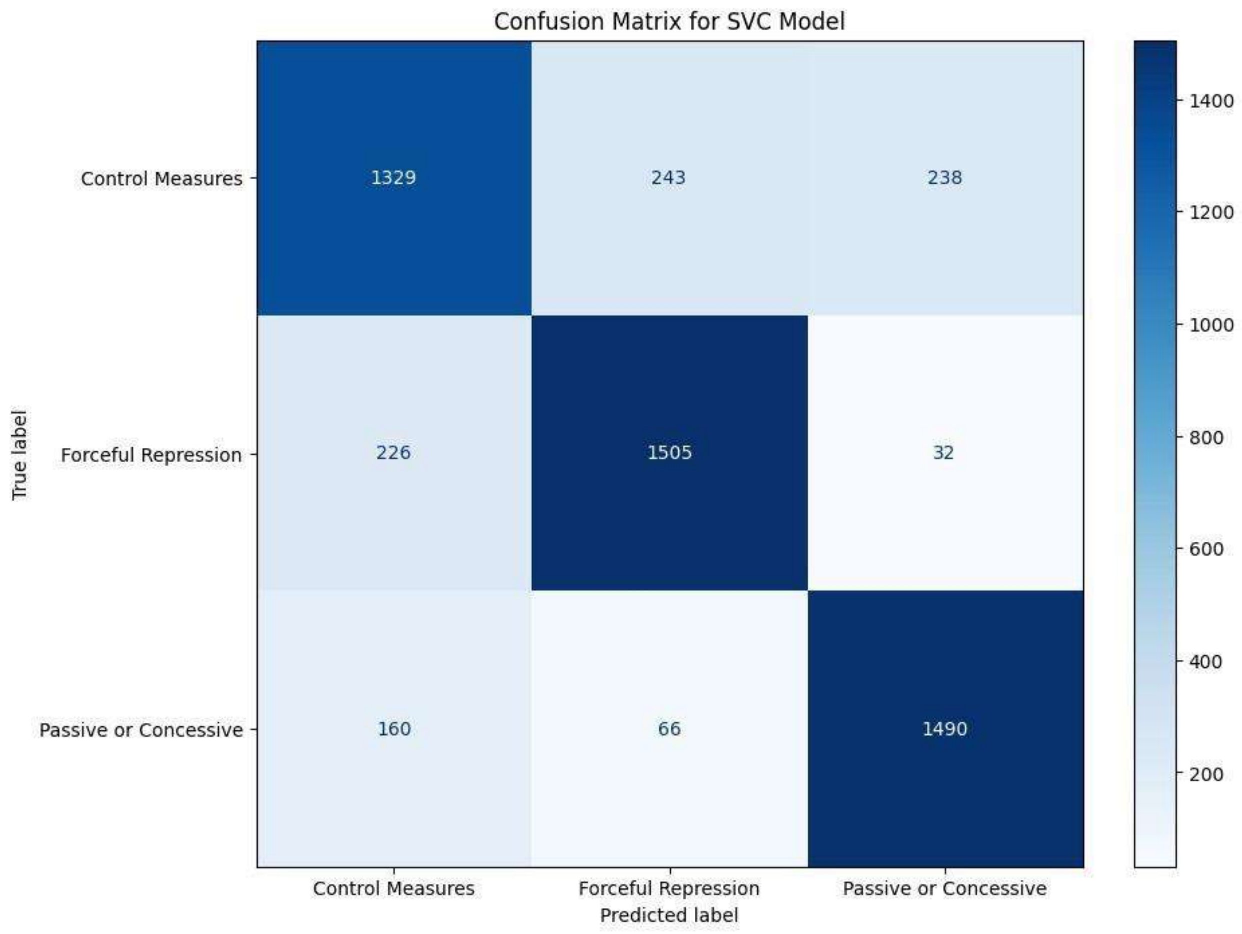
4.5.6 SVC (ROC, PR AUC Curves)

```
# Binarize the output for multi-class ROC and Precision-Recall
y_test_bin = label_binarize(y_test, classes=np.unique(y_test))

# Plot ROC and Precision-Recall curves for each class

n_classes = y_test_bin.shape[1]

plt.figure(figsize=(12, 12))

# Plot ROC curve for each class plt.subplot(2, 1, 1)
for i in range(n_classes):     fpr_test, tpr_test, _ =
roc_curve(y_test_bin[:, i],
svc_model.predict_proba(X_test)[:, i])
    roc_auc_test = roc_auc_score(y_test_bin[:, i],
svc_model.predict_proba(X_test)[:, i])
    plt.plot(fpr_test, tpr_test, lw=2, label=f'Test ROC curve of class
{i} (area = {roc_auc_test:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for Multi-Class Classification (SVC Test Set)')
plt.legend(loc="lower right")

# Plot Precision-Recall curve for each class plt.subplot(2, 1, 2)
for i in range(n_classes):     precision, recall, _ =
precision_recall_curve(y_test_bin[:, i],
svc_model.predict_proba(X_test)[:, i])
    pr_auc = average_precision_score(y_test_bin[:, i],
svc_model.predict_proba(X_test)[:, i])
    plt.plot(recall, precision, lw=2, label=f'Test PR curve of class
{i} (area = {pr_auc:.2f})')

plt.plot([0, 1], [0, 0], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0]) plt.ylim([0.0, 1.05])
plt.xlabel('Recall') plt.ylabel('Precision')
plt.title('Precision-Recall Curves for Multi-Class Classification (SVC
Test Set)')
plt.legend(loc="lower left")

plt.tight_layout()
plt.show()
```





```
# Compute and plot confusion matrix
conf_matrix = confusion_matrix(y_test, y_test_pred_svc) fig, ax =
plt.subplots(figsize=(10, 8))  # Increase the size here disp =
ConfusionMatrixDisplay(confusion_matrix=conf_matrix,
display_labels=np.unique(y_test)) disp.plot(ax=ax,
cmap=plt.cm.Blues) plt.title('Confusion Matrix for SVC Model')
plt.show()
```



Summary

The SVC model shows strong discriminative power with high AUC values across all classes.

- **Control Measures:** AUC = 0.88
- **Forceful Repression:** AUC = 0.95
- **Passive or Concessive:** AUC = 0.95

Explanation of Visualizations

ROC and PR Curves

1. **ROC Curves (Top):**
   - The ROC curves illustrate the trade-off between the true positive rate (TPR) and false positive rate (FPR) for each class.
   - The AUC (Area Under the Curve) values reflect the model's discriminative power.
     - **Control Measures** (Class 0): AUC = 0.88 (Blue Line)
     - **Forceful Repression** (Class 1): AUC = 0.95 (Orange Line)
     - **Passive or Concessive** (Class 2): AUC = 0.95 (Green Line)
   - Higher AUC values indicate better performance of the model in distinguishing between classes.

2. **PR Curves (Bottom):**
   - The PR (Precision-Recall) curves show the precision against recall for each class, useful for evaluating models on imbalanced datasets.
   - High AUC values in PR curves indicate high precision and recall, meaning the model can identify relevant instances effectively.
     - **Control Measures** (Class 0): AUC = 0.79 (Blue Line)
     - **Forceful Repression** (Class 1): AUC = 0.88 (Orange Line)
     - **Passive or Concessive** (Class 2): AUC = 0.89 (Green Line)

Confusion Matrix

The confusion matrix provides a detailed breakdown of the model's classification performance by showing the actual versus predicted labels.

- **True Positives (Diagonal Elements):** Correctly predicted instances.
  - **Control Measures:** 1329
  - **Forceful Repression:** 1505
  - **Passive or Concessive:** 1490
- **False Positives and False Negatives (Off-Diagonal Elements):** Instances where the model misclassified the labels.
  - Misclassified as **Control Measures:** 243 (from Forceful Repression), 238 (from Passive or Concessive)
  - Misclassified as **Forceful Repression:** 226 (from Control Measures), 32 (from Passive or Concessive)
  - Misclassified as **Passive or Concessive:** 160 (from Control Measures), 66 (from Forceful Repression)

Conclusion

The SVC model demonstrates robust classification performance with high accuracy and an excellent ability to distinguish between classes. The high AUC values across ROC and PR curves indicate the model's strong discriminative power, and the confusion matrix reflects a high number of correct predictions with relatively few misclassifications. This analysis underscores the model's effectiveness in classifying instances across the given categories.

Best Performing Model.

- XGBoost is recommended as the better model based on the following points:
- It has higher ROC AUC scores, indicating better performance in distinguishing between classes.
- It has higher Precision-Recall AUC scores, which is crucial for handling imbalanced datasets.
- Though the overall accuracy is slightly lower than SVC, the more robust ROC and PR curves suggest better generalization and reliability.

Despite SVC having a marginally higher test accuracy, XGBoost's superior ROC and PR metrics make it the preferred choice for this classification task.

```
joblib.dump(label_encoder, 'label_encoder_region.pkl')
joblib.dump(scaler, 'scaler.pkl')
joblib.dump(le, 'target_encoder.pkl') joblib.dump(xgb_clf,
'xgboost_model.pkl')

['xgboost_model.pkl']
```

5.0 RECOMMENDATIONS

In relation to the project's goals of analyzing global protest events, here are the recommendations based on the performance of our models:

1. Adopt XGBoost for Predictive Analysis:

   Why: XGBoost has shown the most reliable performance in accurately classifying and predicting protest events based on various factors.

   Benefit: This ensures that our analysis of protest events is based on robust and precise data, leading to more accurate identification of key factors leading to mass protests.

2. Regular Validation and Updates:

   Why: To maintain the accuracy and relevance of our analysis, it's essential to regularly validate and update our model with new data.

   Benefit: This ongoing validation will help us keep our insights current and reflective of the latest trends, ensuring our recommendations remain actionable and effective.

3. Focus on Feature Importance:

   Why: Understanding which factors are most influential in predicting protests will guide our data collection and analysis efforts.

   Benefit: By concentrating on the most significant features, we can derive deeper insights into the causes of protests and the effectiveness of state responses, informing more precise policy recommendations.

Next Steps

Optimize the Model for Better Insights:

```
What: Fine-tune the XGBoost model to enhance its predictive accuracy
and reliability.

Benefit: This optimization will provide us with sharper insights into
the factors driving protests, improving the quality of our analysis
and recommendations.
```

Integrate and Test the Model with Real-World Data:

```
What: Implement the XGBoost model into our analysis framework and test
it with actual protest event data.

Benefit: Ensuring the model works effectively in practice will
validate our analytical approach and enhance the credibility of our
findings.
```

Monitor Model Performance Continuously:

```
What: Establish a system to regularly monitor the model’s performance
and update it as necessary.

Benefit: Continuous monitoring will help us quickly identify and
rectify any performance issues, maintaining the model’s accuracy over
time.
```

Document Findings and Train the Team:

```
What: Thoroughly document our analysis process and train the team on
how to utilize and maintain the model.

Benefit: Proper documentation and training will ensure consistency in
our analysis efforts and facilitate knowledge
transfer within the team.
```

Conclusion

By adopting the XGBoost model and following these recommendations, we can achieve a comprehensive and accurate analysis of global protest events. This will enable us to:

Identify Key Factors:

```
Gain a clear understanding of the underlying causes of mass protests,
providing valuable insights for
policy development.
```

Analyze Trends:

```
Examine geographical and temporal trends in protests from 1990 to
March 2020, identifying patterns that can inform
future strategies.
```

Understand Characteristics:

```
Analyze the scale and intensity of protests, which is crucial for
anticipating and managing
```

```
socio-political unrest.
```

Evaluate State Responses:

```
Assess the impact and effectiveness of government responses, offering
evidence-based recommendations for improvement.
```

Offer Actionable Insights:

```
Provide informed, actionable policy recommendations aimed at improving
state-citizen relations and managin
social unrest   effectively.
```

4.0 MODELLING

Cleaned data ready for sentiment analysis and other NLP modelling techniques

- After our initial EDA and data preprocessing, we've maintained 88% of the original data resulting in a shape of (15076, 13). We will proceed with this new clean dataset for further analysis

```
#import necessary libraries
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
import string

#Load the data
data = pd.read_csv('mass_mobilization_cleaned.csv')
```

Inspect the data in our feature column

```
data.head()

          region  country  year  start_date    end_date  
protest_duration  \
0  North America   Canada  1990  1990-01-15  1990-01-15  
0
1  North America   Canada  1990  1990-06-25  1990-06-25  
0
2  North America   Canada  1990  1990-07-01  1990-07-01  
0
3  North America   Canada  1990  1990-07-12  1990-09-06  
56
4  North America   Canada  1990  1990-08-14  1990-08-15  
1

   participants_numeric  protesterviolence           protesteridentity  
\
0                  5000                  0                 unspecified  

1                  1000                  0                 unspecified  

2                   500                  0  separatist parti quebecois  

3                   500                  1              mohawk indians  

4                   950                  1             local residents  


   demand_labor wage dispute  ...  demand_tax policy
```

```
response_accomodation  \
0                        1  ...                       0
0
1                        0  ...                       0
0
2                        0  ...                       0
0
3                        0  ...                       0
1
4                        0  ...                       0
1

   response_arrests  response_beatings  response_crowd dispersal  \
0                 0                  0                         0
1                 0                  0                         0
2                 0                  0                         0
3                 0                  0                         0
4                 1                  0                         1

   response_ignore  response_killings  response_shootings  \
0                1                  0                   0
1                1                  0                   0
2                1                  0                   0
3                0                  0                   0
4                0                  0                   0

                                             sources  \
0  1. great canadian train journeys into history;...
1  1. autonomy s cry revived in quebec the new yo...
2  1. quebec protest after queen calls for unity ...
3  1. indians gather as siege intensifies; armed ...
4  1. dozens hurt in mohawk blockade protest the ...

                                               notes
0  canada s railway passenger system was finally ...
1  protestors were only identified as young peopl...
2  the queen, after calling on canadians to remai...
3  canada s federal government has agreed to acqu...
4  protests were directed against the state due t...

[5 rows x 27 columns]

# Check for missing values in 'notes' column
data['notes'].isnull().sum()

0
```

No null values on our target column 'notes'

```
# Preview the 'notes' column
data['notes'].head()

0    canada s railway passenger system was finally ...
1    protestors were only identified as young peopl...
2    the queen, after calling on canadians to remai...
3    canada s federal government has agreed to acqu...
4    protests were directed against the state due t...
Name: notes, dtype: object
```

Remove duplicates

Ensuring that there are no duplicates in the 'notes' column

```
# Remove duplicate rows based on 'notes'
data = data.drop_duplicates(subset='notes')
data

                  region           country  year  start_date
end_date  \
0      North America            Canada  1990  1990-01-15  1990-01-15

1      North America            Canada  1990  1990-06-25  1990-06-25

2      North America            Canada  1990  1990-07-01  1990-07-01

3      North America            Canada  1990  1990-07-12  1990-09-06

4      North America            Canada  1990  1990-08-14  1990-08-15

...              ...               ...   ...         ...         ...

15071        Oceania  Papua New Guinea  2014  2014-02-16  2014-02-18

15072        Oceania  Papua New Guinea  2016  2016-05-15  2016-06-09

15073        Oceania  Papua New Guinea  2017  2017-06-15  2017-06-15

15074        Oceania  Papua New Guinea  2017  2017-07-15  2017-07-15

15075        Oceania  Papua New Guinea  2017  2017-10-31  2017-10-31


       protest_duration  participants_numeric  protesterviolence  \
0                     0                  5000                  0
1                     0                  1000                  0
2                     0                   500                  0
3                    56                   500                  1
4                     1                   950                  1
...                 ...                   ...                ...
15071                 2                   100                  1
```

```
15072                    25                     1000                      1
15073                     0                       50                      0
15074                     0                       50                      1
15075                     0                      100                      0

                                       protesteridentity  \
0                                            unspecified
1                                            unspecified
2                             separatist parti quebecois
3                                         mohawk indians
4                                        local residents
...                                                  ...
15071                                     asylum seekers
15072                                university students
15073  protesters opposed to renewing the licence of ...
15074  protesters opposed to counting irregularities ...
15075                                             locals

       demand_labor wage dispute  ...  demand_tax policy  \
0                              1  ...                  0
1                              0  ...                  0
2                              0  ...                  0
3                              0  ...                  0
4                              0  ...                  0
...                          ...  ...                ...
15071                          0  ...                  0
15072                          0  ...                  0
15073                          0  ...                  0
15074                          0  ...                  0
15075                          0  ...                  0

       response_accomodation  response_arrests  response_beatings  \
0                          0                 0                  0
1                          0                 0                  0
2                          0                 0                  0
3                          1                 0                  0
4                          1                 1                  0
...                      ...               ...                ...
15071                      0                 0                  0
15072                      0                 0                  0
15073                      1                 0                  0
15074                      0                 0                  0
15075                      0                 0                  0

       response_crowd dispersal  response_ignore  response_killings  \
0                             0                1                  0
1                             0                1                  0
2                             0                1                  0
3                             0                0                  0
4                             1                0                  0
```

```
...                        ...              ...                ...
15071                        1                0                  0
15072                        1                0                  1
15073                        0                0                  0
15074                        1                0                  0
15075                        0                1                  0

       response_shootings
sources  \
0                       0  1. great canadian train journeys into
history;...
1                       0  1. autonomy s cry revived in quebec the new
yo...
2                       0  1. quebec protest after queen calls for
unity ...
3                       0  1. indians gather as siege intensifies;
armed ...
4                       0  1. dozens hurt in mohawk blockade protest
the ...
...                   ...
...
15071                   1  probe into killing of manus detainee; manus
is...
15072                   1  papua new guinea: reports of up to four
people...
15073                   0  bougainville imposes moratorium on panguna
min...
15074                   0  violence, chaos and fraud: fraught papua
new g...
15075                   0  refugees dig in as camp closes; manus
situatio...

                                                   notes
0      canada s railway passenger system was finally ...
1      protestors were only identified as young peopl...
2      the queen, after calling on canadians to remai...
3      canada s federal government has agreed to acqu...
4      protests were directed against the state due t...
...                                                  ...
15071  ? a government inquiry will be launched as ser...
15072  police in papua new guinea fired gunshots wedn...
15073  the bougainville government has enacted an ind...
15074  peter o neill has been reappointed as prime mi...
15075  refugees on manus island were braced for poten...

[13816 rows x 27 columns]
```

Working on a dataframe for futher sentiment analysis

Our target feature for sentiment analysis in our dataset notes columns, therefore w have to drop all other columns

```
# Drop all columns except 'notes'
data_notes = data.drop(columns=[col for col in data.columns if col !=
'notes'])

# Check the resulting DataFrame
data_notes.head()

                                                notes
0  canada s railway passenger system was finally ...
1  protestors were only identified as young peopl...
2  the queen, after calling on canadians to remai...
3  canada s federal government has agreed to acqu...
4  protests were directed against the state due t...
```

4.1 NATURAL PROCESSING LANGUAGE

- For NLP preprocessing and text data cleaning, we'll eliminate stopwords, punctuation, and numbers, and convert text to lowercase in the notes columns.
- Subsequently, tokenizing our data is essential because it breaks down text into individual words or tokens, enabling deeper analysis and understanding of the textual content. Preprocess the text data to clean it up and prepare it for analysis

4.1.1 SENTIMENT ANALYSIS

```
import re
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt


# Initialize lemmatizer
lemmatizer = WordNetLemmatizer()

# Define a function for text preprocessing with lemmatization
def preprocess_text(text):
    if isinstance(text, str):  # Check if text is a string
        text = text.lower()  # Convert to lowercase
        text = re.sub(r'\d+', '', text)  # Remove digits
        text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
        tokens = word_tokenize(text)  # Tokenize text
        tokens = [lemmatizer.lemmatize(word) for word in tokens]  #
Lemmatize tokens
        tokens = [word for word in tokens if word not in
stopwords.words('english')]  # Remove stopwords
        text = ' '.join(tokens)  # Join tokens back into a single
string
        return text.strip()  # Remove leading and trailing whitespace
    else:
        return ""  # Return an empty string if text is not a string
```

Creating a copy of our cleaned data for the column notes

```
data_notes_copy = data_notes.copy()

from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

def get_sentiment_nltk(text):
    sentiment = sid.polarity_scores(text)
    return sentiment['compound']

# Apply preprocessing
data_notes['cleaned_notes'] =
data_notes['notes'].apply(preprocess_text)

# Now apply sentiment analysis
data_notes['compound_sentiment'] =
data_notes['cleaned_notes'].apply(get_sentiment_nltk)

[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

We will get new sentiment scores by performing sentiment analysis on the cleaned_notes column of a DataFrame using the VADER sentiment analyzer from the nltk library.

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk

# Initialize VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Define a function to get sentiment scores
def get_sentiment(text):
    scores = sid.polarity_scores(text)
    return pd.Series([scores['compound'], scores['pos'],
scores['neu'], scores['neg']])

# Apply sentiment analysis to the 'cleaned_notes' column
data_notes[['compound', 'positive', 'neutral', 'negative']] =
data_notes['cleaned_notes'].apply(get_sentiment)

# Display sentiment analysis results
data_notes[['cleaned_notes', 'compound', 'positive', 'neutral',
'negative']]
data_notes

                                                    notes  \
0      canada s railway passenger system was finally ...
1      protestors were only identified as young peopl...
2      the queen, after calling on canadians to remai...
3      canada s federal government has agreed to acqu...
4      protests were directed against the state due t...
...                                                   ...
15071  ? a government inquiry will be launched as ser...
15072  police in papua new guinea fired gunshots wedn...
15073  the bougainville government has enacted an ind...
15074  peter o neill has been reappointed as prime mi...
15075  refugees on manus island were braced for poten...

                                            cleaned_notes
compound_sentiment  \
0      canada railway passenger system wa finally cut...             -
0.8316
1      protestors identified young people gathering w...
0.0000
2      queen calling canadian remain united braved pr...
0.8720
3      canada federal government ha agreed acquire tr...             -
0.8481
4      protest directed state due refusal use violenc...             -
0.5423
...                                                   ...
...
15071  government inquiry launched serious question r...             -
0.9920
15072  police papua new guinea fired gunshot wednesda...             -
0.9943
15073  bougainville government ha enacted indefinite ...             -
0.8519
15074  peter neill ha reappointed prime minister papu...             -
0.9561
15075  refugee manus island braced potential calamity...             -
0.9816

       compound  positive  neutral  negative
0       -0.8316     0.000    0.866     0.134
1        0.0000     0.000    1.000     0.000
2        0.8720     0.203    0.695     0.103
3       -0.8481     0.103    0.736     0.160
```

```
4       -0.5423     0.139    0.535     0.326
...         ...       ...      ...       ...
15071   -0.9920     0.058    0.603     0.339
15072   -0.9943     0.046    0.697     0.257
15073   -0.8519     0.061    0.787     0.152
15074   -0.9561     0.075    0.753     0.173
15075   -0.9816     0.069    0.674     0.258

[13816 rows x 7 columns]
```

Sentiment Distribution:

Negative Sentiment: Many of the notes have a negative compound sentiment score (e.g., -0.8316, -0.9943). This suggests that a significant portion of the notes reflects negative sentiments about the protests or related events.

Positive Sentiment: A few notes show positive sentiment scores (e.g., 0.8720). These are less frequent compared to the negative sentiments.

Neutral Sentiment: The neutral proportion can help understand if any notes are perceived as neutral despite the overall sentiment. Sample Notes Analysis:

Notes with High Negative Sentiment:

For example, notes like "canada s railway passenger system was finally…" with a compound score of -0.8316 suggest strong negative feelings or dissatisfaction.

Similarly, notes like "police in papua new guinea fired gunshots…" with -0.9943 indicate a high level of negative sentiment related to police actions.

Notes with High Positive Sentiment:

For example, "the queen, after calling on canadians to remain united…" with a compound score of 0.8720 indicates a positive sentiment, reflecting support or approval.

Proportions Analysis:

Positive Proportions: Low proportions of positive sentiment (e.g., 0.203) indicate that positive sentiments are not very common in the dataset.

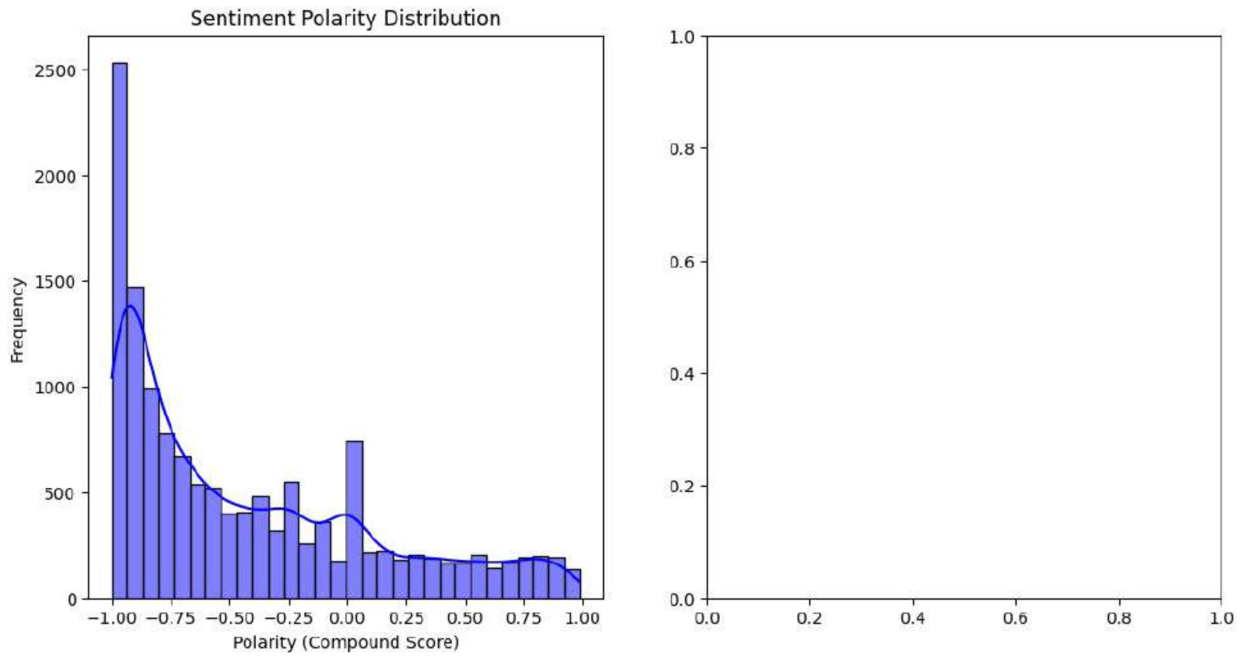
Neutral Proportions: Neutral proportions (e.g., 0.695) suggest that many notes are perceived as neutral, which may imply they are reporting factual information without strong sentiment.

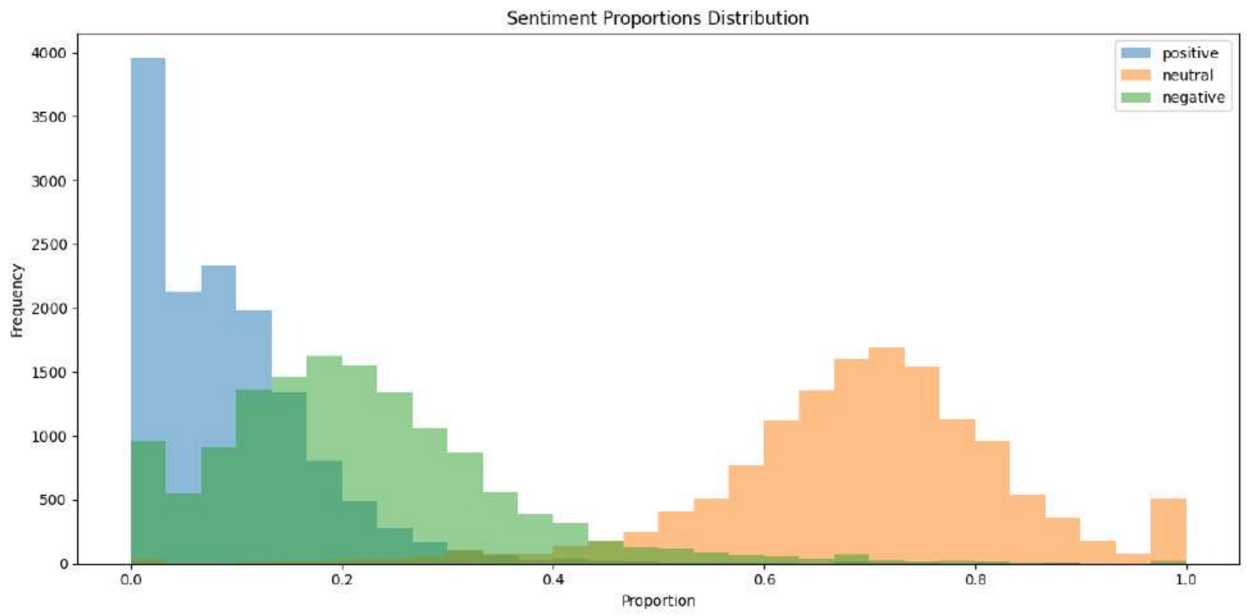
Negative Proportions: High negative proportions (e.g., 0.736) highlight a trend toward negative sentiments in the dataset.

Visualizing Sentiment distribution

This visualization represents the frequency distribution of sentiment proportions (positive, neutral, and negative) in the dataset.

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the sentiment polarity distribution (VADER's compound score)
plt.figure(figsize=(12, 6))

# Plot polarity
plt.subplot(1, 2, 1)
sns.histplot(data_notes['compound'], bins=30, kde=True, color='blue')
plt.title('Sentiment Polarity Distribution')
plt.xlabel('Polarity (Compound Score)')
plt.ylabel('Frequency')

# Plot positive, neutral, and negative sentiment proportions
plt.subplot(1, 2, 2)
data_notes[['positive', 'neutral', 'negative']].plot(kind='hist',
bins=30, alpha=0.5, histtype='bar', figsize=(12, 6))
plt.title('Sentiment Proportions Distribution')
plt.xlabel('Proportion')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```





Key Observations

Distribution Shapes:

Positive Sentiment:

The distribution for positive sentiment is skewed to the right, indicating a higher frequency of texts with a smaller proportion of positive sentiment. There's a long tail towards higher proportions, suggesting a smaller number of texts with predominantly positive sentiment.

Neutral Sentiment:

The neutral sentiment distribution is relatively symmetrical, with a peak around the middle, suggesting a balanced distribution of texts with neutral sentiment proportions.

Negative Sentiment:

Similar to positive sentiment, the negative sentiment distribution is skewed to the right, indicating a higher frequency of texts with a smaller proportion of negative sentiment.

Overlapping Distributions:

The distributions for all three sentiment categories overlap significantly, indicating that many texts exhibit a mix of sentiments rather than being purely positive, neutral, or negative.

Interpretation in Relation to the Dataset

Mixed Sentiments:

The majority of the texts in the dataset likely contain a mix of positive, negative, and neutral sentiments, as evidenced by the overlapping distributions.

Dominant Neutral Sentiment:

The relatively symmetrical distribution of neutral sentiment suggests that a significant portion of the texts expresses a neutral viewpoint.

Less Extreme Sentiments:

The rightward skew of both positive and negative sentiment distributions indicates that most texts lean towards a more neutral stance rather than expressing strong positive or negative emotions.

Potential Insights

This visualization highlights the complexity of sentiment analysis, as many texts exhibit nuanced and mixed emotional expressions.

4.1.2 Textual Coherencing using Latent Dirichlet Allocation (LDA)

```python
import re
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import gensim
from gensim import corpora
from gensim.models import CoherenceModel
import matplotlib.pyplot as plt
import seaborn as sns

# Tokenize text for LDA
tokenized_notes = [preprocess_text(note).split() for note in
data_notes['cleaned_notes']]

# Create dictionary and corpus for LDA
dictionary = corpora.Dictionary(tokenized_notes)
corpus = [dictionary.doc2bow(text) for text in tokenized_notes]

# Create LDA model
lda_model = gensim.models.ldamodel.LdaModel(corpus, num_topics=5,
id2word=dictionary, passes=15)

# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=lda_model,
texts=tokenized_notes, dictionary=dictionary, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print(f'Coherence Score: {coherence_lda}')
```

```
Coherence Score: 0.36928316333084765
```

Intepretation of textual coherence score

A coherence score of 0.3692

This indicates the degree to which the words in each topic are semantically related. Generally, coherence scores range from 0 to 1, with higher values suggesting that the words within a topic are more closely related.

Interpreting Coherence Scores

Low Coherence Score (< 0.3):

Indicates that the topics may not be very interpretable or meaningful. The words in the topics might not co-occur in a way that makes sense.

Moderate Coherence Score (0.3 - 0.5):

Suggests that the topics are somewhat interpretable, but there may be room for improvement. Some topics may be more meaningful than others.

High Coherence Score (> 0.5):

Indicates that the topics are more interpretable and the words within each topic are highly related.

```
from gensim.models import LdaModel
from gensim.corpora import Dictionary
from gensim.models.coherencemodel import CoherenceModel

# Prepare your corpus and dictionary
texts = data_notes['cleaned_notes'].apply(lambda x: x.split())
dictionary = Dictionary(texts)
corpus = [dictionary.doc2bow(text) for text in texts]

# Test different numbers of topics
for num_topics in range(5, 15, 2):
    lda_model = LdaModel(corpus, num_topics=num_topics,
id2word=dictionary, passes=15)
    coherence_model = CoherenceModel(model=lda_model, texts=texts,
dictionary=dictionary, coherence='c_v')
    coherence_score = coherence_model.get_coherence()
    print(f'Number of Topics: {num_topics}, Coherence Score:
{coherence_score}')

Number of Topics: 5, Coherence Score: 0.3937524888802811
Number of Topics: 7, Coherence Score: 0.41678578321054205
Number of Topics: 9, Coherence Score: 0.4276075331199691
Number of Topics: 11, Coherence Score: 0.42304576648646697
Number of Topics: 13, Coherence Score: 0.4987496769992307
```

The coherence scores obtained indicate that the topic model's quality improves as the number of topics increases, with the highest score observed at 13 topics.

```
for num_topics in range(15, 25, 2):
    lda_model = LdaModel(corpus, num_topics=num_topics,
id2word=dictionary, passes=15)
    coherence_model = CoherenceModel(model=lda_model, texts=texts,
dictionary=dictionary, coherence='c_v')
    coherence_score = coherence_model.get_coherence()
    print(f'Number of Topics: {num_topics}, Coherence Score:
{coherence_score}')

Number of Topics: 15, Coherence Score: 0.46794030784001683
Number of Topics: 17, Coherence Score: 0.46380943736401903
Number of Topics: 19, Coherence Score: 0.4285231799935653
Number of Topics: 21, Coherence Score: 0.4568459430544956
Number of Topics: 23, Coherence Score: 0.44087473988551656
```

The coherence scores we have obtained indicate that the model with 15 topics has the highest coherence score of 0.4679, which is quite good.

Let's review the topics generated by the 15-topic model to ensure they align with our expectations and make sense in the context of the dataset.

```
# Review the 15 topic model
lda_model_15 = LdaModel(corpus, num_topics=15, id2word=dictionary,
passes=30, alpha='auto', eta='auto')
topics = lda_model_15.print_topics(num_words=10)
for topic in topics:
    print(topic)

(0, '0.056*"police" + 0.018*"protester" + 0.013*"officer" +
0.012*"riot" + 0.011*"city" + 0.010*"road" + 0.009*"town" +
0.008*"hundred" + 0.008*"yesterday" + 0.007*"wa"')
(1, '0.028*"government" + 0.026*"worker" + 0.025*"strike" +
0.024*"union" + 0.024*"protest" + 0.011*"farmer" + 0.010*"demand" +
0.010*"yesterday" + 0.009*"price" + 0.008*"public"')
(2, '0.138*"woman" + 0.028*"child" + 0.019*"rape" + 0.017*"girl" +
0.016*"court" + 0.014*"right" + 0.014*"case" + 0.013*"men" +
0.012*"abortion" + 0.010*"murder"')
(3, '0.273*"student" + 0.103*"university" + 0.074*"school" +
0.063*"teacher" + 0.037*"education" + 0.026*"campus" + 0.015*"college"
+ 0.014*"fee" + 0.013*"nuclear" + 0.011*"parent"')
(4, '0.026*"fisherman" + 0.020*"reuters" + 0.020*"fighter" +
0.019*"ex" + 0.012*"fishing" + 0.012*"fish" + 0.011*"rice" +
0.011*"monument" + 0.010*"beach" + 0.010*"tribe"')
(5, '0.075*"police" + 0.039*"gas" + 0.036*"tear" + 0.027*"fired" +
0.025*"injured" + 0.024*"said" + 0.021*"clash" + 0.018*"killed" +
0.017*"disperse" + 0.016*"least"')
(6, '0.031*"gov" + 0.026*"parade" + 0.025*"magistrate" + 0.023*"junta"
+ 0.015*"aged" + 0.011*"combatant" + 0.010*"enforce" + 0.009*"sharing"
+ 0.009*"execution" + 0.009*"disappearance"')
(7, '0.099*"opposition" + 0.090*"party" + 0.065*"election" +
0.035*"supporter" + 0.024*"president" + 0.020*"vote" +
0.019*"presidential" + 0.019*"ruling" + 0.018*"democratic" +
0.015*"court"')
(8, '0.015*"would" + 0.014*"said" + 0.013*"yesterday" +
0.012*"parliament" + 0.012*"minister" + 0.011*"ha" + 0.010*"law" +
0.009*"government" + 0.009*"new" + 0.008*"wa"')
(9, '0.071*"based" + 0.060*"source" + 0.055*"coding" + 0.050*"date" +
0.048*"decision" + 0.047*"actual" + 0.037*"included" + 0.027*"number"
+ 0.025*"king" + 0.025*"provided"')
(10, '0.109*"muslim" + 0.033*"christian" + 0.030*"turkish" +
0.024*"iran" + 0.024*"trader" + 0.017*"citizenship" +
0.015*"macedonia" + 0.013*"import" + 0.013*"religious" +
0.011*"equal"')
(11, '0.081*"al" + 0.059*"prison" + 0.036*"prisoner" +
0.023*"sentence" + 0.018*"jail" + 0.016*"inmate" + 0.015*"sentenced" +
0.014*"vladimir" + 0.014*"immigrant" + 0.013*"convicted"')
(12, '0.040*"protest" + 0.025*"wa" + 0.022*"people" +
0.019*"government" + 0.016*"said" + 0.016*"protester" +
0.013*"demonstration" + 0.010*"president" + 0.010*"capital" +
0.009*"country"')
(13, '0.054*"thousand" + 0.027*"marched" + 0.026*"street" +
0.020*"march" + 0.020*"rally" + 0.017*"city" + 0.017*"square" +
0.016*"anti" + 0.015*"demonstrator" + 0.014*"ten"')
(14, '0.031*"iraq" + 0.019*"un" + 0.017*"drug" + 0.017*"cleric" +
0.013*"u" + 0.013*"american" + 0.012*"asylum" + 0.011*"mexico" +
0.009*"war" + 0.009*"embassy"')
```

Topic 0: Law Enforcement and Public Order

Focuses on police, officers, and riot-related terms, suggesting this topic is concerned with issues related to police actions, public order, and possibly protests or civil unrest.

Topic 1: Labor and Strikes

Includes terms related to government, workers, strikes, unions, and demands. This topic likely revolves around labor movements, strikes, and worker demands.

Topic 2: Gender and Violence

Centers on terms like woman, child, rape, and court. This topic is related to gender issues, violence against women, and legal cases involving gender-based violence.

Topic 3: Education

Focuses on students, universities, schools, and related educational terms. This topic likely deals with issues related to education, student protests, and educational institutions.

Topic 4: Fishing and Local Community

Includes terms like fisherman, fishing, and beach. This topic seems to address issues related to fishing communities and local economic activities related to fishing.

Topic 5: Police Violence and Conflict

Features terms related to police actions (gas, tear, fired, injured) and conflicts. This topic seems to cover incidents of police violence and confrontations during protests or riots.

Topic 6: Government and Authority

Contains terms like parade, junta, and magistrate. This topic might be related to government authority, military or political parades, and legal or judicial matters.

Topic 7: Political Opposition and Elections

Includes terms like opposition, party, and election. This topic is likely focused on political parties, elections, and political opposition.

Topic 8: Government and Legislative Matters

Features terms related to government, parliament, and ministers. This topic might cover legislative processes, government announcements, and political discussions.

Topic 9: Data and Decision Making

Contains terms related to data analysis (source, coding, decision) and possibly administrative or managerial aspects. This topic might focus on data handling and decision-making processes.

Topic 10: Religion and Citizenship

Focuses on religious and citizenship terms, including Muslim, Christian, and Turkish. This topic is likely concerned with issues related to religion, religious groups, and citizenship rights.

Topic 11: Criminal Justice

Includes terms related to prison, prisoner, and sentence. This topic likely addresses issues related to the criminal justice system, incarceration, and sentencing.

Topic 12: General Protests and Demonstrations

Contains terms related to protests, people, and government. This topic appears to cover general aspects of protests and demonstrations, including participant numbers and governmental responses.

Topic 13: Mass Mobilization

Features terms like thousand, marched, and rally. This topic likely focuses on large-scale protests or rallies and the organization of mass mobilization events.

Topic 14: International Affairs and Conflict

Includes terms related to international issues (Iraq, UN, asylum) and conflict. This topic seems to address global issues, international relations, and conflicts.

4.1.3 Assign topics to each document

```
# Initialize the VADER sentiment intensity analyzer
sid = SentimentIntensityAnalyzer()

# Function to get sentiment scores
```

```
def get_sentiment(text):
    sentiment = sid.polarity_scores(text)
    return sentiment['compound'], sentiment['neu'], sentiment['pos'],
sentiment['neg']

# Apply sentiment analysis to the 'cleaned_notes' column
data_notes[['sentiment_polarity', 'sentiment_neutral',
'sentiment_positive', 'sentiment_negative']] =
data_notes['cleaned_notes'].apply(lambda x:
pd.Series(get_sentiment(x)))
# Display sentiment analysis results
data_notes[['cleaned_notes', 'sentiment_polarity',
'sentiment_neutral', 'sentiment_positive', 'sentiment_negative']]
```

```
                                                cleaned_notes
sentiment_polarity  \
0      canada railway passenger system wa finally cut...              -
0.8316
1      protestors identified young people gathering w...
0.0000
2      queen calling canadian remain united braved pr...
0.8720
3      canada federal government ha agreed acquire tr...              -
0.8481
4      protest directed state due refusal use violenc...              -
0.5423
...                                                  ...
...
15071  government inquiry launched serious question r...              -
0.9920
15072  police papua new guinea fired gunshot wednesda...              -
0.9943
15073  bougainville government ha enacted indefinite ...              -
0.8519
15074  peter neill ha reappointed prime minister papu...              -
0.9561
15075  refugee manus island braced potential calamity...              -
0.9816

       sentiment_neutral  sentiment_positive  sentiment_negative
0                  0.866               0.000               0.134
1                  1.000               0.000               0.000
2                  0.695               0.203               0.103
3                  0.736               0.103               0.160
4                  0.535               0.139               0.326
...                  ...                 ...                 ...
15071              0.603               0.058               0.339
15072              0.697               0.046               0.257
15073              0.787               0.061               0.152
15074              0.753               0.075               0.173
```

```
15075                   0.674                   0.069                   0.258

[13816 rows x 5 columns]
```

Sentiment Polarity:

Indicates the overall sentiment of the text, ranging from -1 (very negative) to 1 (very positive). For example, a polarity of -0.8316 suggests a very negative sentiment, while 0.8720 suggests a very positive sentiment.

Sentiment Neutrality: Indicates the proportion of neutral sentiment in the text. A high value, like 1.000, means the text is neutral with no positive or negative sentiment.

Sentiment Positivity: Measures the proportion of positive sentiment in the text. A value of 0.203 indicates a low level of positivity.

Sentiment Negativity: Measures the proportion of negative sentiment in the text. A value of 0.134 indicates a moderate level of negativity.

```
from gensim.models import LdaModel

# Save the LDA model
lda_model.save('lda_model_15')

# Assign Topics to each Document
from gensim.models import LdaModel

# Load LDA model
lda_model = LdaModel.load('lda_model_15')

# Assign topics to each document
data_notes['topic'] = [max(lda_model[doc], key=lambda x: x[1])[0] for
doc in corpus]

# Extract Top words for each topic
num_words = 10  # Number of top words to display per topic

topics = lda_model.show_topics(num_topics=-1, num_words=num_words,
formatted=False)
top_words_per_topic = {topic[0]: [word[0] for word in topic[1]] for
topic in topics}

# Display top words for each topic
for topic_num, words in top_words_per_topic.items():
    print(f"Topic {topic_num}: {', '.join(words)}")

Topic 0: iraq, greece, italy, spain, bombing, u, spanish, prayer,
withdrawal, grand
Topic 1: woman, rape, violence, girl, turkey, men, child, abortion,
islamabad, gov
Topic 2: prison, prisoner, turkish, release, iran, chain, jail,
sentenced, inmate, jailed
Topic 3: farmer, protest, date, plant, unclear, russia, european,
miner, eu, german
Topic 4: health, hospital, pakistan, doctor, trader, medical, market,
nurse, drug, treatment
Topic 5: student, university, school, education, campus, protest,
teacher, college, high, class
Topic 6: said, police, people, wa, killed, protest, security, two,
arrested, least
Topic 7: based, decision, prior, number, source, actual, included,
provided, wa, protester
Topic 8: school, child, parent, pupil, councillor, asylum, asia, john,
tribal, seeker
Topic 9: military, soldier, army, rebel, minister, indian, base,
commander, civilian, force
Topic 10: protest, government, said, city, wa, street, ha, people,
demonstration, protester
Topic 11: wa, protest, right, people, group, ha, law, parliament,
protester, said
Topic 12: korea, funeral, primary, waste, port, mourner, mr, spread,
siege, monument
Topic 13: resident, council, note, office, land, yesterday, coding,
local, wa, petition
Topic 14: police, protester, riot, gas, tear, officer, fired,
demonstrator, crowd, disperse
Topic 15: rally, pro, activist, anti, democracy, independence,
yesterday, supporter, leader, th
Topic 16: paris, water, verdict, cold, changed, conduct, art, nelson,
bailout, courthouse
Topic 17: france, french, migrant, de, track, terrorism, cell, yellow,
homeless, brazil
Topic 18: worker, strike, union, government, protest, teacher, trade,
wage, demand, pay
Topic 19: camp, china, island, chinese, airport, refugee, flight,
separatist, jordan, uber
Topic 20: protest, article, size, clear, coded, estimated, th,
previous, participation, number
Topic 21: opposition, party, president, protest, government, election,
thousand, people, protester, capital
Topic 22: road, blocked, muslim, driver, traffic, highway, price, bus,
fuel, main
```

Topic 0: Likely related to geopolitical issues involving countries like Iraq, Greece, Italy, and Spain, possibly concerning military actions or interventions (e.g., "bombing," "withdrawal").

Topic 1: Focused on gender-based violence and issues concerning women and children, including rape, violence, and abortion.

Topic 2: Related to prisons and incarceration, possibly highlighting issues around prisoners, sentences, and prison conditions.

Topic 3: Likely related to protests involving farmers or other labor groups, possibly in the context of agricultural or labor disputes, with mentions of Russia and the EU.

Topic 4: Centered on healthcare issues, including hospitals, doctors, and medical treatments, with a focus on Pakistan.

Topic 5: Related to education, including student protests, universities, schools, and teachers.

Topic 6: Describes incidents involving police and public security, including protests where people were arrested or killed.

Topic 7: Appears to be about decision-making processes, possibly in a bureaucratic or governmental context, with mentions of sources and provided information.

Topic 8: Focused on issues concerning children, schools, and parents, possibly in the context of asylum and tribal issues.

Topic 9: Related to military actions and conflicts, with mentions of soldiers, armies, and commanders, possibly in an Indian context.

Topic 10: General protest topic involving government, cities, and street demonstrations.

Topic 11: A broader topic on rights, laws, and groups involved in protests, possibly focusing on legal and civil rights issues.

Topic 12: This topic might be more localized, focusing on community issues such as local governance (councils) and land-related protests.

Topic 13: Involves police actions during protests, including the use of tear gas and riot control measures.

Topic 14: Focused on rallies, often associated with activism, democracy movements, or independence struggles.

Topic 15: May be related to specific events in Paris or other cities, possibly involving public demonstrations and legal or artistic matters.

Topic 16: Focused on French-related issues, possibly involving migrants, terrorism, and social unrest.

Topic 17: Related to labor strikes and workers' protests, focusing on unions, wages, and demands for better conditions.

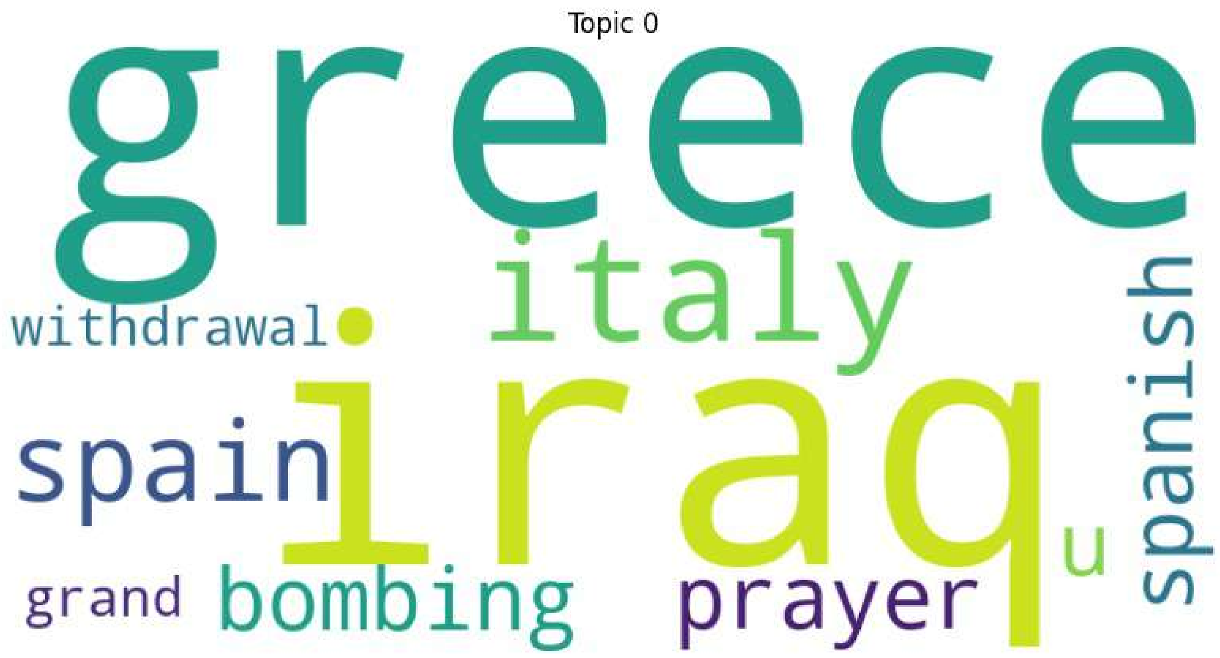
Topic 18: Involves international issues, possibly concerning refugee camps or conflicts involving China.

Topic 19: Focuses on articles or reports concerning protests, with an emphasis on participation and numbers.

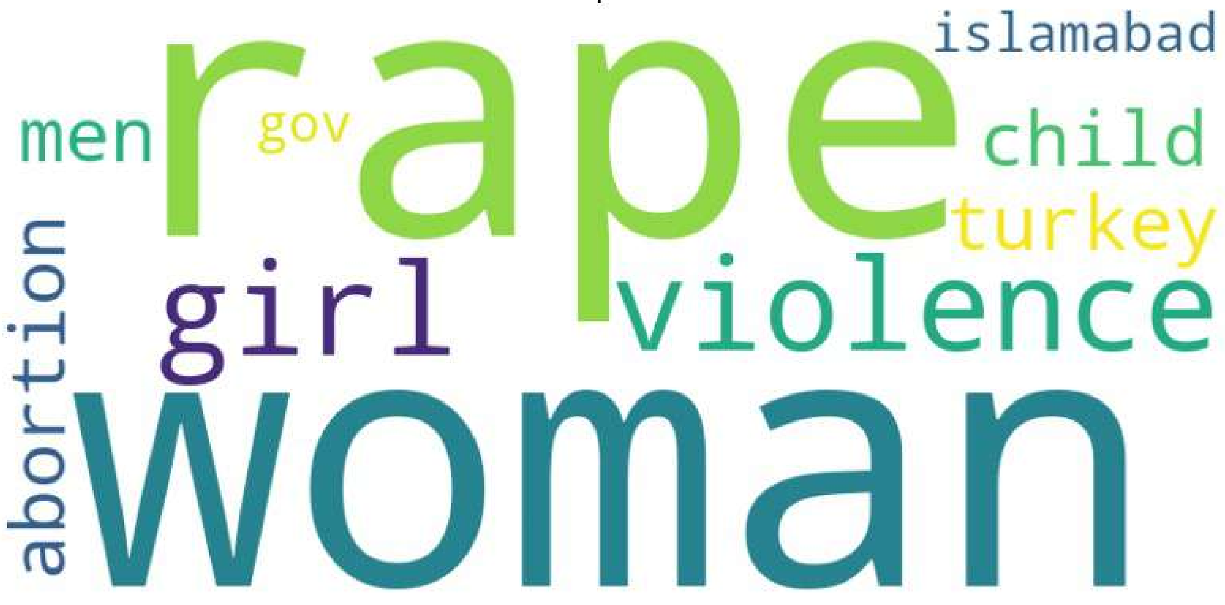
Topic 20: Centered on opposition parties, elections, and political protests, often involving large numbers of participants.

Topic 21: Focused on road blockages and traffic disruptions, possibly in the context of fuel prices or religious (Muslim) issues.

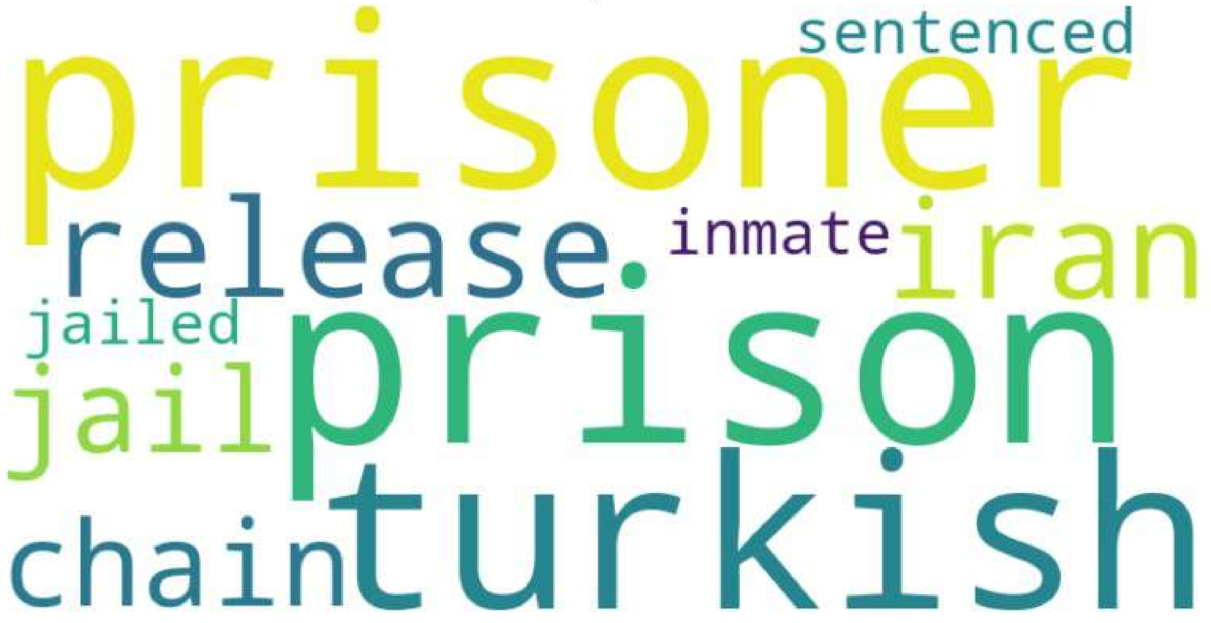
```python
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# Function to generate word cloud from top words
def generate_wordcloud(words):
    text = ' '.join(words)
    wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text)
    return wordcloud

# Plot word clouds for each topic
for topic_num, words in top_words_per_topic.items():
    plt.figure(figsize=(10, 5))
    wordcloud = generate_wordcloud(words)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(f"Topic {topic_num}")
    plt.show()
```



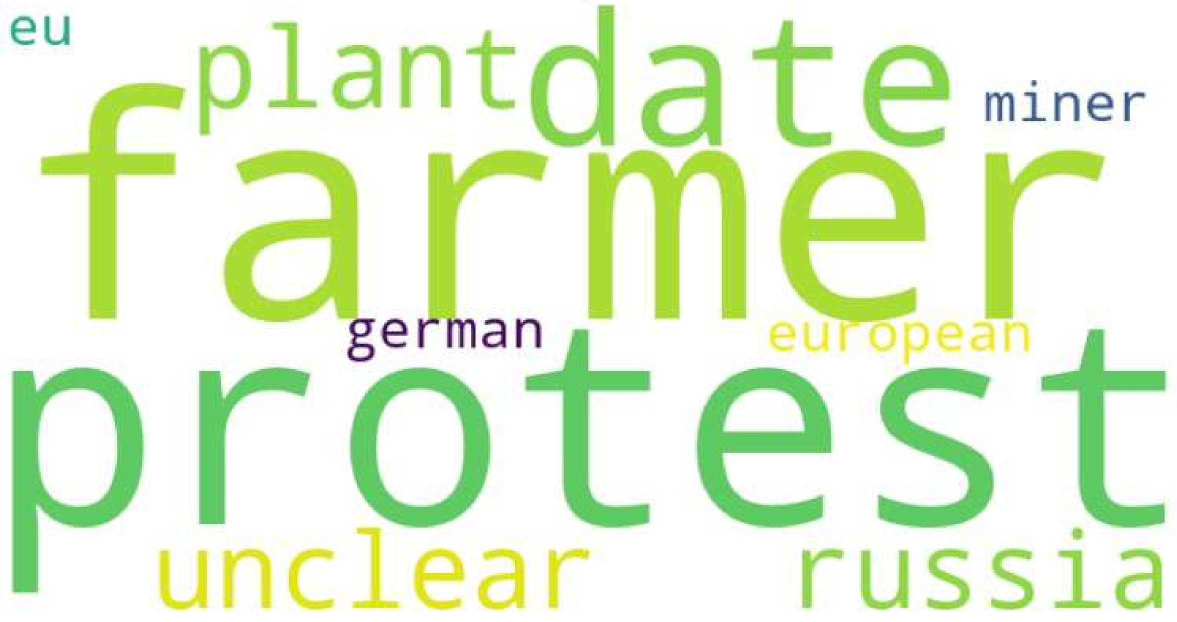
Topic 1



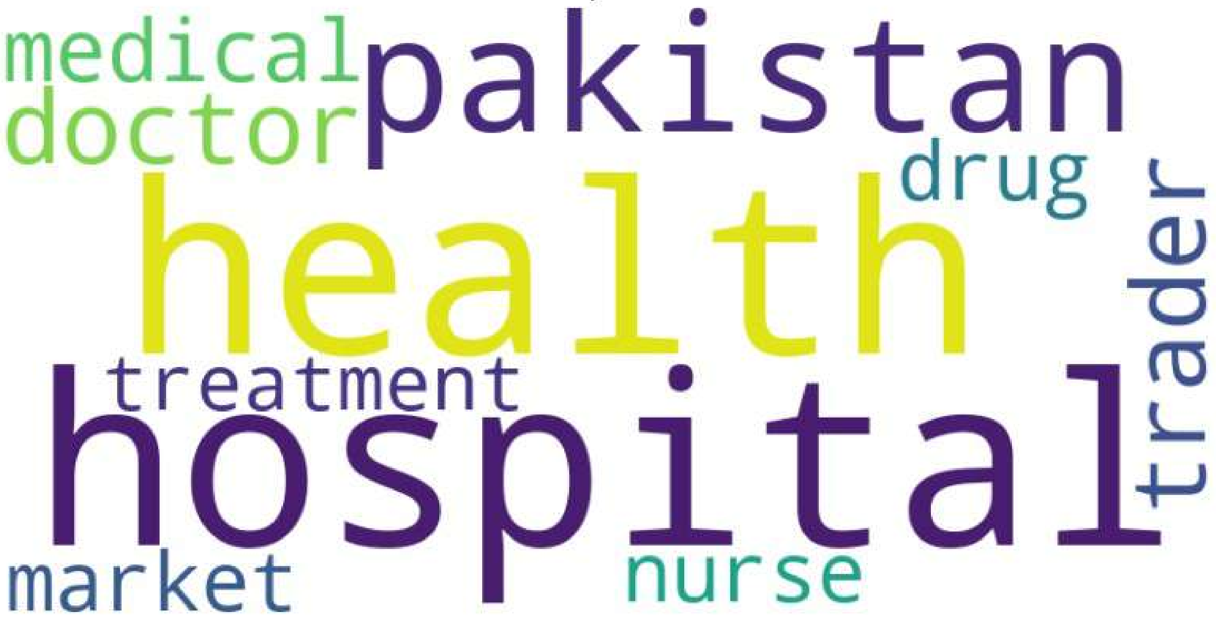
Topic 2



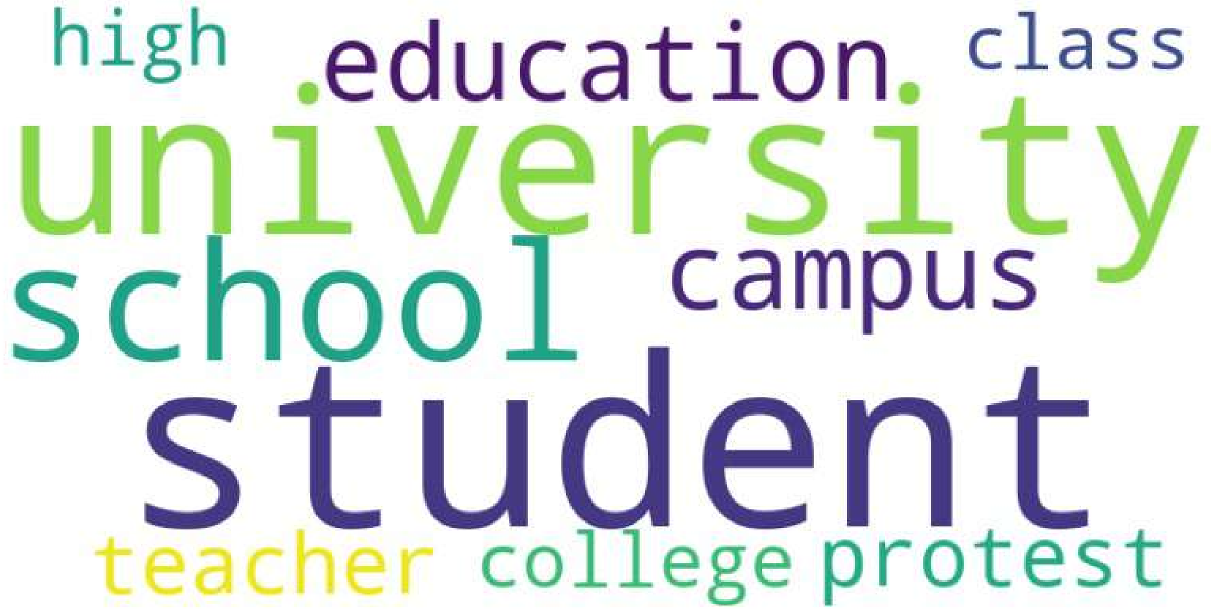
Topic 3



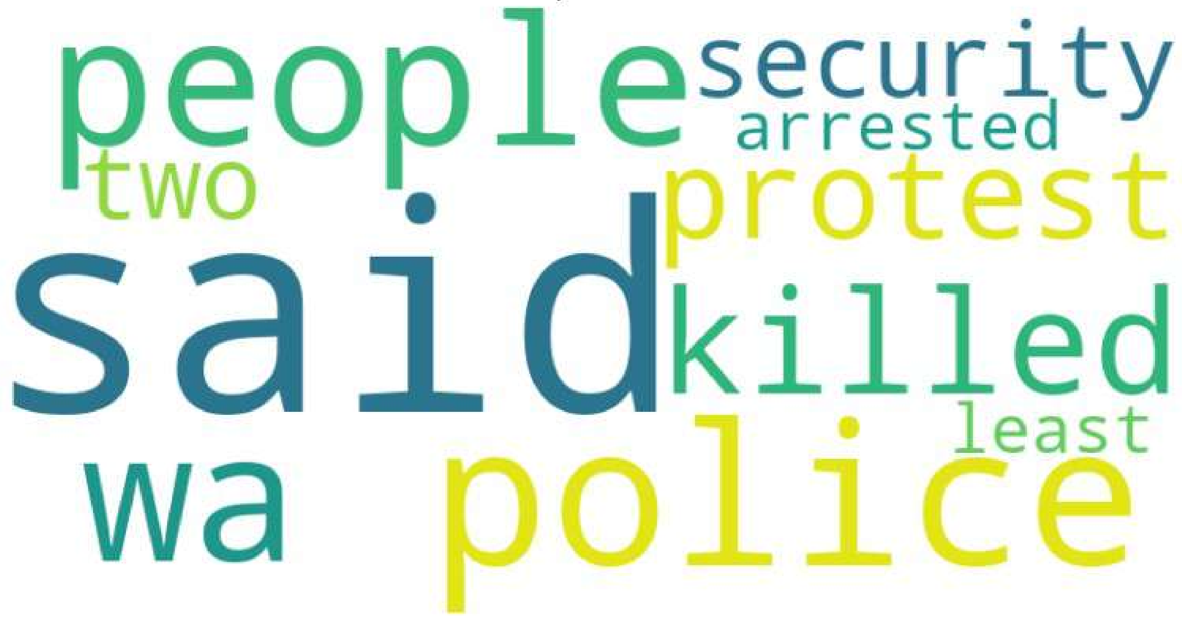
Topic 4



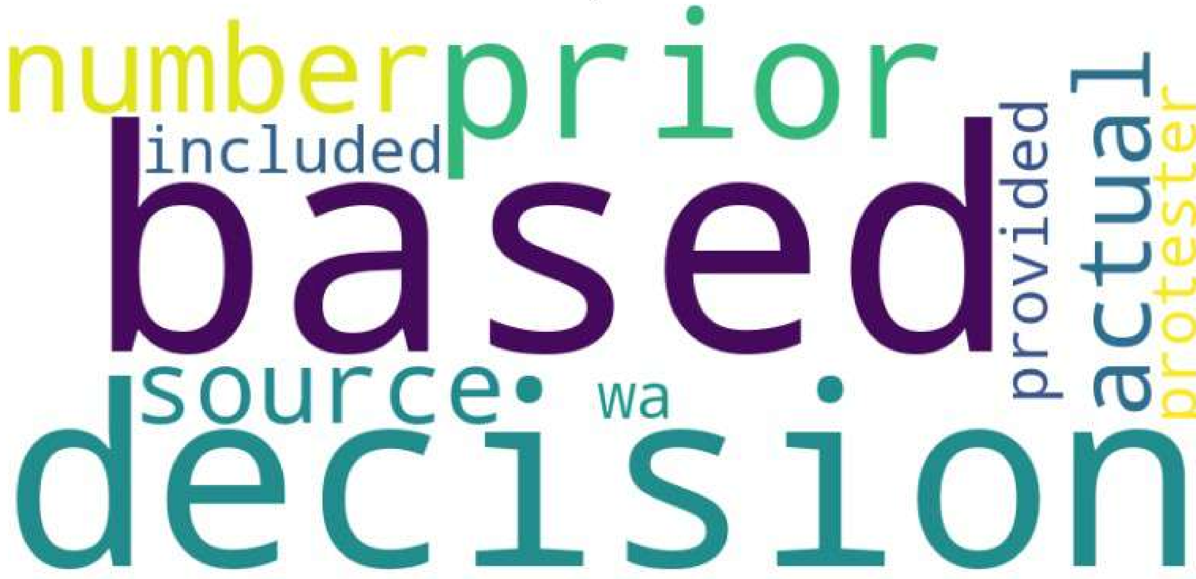
Topic 5



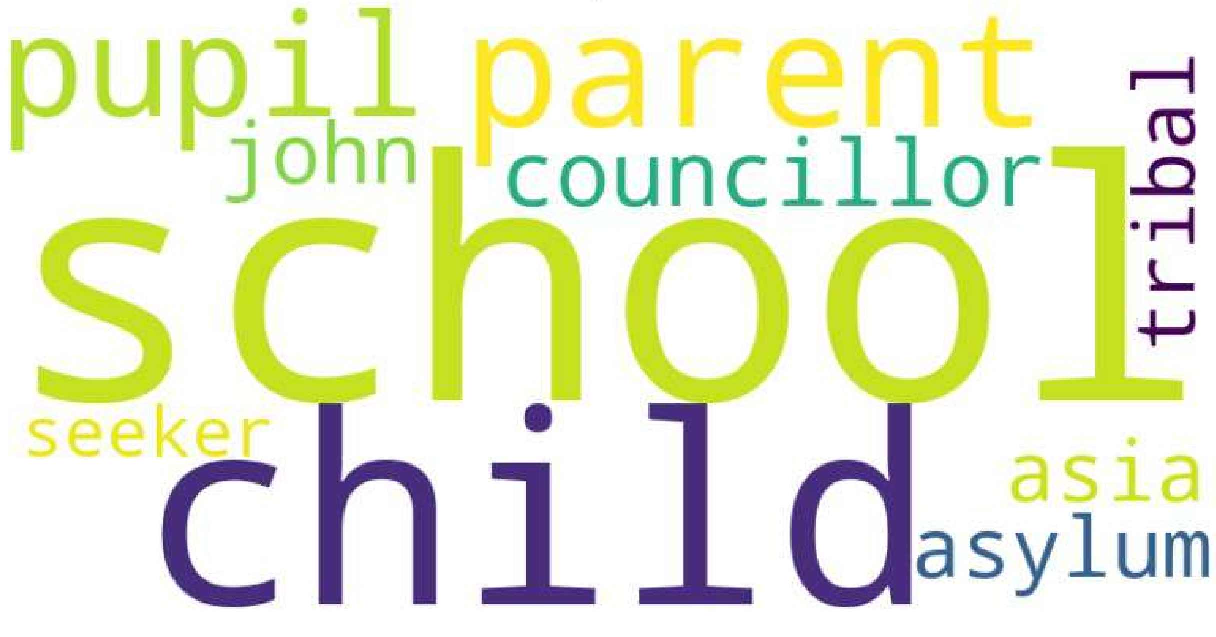
Topic 6



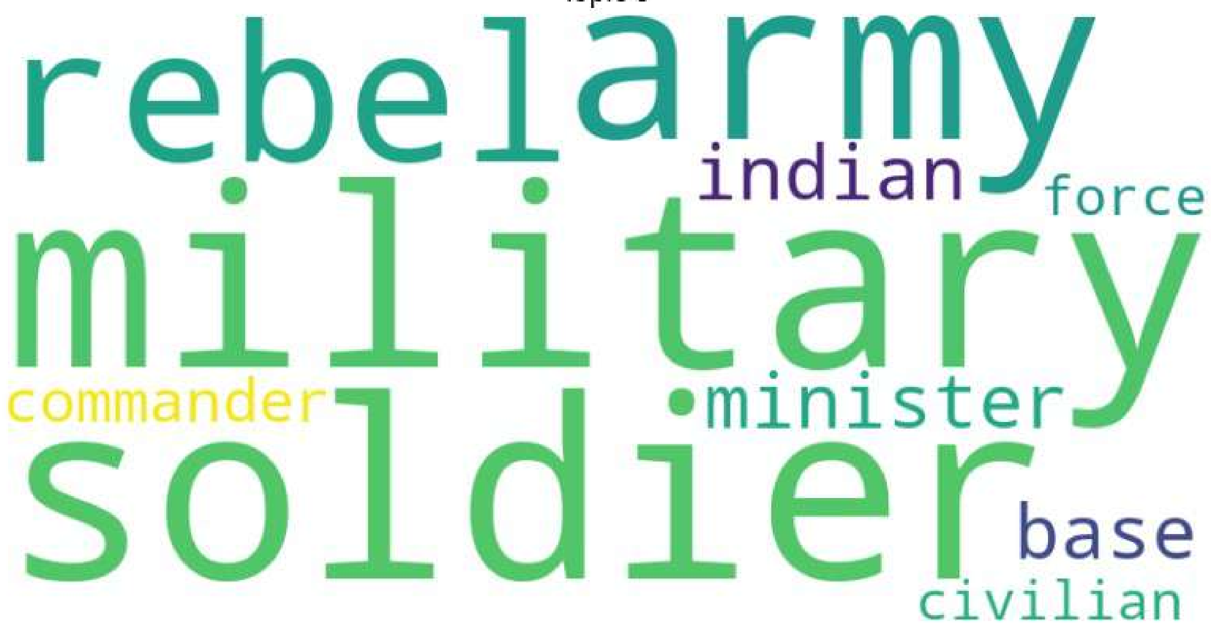
Topic 7



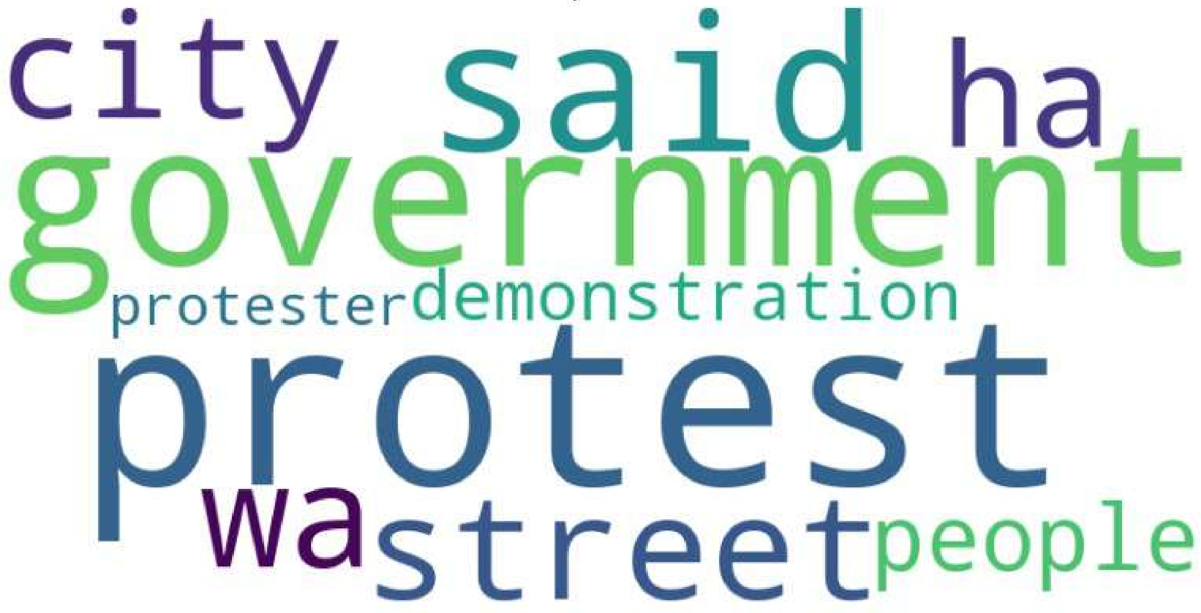
Topic 8



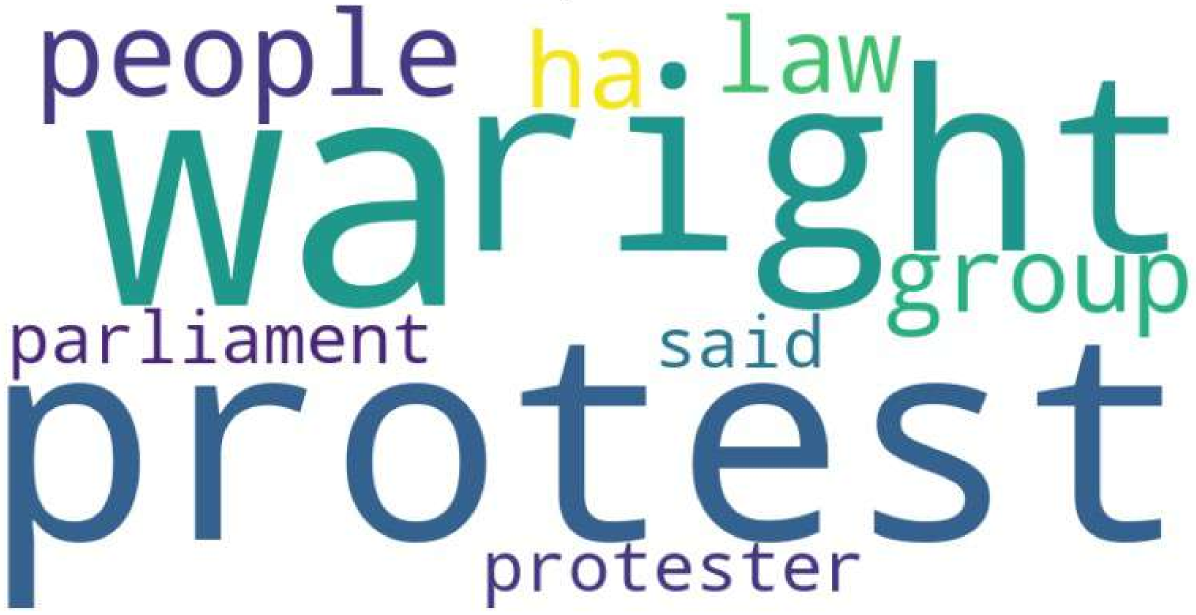
Topic 9



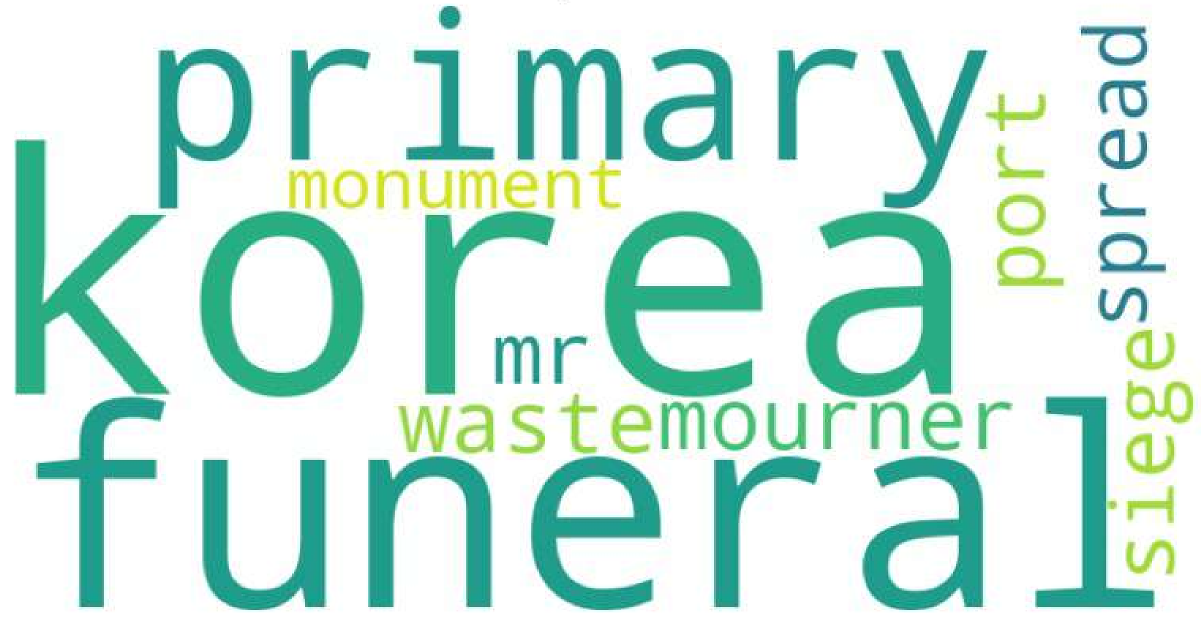
Topic 10



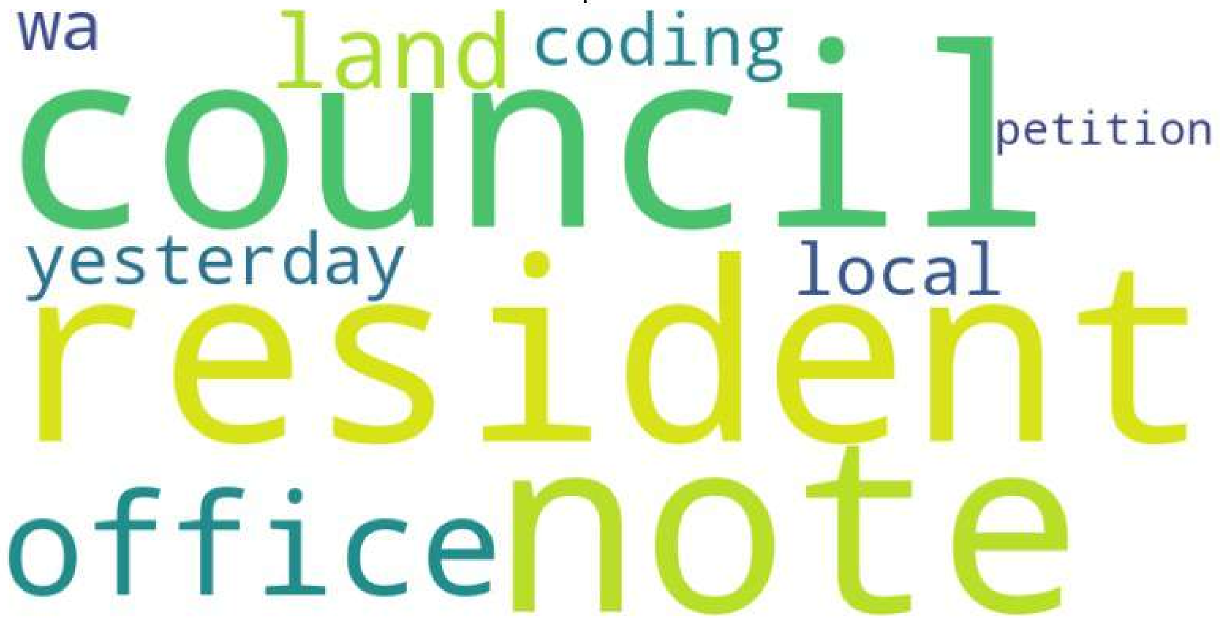
Topic 11



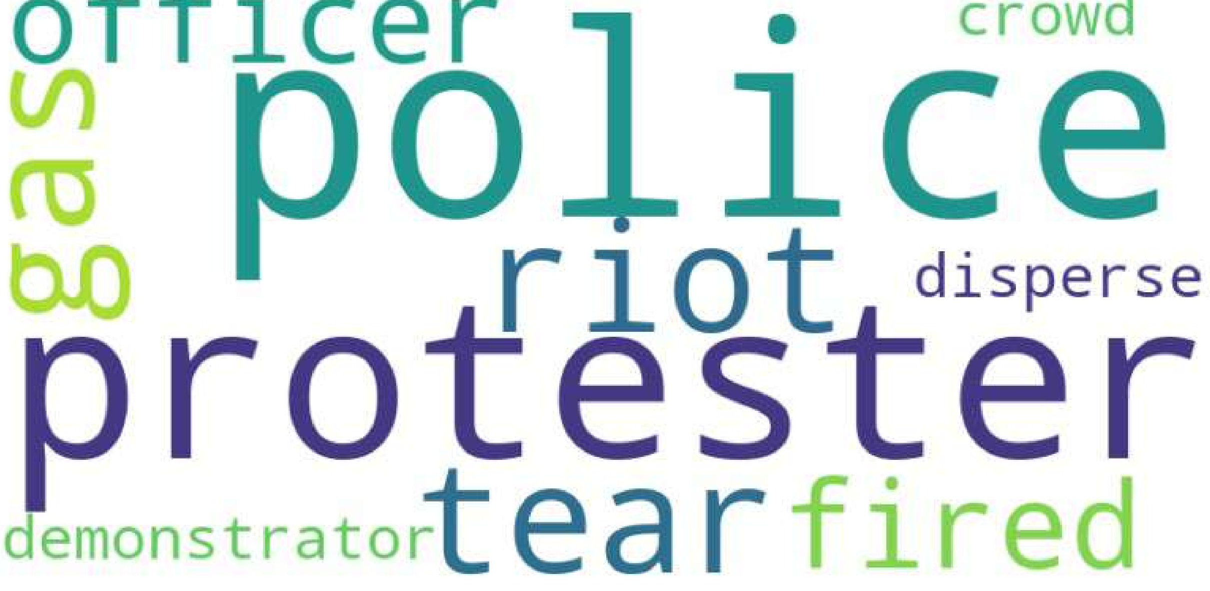
Topic 12



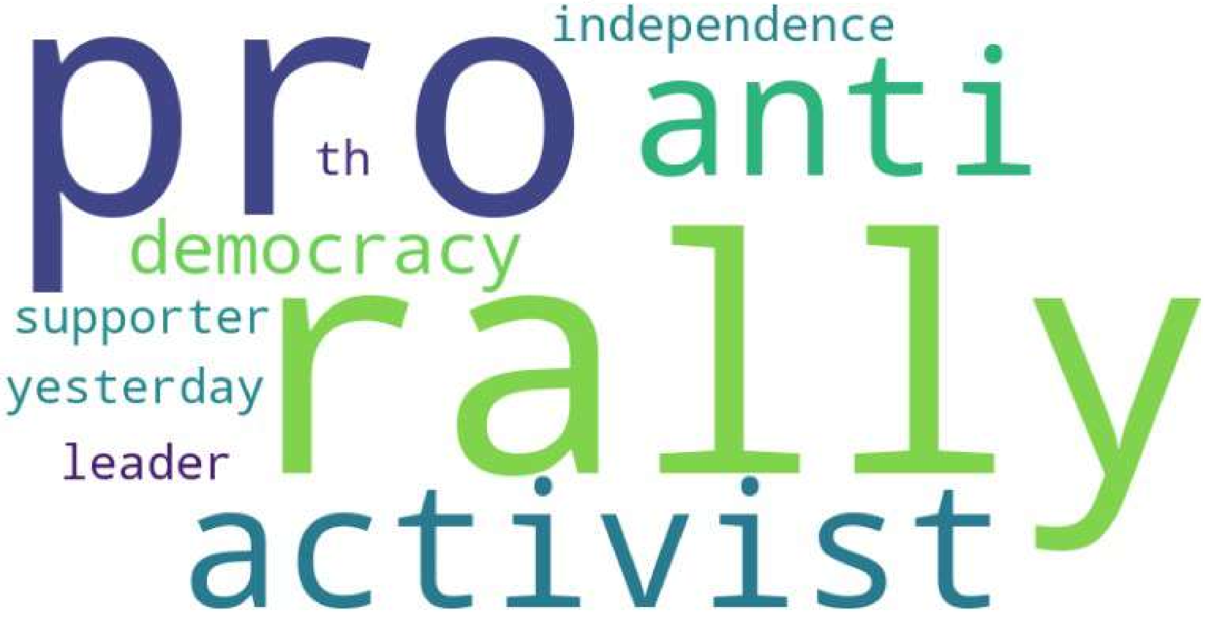
Topic 13



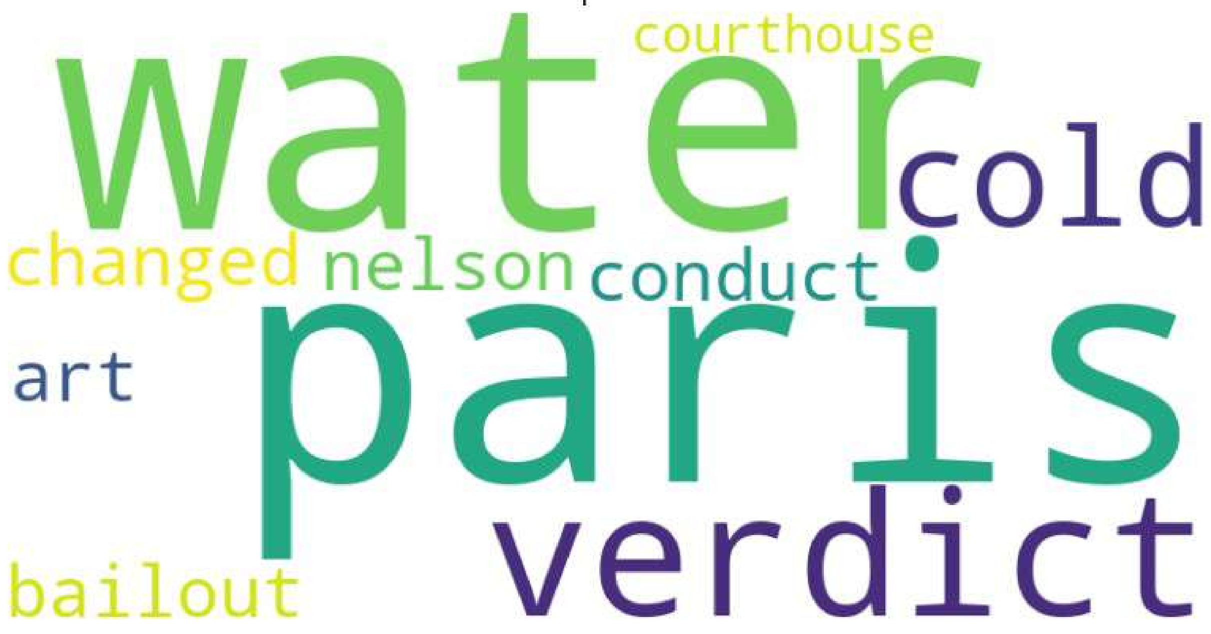
Topic 14



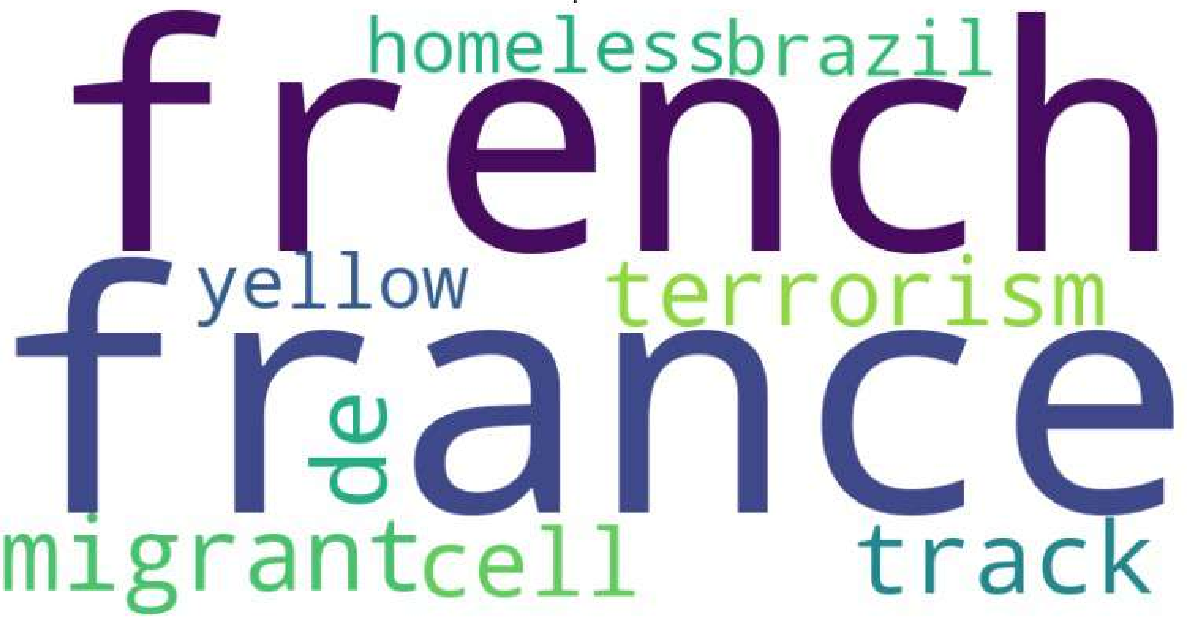
Topic 15



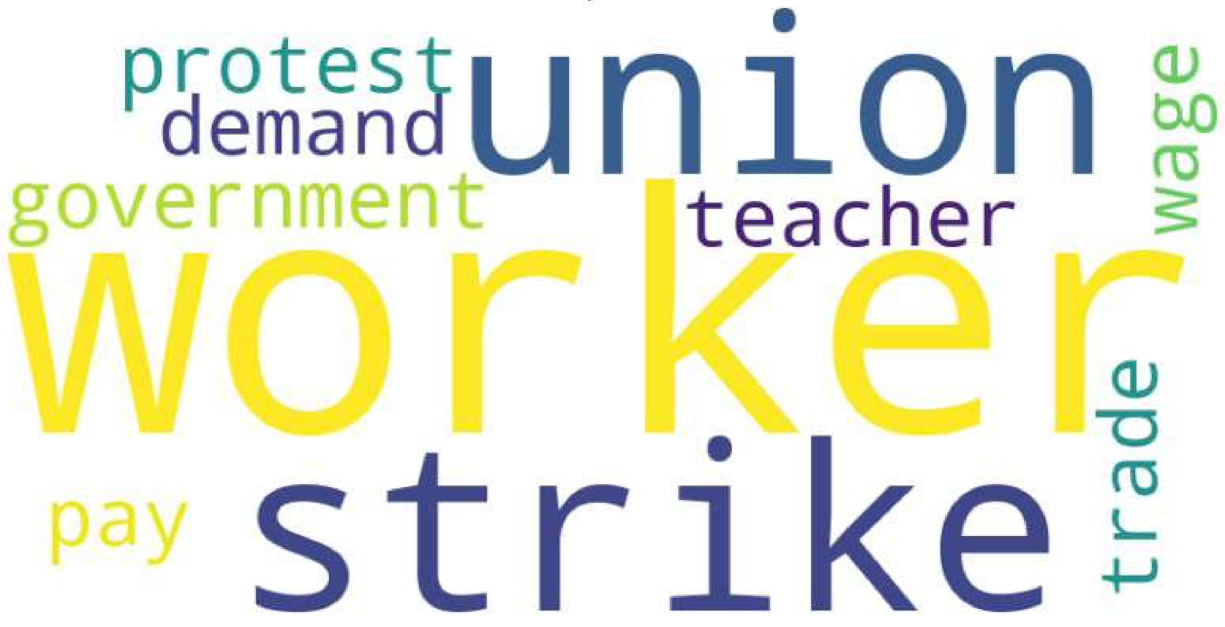
Topic 16



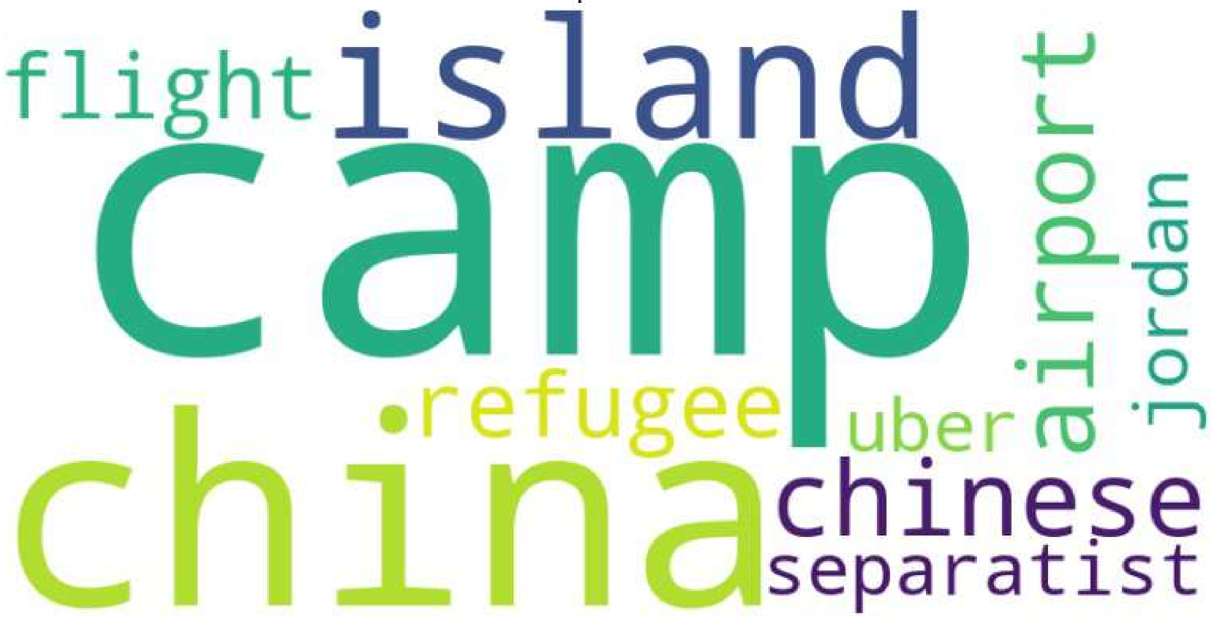
Topic 17



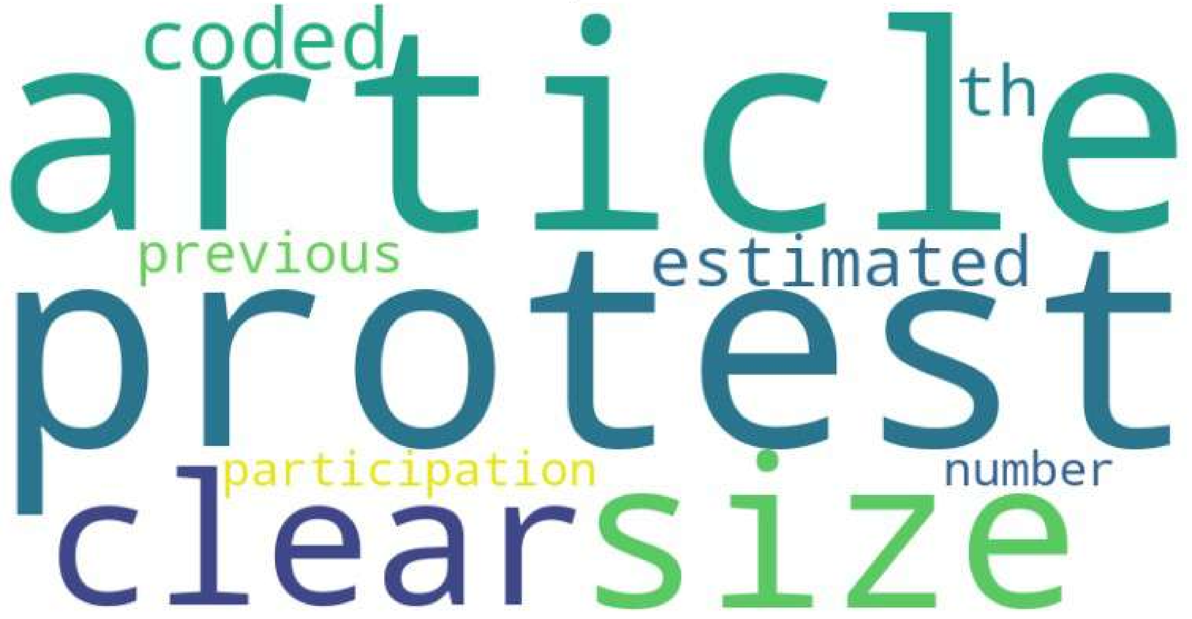
Topic 18



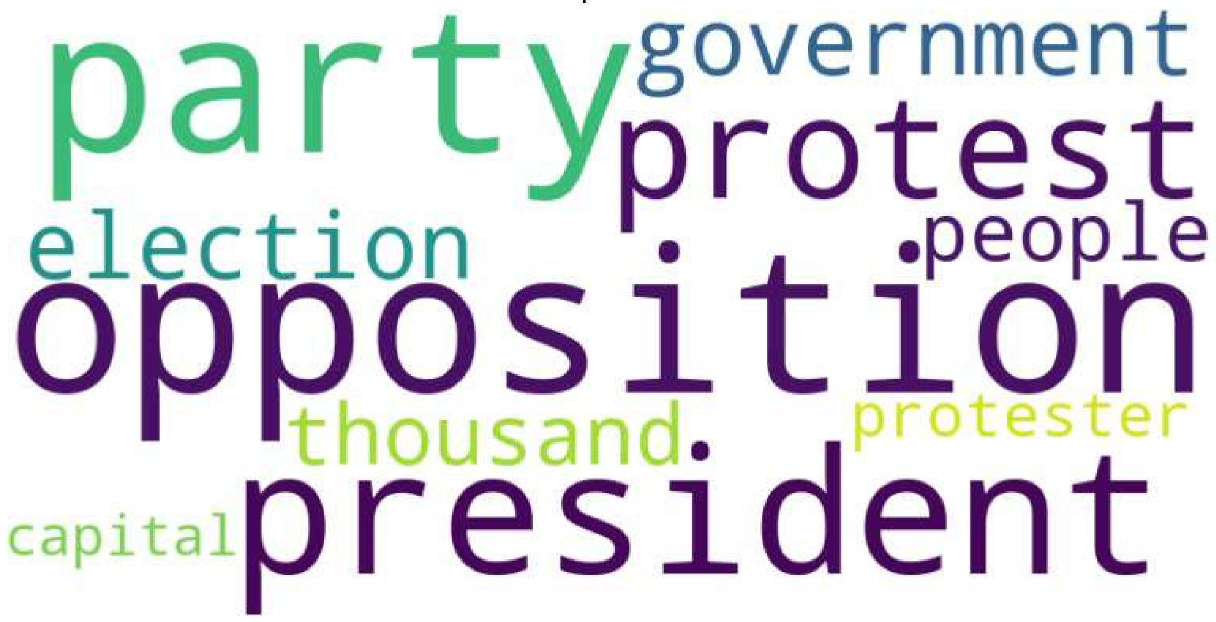
Topic 19



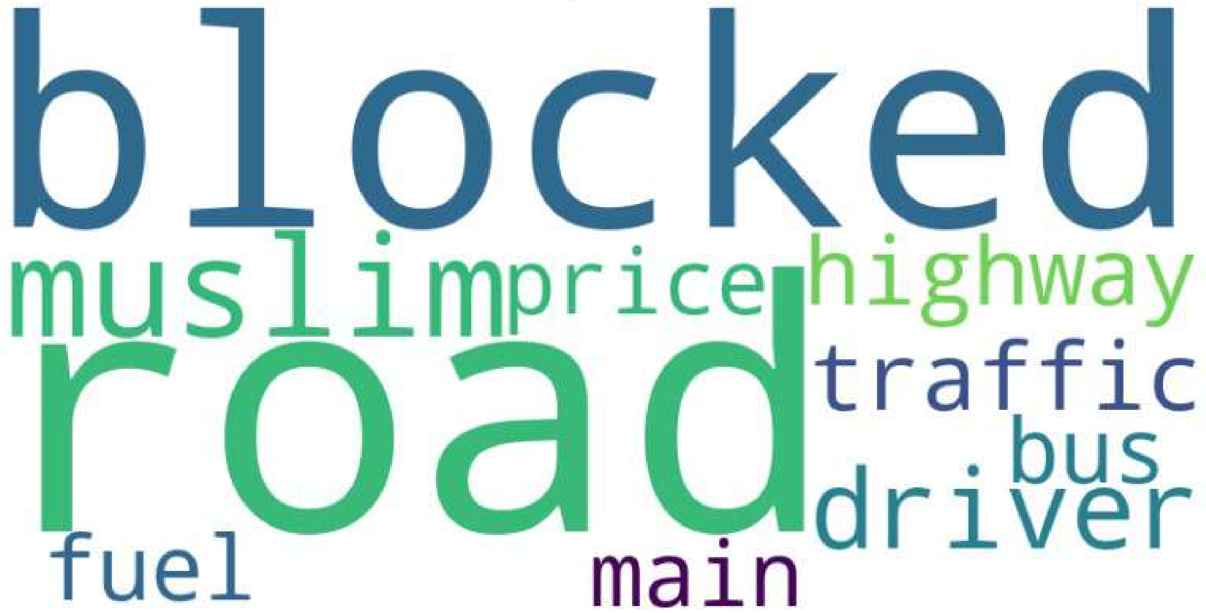
Topic 20



Topic 21



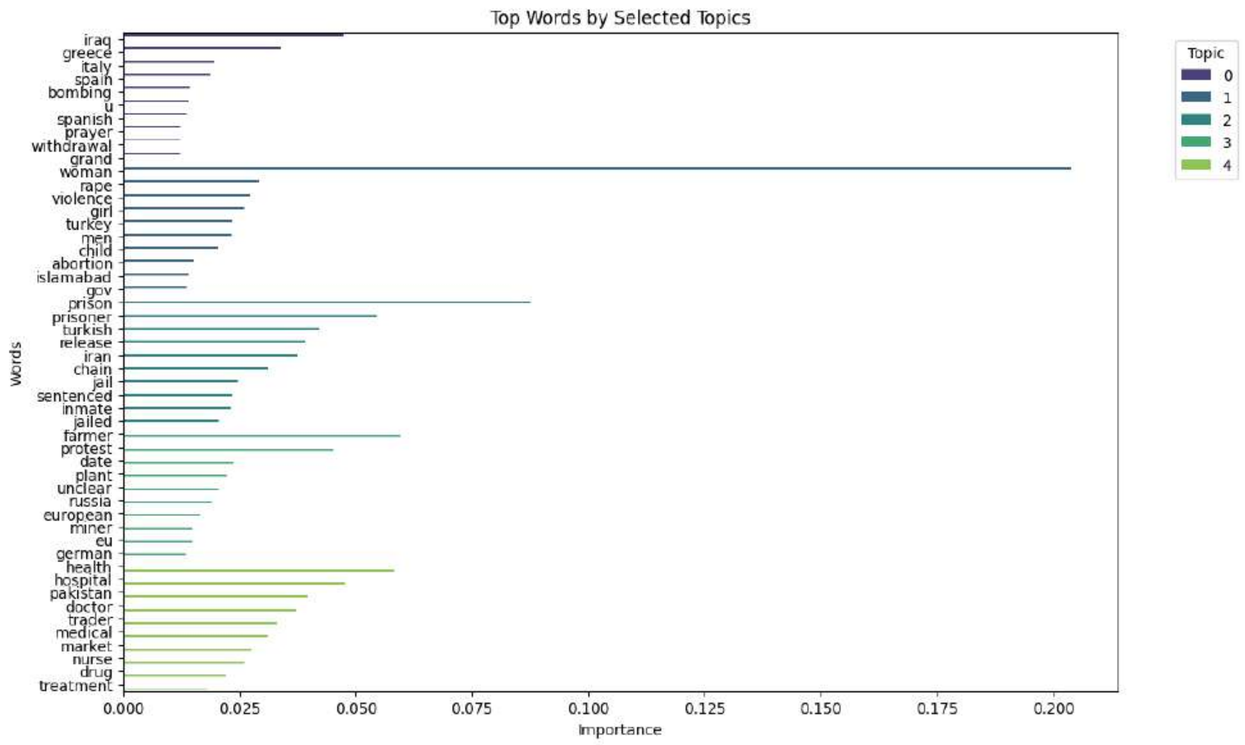
Topic 22



The word clouds show the fifteentopics and the words which are commonly used in each topic

```python
# Prepare data for a subset of topics
def prepare_word_data(topic_id, words):
    word_df = pd.DataFrame(words, columns=['Word', 'Importance'])
    word_df['Topic'] = topic_id
    return word_df

# Select a subset of topics (topics 0 to 4)
```

```
selected_topics = [0, 1, 2, 3, 4]
word_data_subset = pd.concat([prepare_word_data(topic_num,
lda_model.show_topic(topic_num, num_words)) for topic_num in
selected_topics])

# Plot bar chart of most significant words
plt.figure(figsize=(12, 8))
sns.barplot(x='Importance', y='Word', hue='Topic',
data=word_data_subset, palette='viridis')
plt.title('Top Words by Selected Topics')
plt.xlabel('Importance')
plt.ylabel('Words')
plt.legend(title='Topic', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



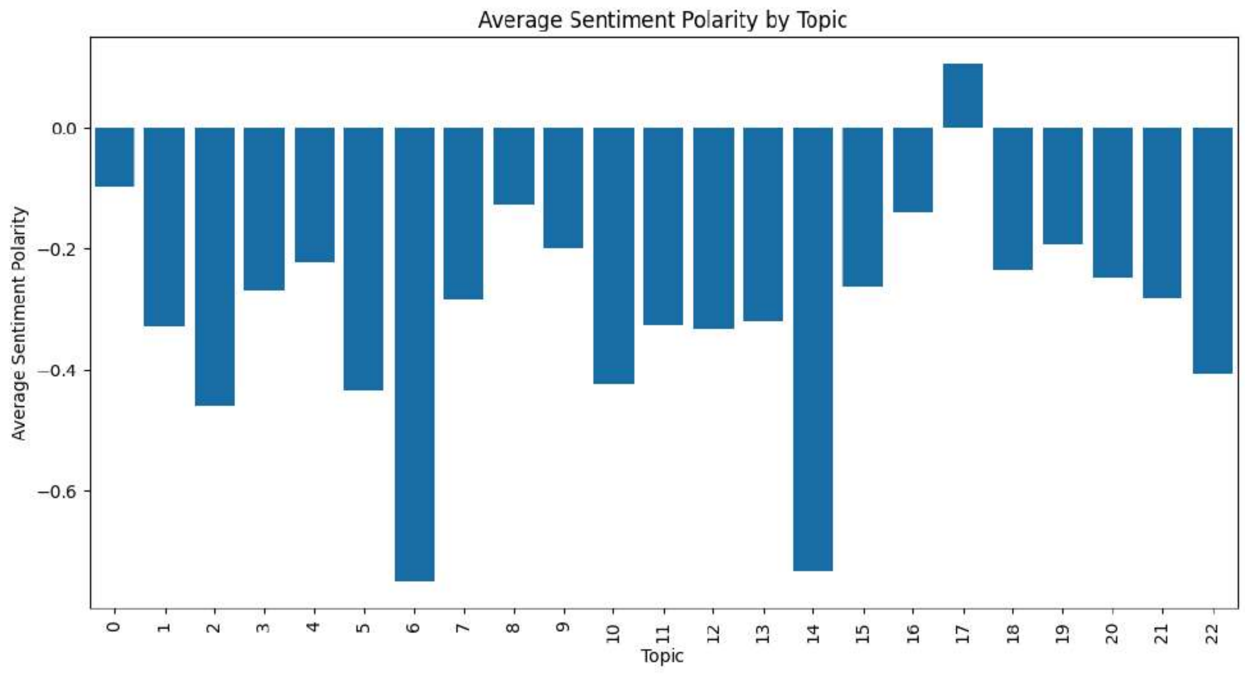
The plot provides insights into the thematic composition of each topic.

Word Importance: Longer bars indicate that a word is more central or representative of a particular topic. For example, "police," "protester," and "protest" are highly important for Topic 0.

Topic Overlap: Some words appear in multiple topics, suggesting potential overlaps or related themes between these topics. For instance, "government" appears in several topics, indicating its relevance across different contexts.

Topic Differentiation: The distinct word distributions for different topics highlight the unique characteristics of each theme. For instance, Topic 1 focuses on health-related terms like "hospital," "doctor," and "nurse," while Topic 2 is more associated with public gatherings and protests, with words like "rally," "thousand," and "marched."

```
# Analyze sentiment by topic
sentiment_by_topic = data_notes.groupby('topic').agg({
    'sentiment_polarity': 'mean',
    'sentiment_neutral': 'mean',
    'sentiment_positive': 'mean',
    'sentiment_negative': 'mean'
}).reset_index()

# Plot sentiment polarity by topic
plt.figure(figsize=(12, 6))
sns.barplot(x='topic', y='sentiment_polarity',
data=sentiment_by_topic, color='#0072bc')
plt.title('Average Sentiment Polarity by Topic')
plt.xlabel('Topic')
plt.ylabel('Average Sentiment Polarity')
plt.xticks(rotation=90)
plt.show()
```



This bar chart shows the Average Sentiment Polarity by Topic. The sentiment polarity likely ranges from -1 (most negative) to +1 (most positive), with 0 being neutral.

Interpretation:

Negative Sentiment (Below 0):

Most Negative Topics: Topic 14 (Police and riots, likely involving harsh responses like tear gas and crowd dispersal) and Topic 6 (Police actions and security during protests where people were killed or arrested) have the most negative sentiment polarity, indicating that the discussions surrounding these topics tend to be particularly negative.

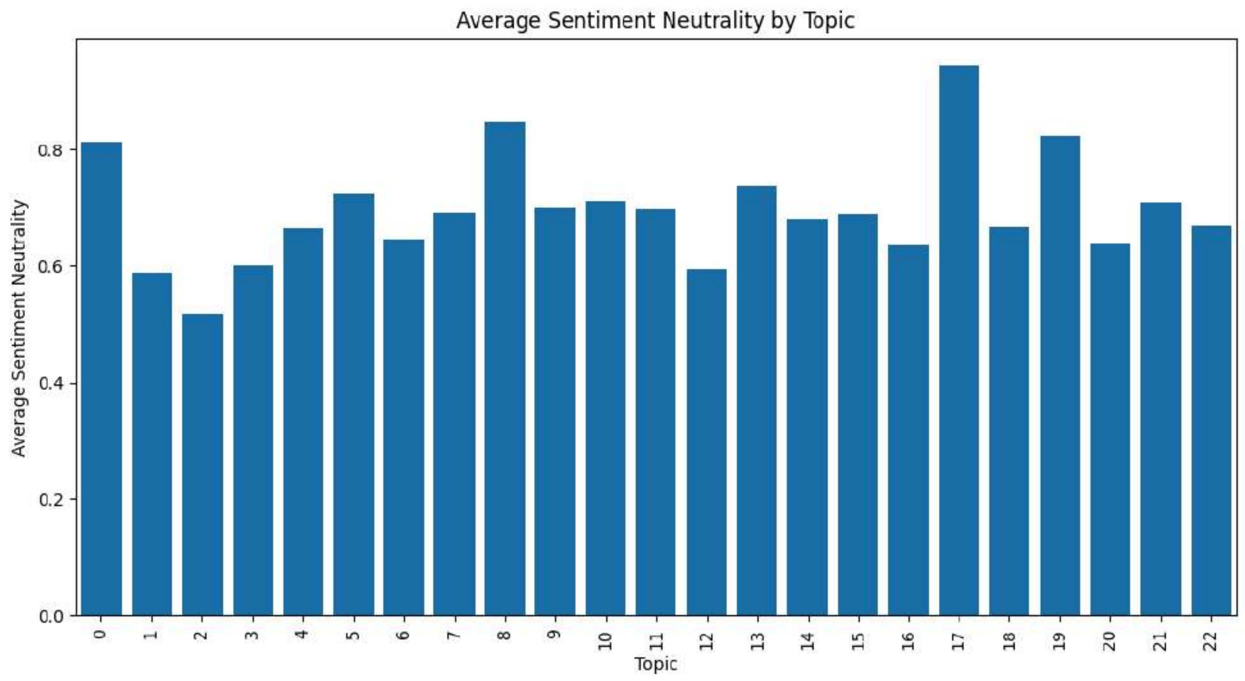
Other Negative Topics: Topics like 2, 3, 4, 5, 7, 8, 9, 10, 11, 12, 18, 20, and 22 also have negative sentiment but are less extreme. These topics likely involve discussions around violence, protests, government actions, and possibly other issues like incarceration, health concerns, and strikes, all of which tend to be associated with more negative sentiment.

Neutral to Slightly Negative Sentiment (Around 0): Topics 0, 1, and 16 have sentiment polarities close to 0, indicating a more neutral or slightly negative sentiment. These topics might involve discussions that are less emotionally charged or more balanced between negative and positive aspects. Positive Sentiment (Above 0):

Topic 17 is the only one with a positive sentiment polarity. This could indicate discussions that are less conflict-driven or focus on more constructive or positive outcomes, possibly related to topics like "French issues" or "social movements" mentioned in your earlier topic descriptions.

Conclusion: The topics with the most negative sentiment likely reflect events or discussions with high emotional impact, such as violence, oppression, or severe social issues. The more neutral topics could involve complex geopolitical or social issues where perspectives might be more balanced. The single positive topic suggests an area where there is less conflict or more positive discourse.

```python
# Plot sentiment neutrality by topic
plt.figure(figsize=(12, 6))
sns.barplot(x='topic', y='sentiment_neutral', data=sentiment_by_topic,
color='#0072bc')
plt.title('Average Sentiment Neutrality by Topic')
plt.xlabel('Topic')
plt.ylabel('Average Sentiment Neutrality')
plt.xticks(rotation=90)
plt.show()
```



The above graph illustrates the average sentiment neutrality for 22 different topics. Sentiment neutrality is a measure of how objective or neutral the text is, with values ranging from 0 to 1. A higher value indicates a more neutral sentiment.

Key Observations

Wide Range of Neutrality: The graph shows significant variation in average sentiment neutrality across the different topics.

Neutral Topics: Topics represented by bars closer to the top of the graph exhibit higher levels of neutrality.

Non-Neutral Topics: Topics with bars closer to the bottom of the graph tend to have more subjective or opinionated content.

The graph provides insights into the overall sentiment tone of the text data associated with these topics.

Diverse Sentiment Landscape: The wide range of neutrality scores suggests that the dataset encompasses topics with varying degrees of objectivity and subjectivity.

Neutral Topics: Topics with high average neutrality might involve factual reporting, news articles, or data-driven content.

Non-Neutral Topics: Topics with low average neutrality could be related to opinion pieces, reviews, or discussions that express personal viewpoints.

This bar chart shows the Average Sentiment Neutrality by Topic. Neutrality scores likely range from 0 to 1, where 0 indicates no neutrality (i.e., very polarized sentiment) and 1 indicates complete neutrality.

Interpretation:

High Neutrality (Above 0.7): Topic 0: Exhibits the highest neutrality score, suggesting that discussions in this topic are generally more neutral, with less emotional or polarized sentiment. Topic 17: Also has a high neutrality score, indicating that the content related to this topic is fairly balanced and neutral in tone. Topic 16: This topic is also quite neutral, which could imply that the discussions are less charged and more factual or balanced.

Moderate Neutrality (Between 0.5 and 0.7): A large number of topics fall within this range, such as Topics 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 18, 19, 20, and 21. These topics show a mix of neutrality and some polarized sentiment, indicating a balance between factual content and emotionally charged discussions.

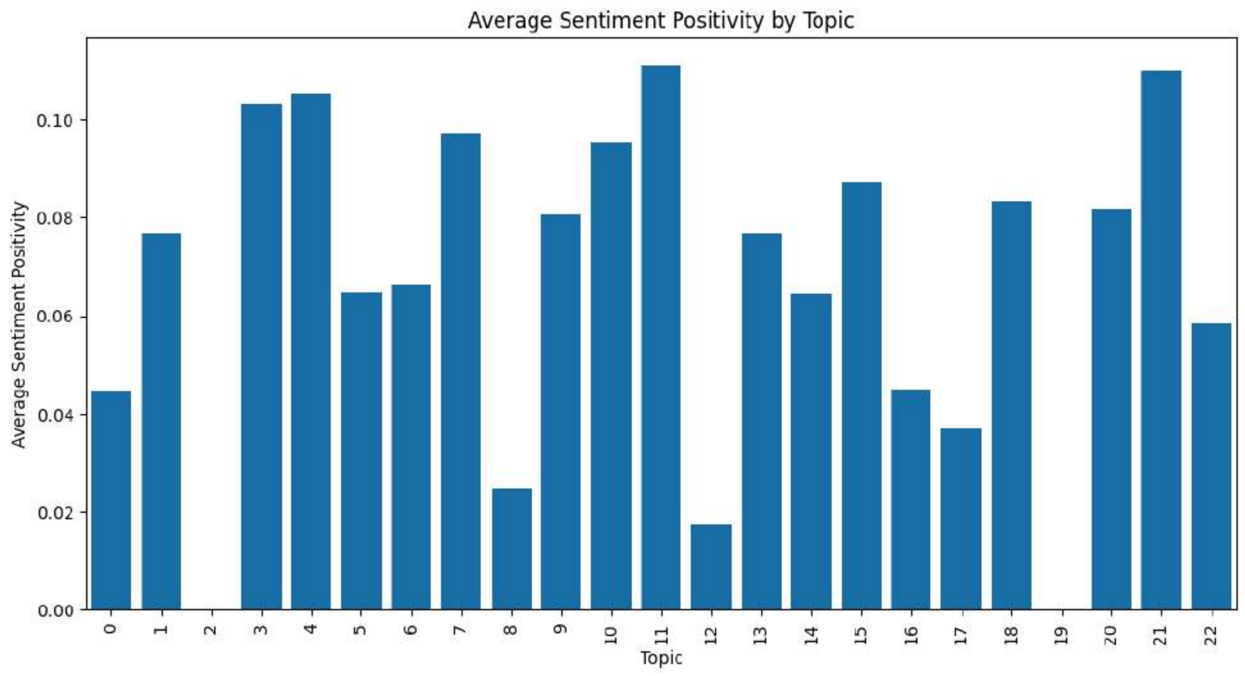
Lower Neutrality (Below 0.5): Topics 1, 2, 3: These topics have lower neutrality scores, meaning that discussions around these topics are more emotionally charged, with less neutral content. Topic 22: Also has a relatively lower neutrality score, suggesting more polarized or emotionally driven discussions.

Correlation with Polarity: The topics with higher neutrality often correlate with lower absolute sentiment polarity values, indicating that these discussions are less likely to be extremely positive or negative. Topics with lower neutrality scores (e.g., Topics 1, 2, 3) tend to correspond with topics that had higher absolute sentiment polarity values, meaning these topics evoke stronger emotions.

Conclusion:

Topics 0 and 17 are generally more neutral, reflecting discussions that are less emotionally charged and more balanced. Topics 1, 2, 3, and 22 are less neutral, indicating that these topics evoke stronger emotional reactions, leading to more polarized sentiment. Understanding neutrality in discussions can help identify which topics are more prone to emotional or biased discourse and which are approached in a more balanced or factual manner. This insight is valuable in analyzing public sentiment and the nature of discussions surrounding various protest-related topics.

```
# Plot sentiment positivity by topic
plt.figure(figsize=(12, 6))
sns.barplot(x='topic', y='sentiment_positive',
data=sentiment_by_topic, color='#0072bc')
plt.title('Average Sentiment Positivity by Topic')
plt.xlabel('Topic')
plt.ylabel('Average Sentiment Positivity')
plt.xticks(rotation=90)
plt.show()
```



This bar chart shows the Average Sentiment Positivity by Topic. The positivity score ranges from 0 to 1, where 0 indicates no positivity and 1 indicates complete positivity in sentiment.

Interpretation:

High Positivity (Above 0.09):

Topics 10, 11, 22, and 3 exhibit the highest positivity scores, indicating that discussions within these topics contain more positive sentiment. Topic 10: Possibly involves some constructive or optimistic discussions related to protests. Topic 11: Could reflect positive or hopeful aspects regarding rights, law, or group efforts. Topic 22: Indicates high positivity, possibly related to discussions about opposition, parties, or elections. Topic 3: Related to areas such as farming, possibly reflecting positive developments or community support.

Moderate Positivity (Between 0.05 and 0.09):

Topics 1, 2, 4, 5, 6, 7, 12, 13, 15, 20, and 21 fall into this range, indicating a moderate level of positive sentiment. Topic 15: Likely includes discussions on democracy and independence, which could evoke a sense of positivity and progress. Topic 7: Could relate to decision-making or evaluations, which may contain some positive outcomes.

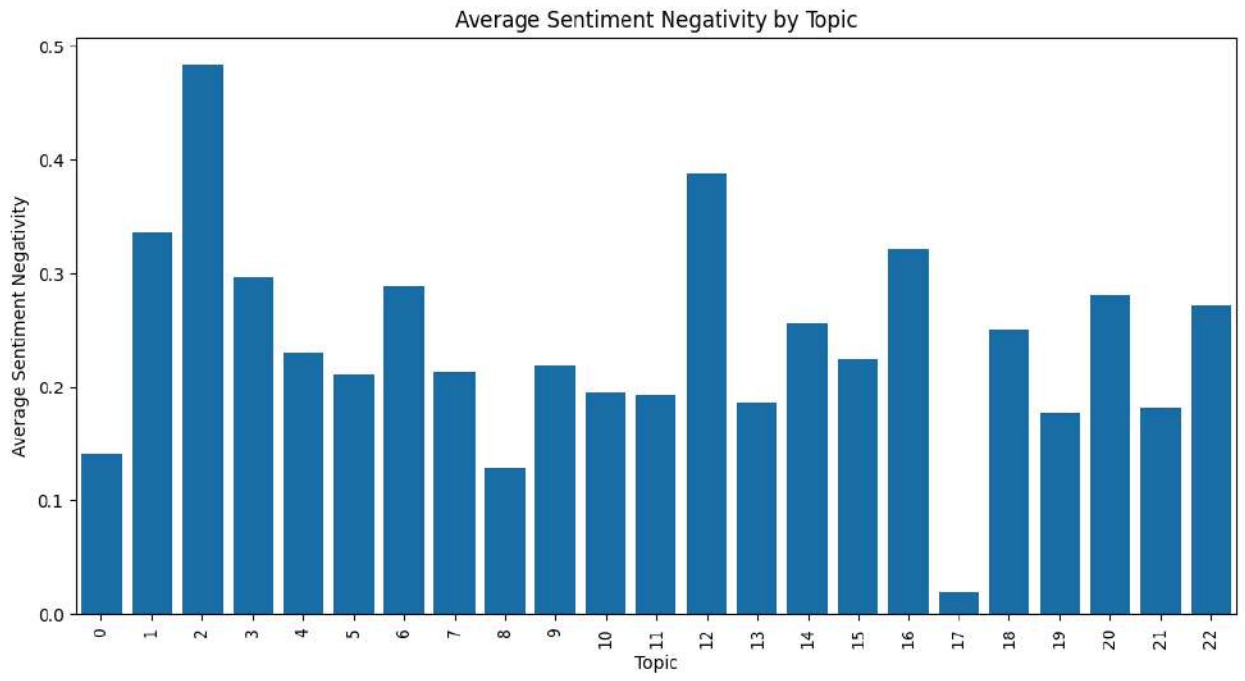
Lower Positivity (Below 0.05):

Topics 0, 8, 14, 16, 17, 18 display lower positivity scores, suggesting that the discussions within these topics are less positive and might lean toward more neutral or negative tones. Topic 14: Involves police and riots, which is consistent with lower positivity due to the negative nature of these discussions. Topic 16 and 17: Could relate to more serious or contentious issues that don't evoke much positivity. Correlation with Polarity and Neutrality: Higher positivity often correlates with lower negative polarity, indicating topics where positive aspects outweigh negative sentiment. Topics with higher positivity are likely to show lower neutrality, as positive sentiment indicates a deviation from neutrality.

Conclusion:

Topics 10, 11, 22, and 3 are characterized by relatively higher positivity, indicating that these topics are generally viewed more favorably or involve more optimistic discussions. Topics 14, 16, and 17 have lower positivity, indicating that discussions are more neutral or negative, likely due to the nature of the topics involved. Understanding the sentiment positivity helps in identifying which protest-related topics are seen in a more positive light and which are more likely to be associated with negative or neutral sentiment.

```python
# Plot sentiment negativity by topic
plt.figure(figsize=(12, 6))
sns.barplot(x='topic', y='sentiment_negative',
data=sentiment_by_topic, color='#0072bc')
plt.title('Average Sentiment Negativity by Topic')
plt.xlabel('Topic')
plt.ylabel('Average Sentiment Negativity')
plt.xticks(rotation=90)
plt.show()
```



###**Average sentiment negativity**

The above graph illustrates the average sentiment negativity for 22 different topics. Sentiment negativity is a measure of how negative the text is, with values ranging from 0 to 0.5. A higher value indicates a more negative sentiment.

Key Observations

Wide Range of Negativity: The graph shows significant variation in average sentiment negativity across the different topics.

Negative Topics: Topics represented by bars closer to the top of the graph exhibit higher levels of negativity.

Neutral/Positive Topics: Topics with bars closer to the bottom of the graph tend to have more neutral or positive content.

The graph provides insights into the overall sentiment tone of the text data associated with these topics.

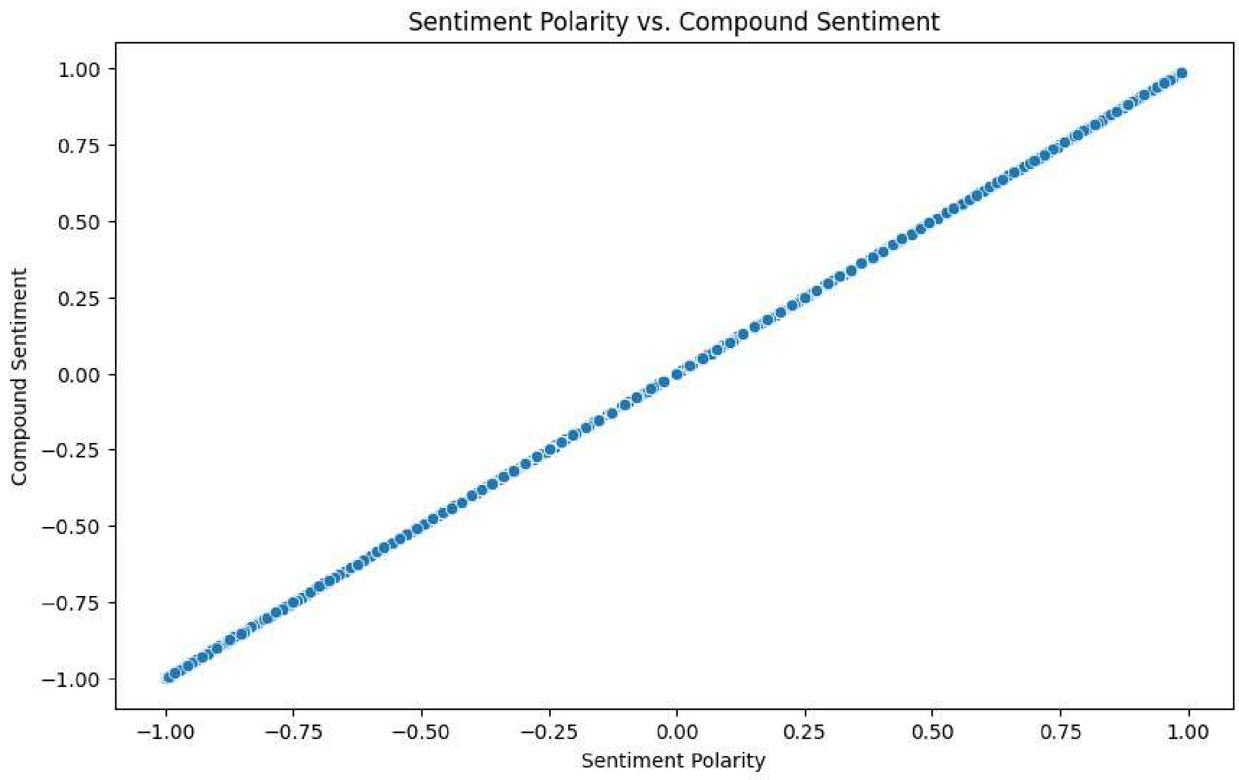
Diverse Sentiment Landscape: The wide range of negativity scores suggests that the dataset encompasses topics with varying degrees of pessimism and optimism.

Negative Topics: Topics with high average negativity might involve negative events, criticisms, or problems.

Neutral/Positive Topics: Topics with low average negativity could be related to positive news, achievements, or neutral discussions.

```
data_notes

                                                   notes  \
0      canada s railway passenger system was finally ...
1      protestors were only identified as young peopl...
2      the queen, after calling on canadians to remai...
3      canada s federal government has agreed to acqu...
4      protests were directed against the state due t...
...                                                  ...
15071  ? a government inquiry will be launched as ser...
15072  police in papua new guinea fired gunshots wedn...
15073  the bougainville government has enacted an ind...
15074  peter o neill has been reappointed as prime mi...
15075  refugees on manus island were braced for poten...

                                           cleaned_notes
compound_sentiment  \
0      canada railway passenger system wa finally cut...              -
0.8316
1      protestors identified young people gathering w...
0.0000
2      queen calling canadian remain united braved pr...
0.8720
3      canada federal government ha agreed acquire tr...              -
0.8481
4      protest directed state due refusal use violenc...              -
0.5423
...                                                  ...
...
15071  government inquiry launched serious question r...              -
0.9920
15072  police papua new guinea fired gunshot wednesda...              -
0.9943
```

```
15073  bougainville government ha enacted indefinite ...          -
0.8519
15074  peter neill ha reappointed prime minister papu...          -
0.9561
15075  refugee manus island braced potential calamity...          -
0.9816

       compound  positive  neutral  negative  sentiment_polarity  \
0       -0.8316     0.000    0.866     0.134             -0.8316
1        0.0000     0.000    1.000     0.000              0.0000
2        0.8720     0.203    0.695     0.103              0.8720
3       -0.8481     0.103    0.736     0.160             -0.8481
4       -0.5423     0.139    0.535     0.326             -0.5423
...         ...       ...      ...       ...                 ...
15071   -0.9920     0.058    0.603     0.339             -0.9920
15072   -0.9943     0.046    0.697     0.257             -0.9943
15073   -0.8519     0.061    0.787     0.152             -0.8519
15074   -0.9561     0.075    0.753     0.173             -0.9561
15075   -0.9816     0.069    0.674     0.258             -0.9816

       sentiment_neutral  sentiment_positive  sentiment_negative
topic
0                  0.866               0.000               0.134
10
1                  1.000               0.000               0.000
6
2                  0.695               0.203               0.103
10
3                  0.736               0.103               0.160
13
4                  0.535               0.139               0.326
22
...                  ...                 ...                 ...    ..
.
15071              0.603               0.058               0.339
6
15072              0.697               0.046               0.257
21
15073              0.787               0.061               0.152
13
15074              0.753               0.075               0.173
21
15075              0.674               0.069               0.258
10

[13816 rows x 12 columns]

# Plot correlation between sentiment polarity and compound sentiment
plt.figure(figsize=(10, 6))
sns.scatterplot(x='sentiment_polarity', y='compound_sentiment',
data=data_notes)
plt.title('Sentiment Polarity vs. Compound Sentiment')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Compound Sentiment')
plt.show()

# Calculate correlation coefficient
correlation = data_notes[['sentiment_polarity',
'compound_sentiment']].corr().iloc[0, 1]
print(f'Correlation between sentiment polarity and compound sentiment:
{correlation}')
```



```
Correlation between sentiment polarity and compound sentiment: 1.0
```

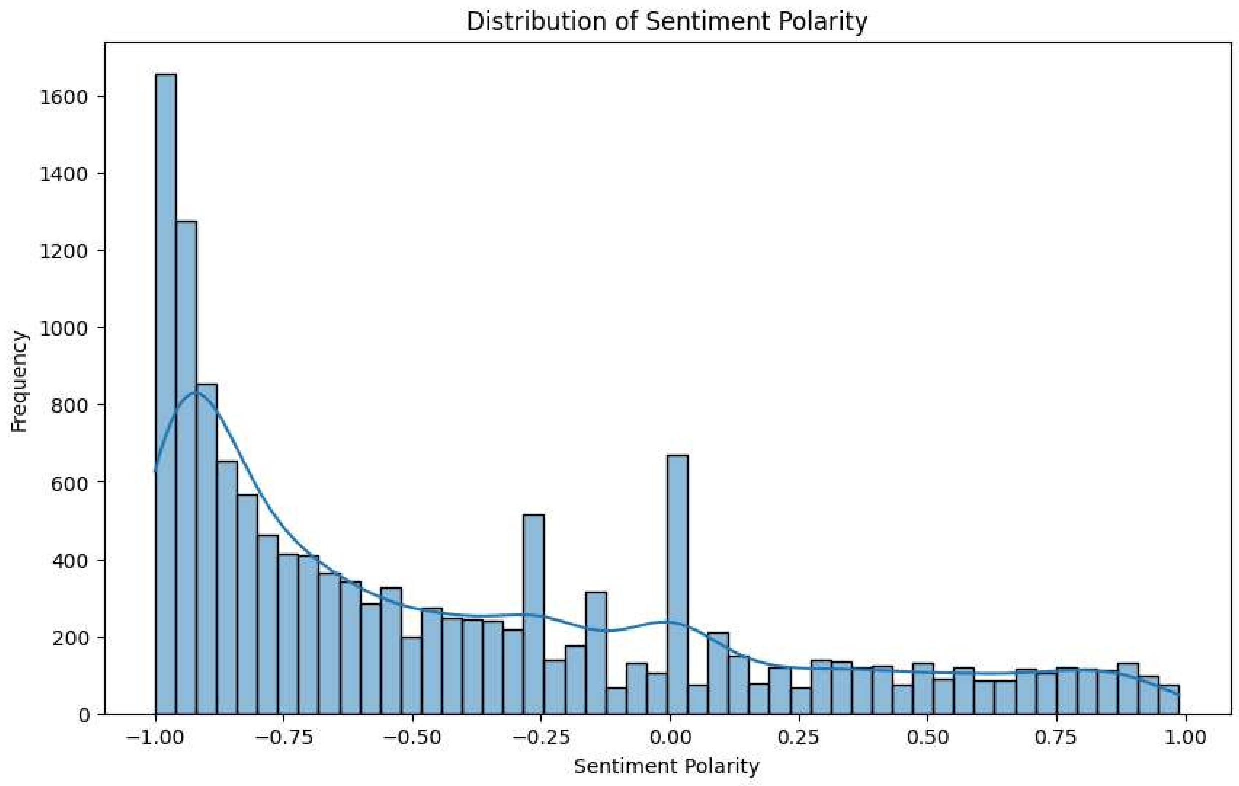
Correlation between Sentiment Polarity and Compound Sentiment

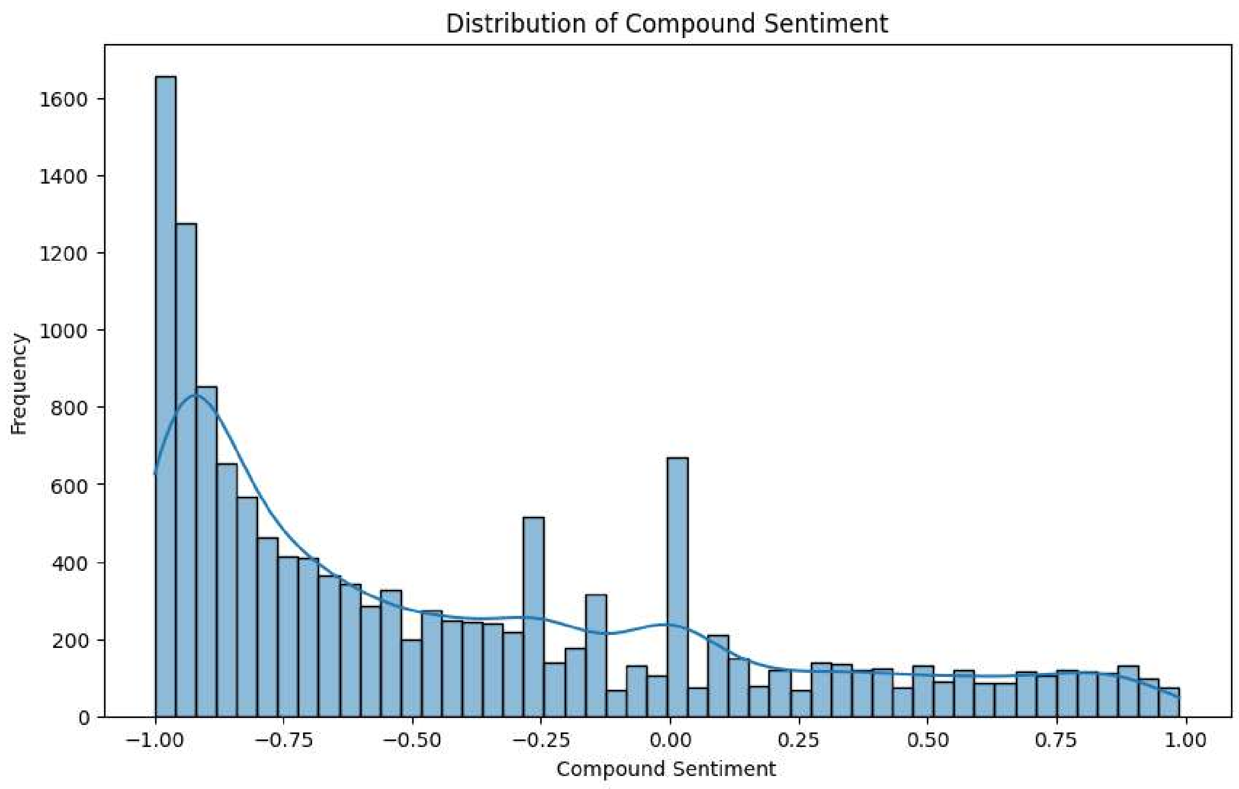
The graph achieved a correlation of 1.0 between sentiment_polarity and compound_sentiment, indicating perfect agreement between the two sentiment scoring methods.

4.2.3 Distribution of Scores

This is to analyze the distribution of sentiment scores to understand the overall sentiment trends in the data.

```python
# Distribution of sentiment polarity
plt.figure(figsize=(10, 6))
sns.histplot(data_notes['sentiment_polarity'], bins=50, kde=True,)
plt.title('Distribution of Sentiment Polarity')
plt.xlabel('Sentiment Polarity')
plt.ylabel('Frequency')
plt.show()

# Distribution of compound sentiment
plt.figure(figsize=(10, 6))
sns.histplot(data_notes['compound_sentiment'], bins=50, kde=True,)
plt.title('Distribution of Compound Sentiment')
plt.xlabel('Compound Sentiment')
plt.ylabel('Frequency')
plt.show()
```





Distribution of Sentiment Polarity:

Sentiment Polarity:

Measures the sentiment expressed in a piece of text, typically on a scale from -1 (very negative) to 1 (very positive). Positive Polarity: Scores greater than 0 indicate positive sentiment, with higher scores indicating stronger positivity.

Distribution of compound sentiments

Right Skew (Positive Skew): In the above right-skewed distribution, the majority of the data points are concentrated on the left side of the distribution (lower values), with a long tail extending to the right (higher values). This means there are fewer data points with high values compared to the lower ones.

4.3 Feature Extraction

Extraction of Bigrams and Trigrams

we will extract bigrams from the text data and analyze their frequency.

```
# Load stop words
stop_words = set(stopwords.words('english'))

# Function to preprocess text and remove stopwords
def preprocess_text(text):
    if isinstance(text, str):  # Check if text is a string
```

```
        tokens = [word for word in text.split() if word.lower() not in
stop_words]
        return ' '.join(tokens)
    else:
        return ''  # Return an empty string if text is not a string

data['cleaned_notes'] = data['notes'].apply(preprocess_text)

from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer

# Extract Bigrams
vectorizer = CountVectorizer(ngram_range=(2, 2))

# Check if 'cleaned_notes' exists, if not, try 'notes' or another
relevant column
if 'cleaned_notes' in data.columns:
    X = vectorizer.fit_transform(data['cleaned_notes'].fillna('')) #
Fill NaN values with empty strings
else:
    # Replace 'notes' with the correct column name if necessary
    print("Warning: 'cleaned_notes' column not found, using 'notes'
instead.")
    X = vectorizer.fit_transform(data['notes'].fillna('')) # Fill NaN
values with empty strings

bigrams = vectorizer.get_feature_names_out()
```

Let's visualize the top 25 most frequent bigrams

```
# Get top 25 most frequent bigrams
bigram_freq = X.sum(axis=0).A1
bigram_freq_df = pd.DataFrame(list(zip(bigrams, bigram_freq)),
columns=['bigram', 'frequency'])
top_25_bigrams = bigram_freq_df.nlargest(25, 'frequency')
```

Extraction of Tigrams

```
# Extract Trigrams
vectorizer = CountVectorizer(ngram_range=(3, 3))


if 'cleaned_notes' in data.columns:
    X = vectorizer.fit_transform(data['cleaned_notes'].fillna('')) #
Fill NaN values with empty strings
else:
    # Replace 'notes' with the correct column name if necessary
    print("Warning: 'cleaned_notes' column not found, using 'notes'
instead.")
    X = vectorizer.fit_transform(data['notes'].fillna('')) # Fill NaN
```

```
values with empty strings

trigrams = vectorizer.get_feature_names_out()

# Get top 10 most frequent trigrams
trigram_freq = X.sum(axis=0).A1
trigram_freq_df = pd.DataFrame(list(zip(trigrams, trigram_freq)),
columns=['trigram', 'frequency'])
top_25_trigrams = trigram_freq_df.nlargest(25, 'frequency')

top_25_bigrams
```

```
                   bigram  frequency
408390           tear gas       1684
349696        riot police       1271
313908     prime minister       1132
604            000 people       1088
304553    police officers        924
419072       took streets        836
365597    security forces        787
219164          last week        643
409615     tens thousands        616
28774     anti government        562
304037       police fired        520
413516   thousands people        503
304865        police said        457
153916         fired tear        422
457300           year old        409
1010               10 000        374
452328      witnesses said        368
13141      across country        357
219101         last night        355
296252     people injured        352
271853  number protesters        345
305229        police used        343
314093       prior coding        342
85817    coding decisions        341
45725         based prior        340
```

This table represents the frequency of specific bigrams (pairs of consecutive words) in a text dataset, likely related to the analysis of protest events or social unrest. The first column seems to be some form of identifier, and the second column lists the bigrams. The third column shows the frequency of each bigram in the dataset. Here’s an interpretation of some of the most frequent bigrams:

"tear gas" (1684 occurrences): This indicates that the phrase "tear gas" is frequently mentioned, suggesting that tear gas is a common tool used in the context described by the text, likely in protests or crowd control situations.

"riot police" (1271 occurrences): The frequent mention of "riot police" reflects the presence or involvement of riot police in many of the described events.

"prime minister" (1132 occurrences): The occurrence of "prime minister" indicates discussions or references to political leaders, possibly in the context of their actions or involvement in the events.

"000 people" (1088 occurrences): This might refer to phrases like "10,000 people," suggesting frequent references to the number of participants in these events.

"police officers" (924 occurrences): The mention of "police officers" highlights their role in these events, perhaps indicating law enforcement's involvement in managing or responding to the situations.

"took streets" (836 occurrences): The phrase "took to the streets" is common in describing protests or mass gatherings, indicating significant public demonstrations.

"security forces" (787 occurrences): This term is likely used to describe the involvement of various government security personnel in the events.

"last week" (643 occurrences): The phrase "last week" suggests frequent references to recent events, indicating a focus on current or ongoing protests.

"tens thousands" (616 occurrences): This likely refers to "tens of thousands," indicating large crowds or participants in these events.

"anti government" (562 occurrences): The presence of "anti-government" suggests that many protests or events have an anti-government stance.

These bigrams provide insight into common themes or elements present in the dataset, indicating frequent references to protests, government responses, large gatherings, and law enforcement actions.

```
top_25_trigrams
```

```
                           trigram  frequency
217031              fired tear gas        422
456769      prior coding decisions        341
65813           based prior coding        338
642592               used tear gas        310
438441           police fired tear        261
385781          note actual number        249
23488     actual number protesters        241
387355  number protesters included        224
561640          source based prior        214
599989           tear gas disperse        187
443086            police used tear        179
279654       included source based        165
471663  protesters included source        162
511875         riot police officers        150
602082       tens thousands people        139
427402          people took streets        128
2792                 10 000 people        109
427398            people took part        109
1454             000 people marched        107
```

```
600290              tear gas water        104
1559               000 people took        100
217409             firing tear gas        100
235143          gas rubber bullets         96
600187            tear gas rubber          96
42698   anti government protesters         94
```

This table represents the frequency of specific trigrams (three consecutive words) in a text dataset, likely focused on protest events or social unrest. Here's an interpretation of some of the most frequent trigrams:

"fired tear gas" (422 occurrences): This trigram indicates that "fired tear gas" is a common phrase, suggesting frequent reporting of tear gas being used by authorities to disperse crowds.

"prior coding decisions" (341 occurrences) & "based prior coding" (338 occurrences): These trigrams seem to relate to the methodology or decisions made during the coding or categorization process of the data, possibly indicating frequent references to how data was handled or analyzed.

"used tear gas" (310 occurrences): The frequent use of this phrase highlights that tear gas is a commonly mentioned method employed by authorities in the context described.

"police fired tear" (261 occurrences): This trigram likely refers to "police fired tear gas," reinforcing the frequent mention of law enforcement using tear gas.

"note actual number" (249 occurrences), "actual number protesters" (241 occurrences), "number protesters included" (224 occurrences): These trigrams suggest that there is a focus on accurately reporting or analyzing the number of protesters, indicating attention to the scale of the events.

"source based prior" (214 occurrences), "included source based" (165 occurrences), "protesters included source" (162 occurrences): These trigrams appear to relate to how information was sourced or included in the analysis, potentially highlighting the methodology or data sources used in the research.

"tear gas disperse" (187 occurrences): This phrase indicates that tear gas was frequently used to disperse crowds, which aligns with the context of protest events.

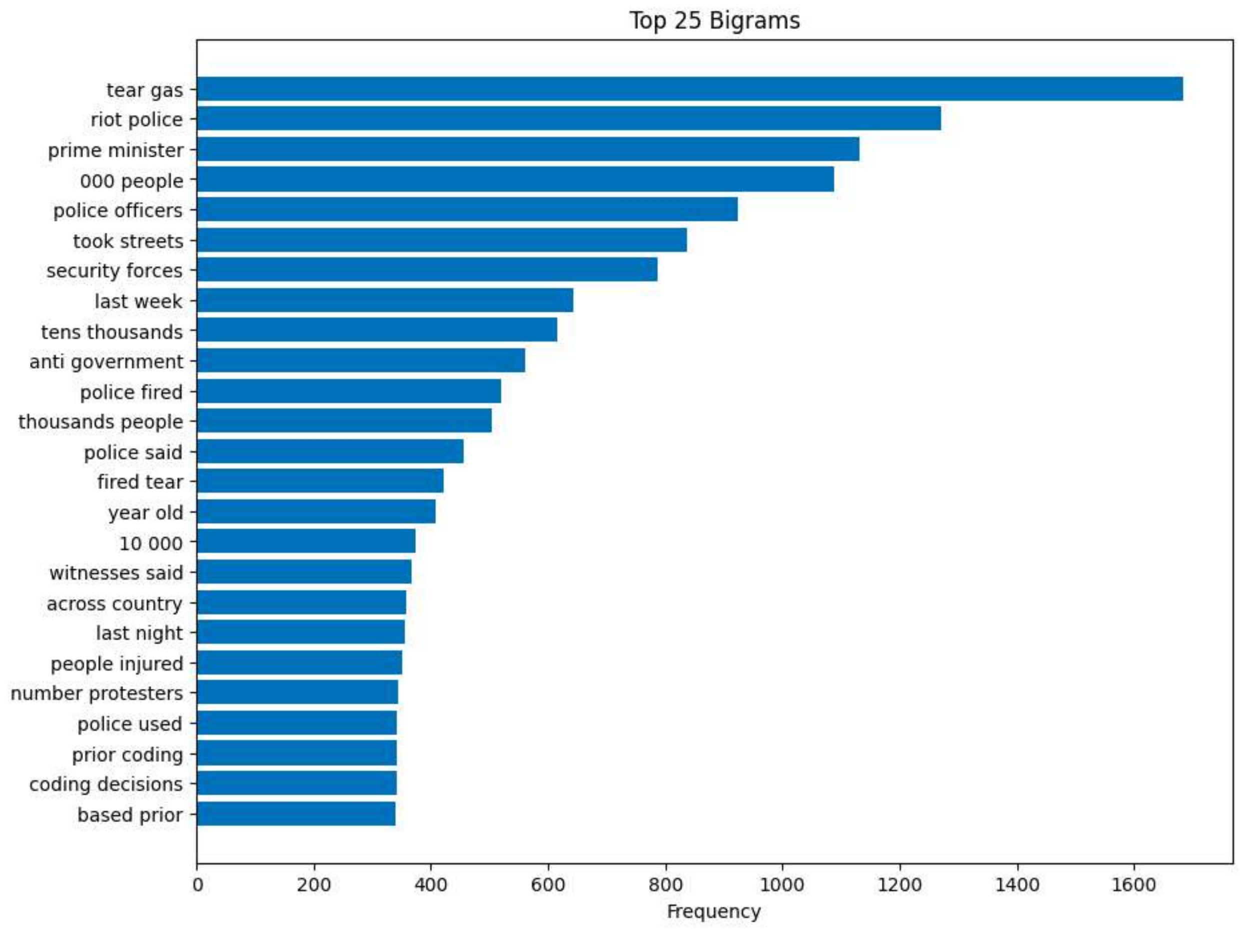
"riot police officers" (150 occurrences): The mention of "riot police officers" indicates the involvement of specialized law enforcement units in managing the events.

"tens thousands people" (139 occurrences), "people took streets" (128 occurrences), "people took part" (109 occurrences): These trigrams emphasize the large number of participants and the widespread nature of the protests, with people actively taking to the streets.

"tear gas rubber bullets" (96 occurrences): This phrase suggests that tear gas was often used in conjunction with rubber bullets, highlighting the use of multiple crowd control methods.

These trigrams offer more detailed insights into the common themes and actions present in the dataset, especially focusing on law enforcement tactics like the use of tear gas, the scale of protests, and the methodology used in analyzing or reporting the events.

```
import matplotlib.pyplot as plt

# Plotting the top 25 bigrams
plt.figure(figsize=(10, 8))
plt.barh(top_25_bigrams['bigram'], top_25_bigrams['frequency'],
color='#0072bc')
plt.xlabel('Frequency')
plt.title('Top 25 Bigrams')
plt.gca().invert_yaxis()
plt.show()
```



The chart shows the most frequent bigrams (two-word combinations) in the dataset, along with their frequencies. These bigrams provide insight into common themes and topics present in the text data. Here's a detailed interpretation of the results:

"tear gas": This is the most frequent bigram, indicating a significant focus on events involving the use of tear gas. It suggests that discussions about tear gas are prevalent in the dataset.

"000 people": This bigram is likely part of phrases like "thousands of people" or "hundreds of people," pointing to the large-scale involvement of people in the events being discussed.

"riot police": The presence of riot police is a common topic, highlighting discussions about law enforcement's role during the events.

"prime minister": This indicates that the Prime Minister is frequently mentioned, suggesting discussions about political leadership or actions taken by the Prime Minister.

"police officers": Similar to "riot police," this bigram emphasizes the involvement of police officers in the events.

"took streets": This suggests that people taking to the streets, a common form of protest or demonstration, is a frequent subject.

"tens thousands": This bigram likely extends to "tens of thousands," emphasizing the large number of people involved in the events.

"last week": This indicates that many notes refer to events that happened in the recent past, providing a temporal context.

"last night": Similar to "last week," this bigram shows a focus on recent events, specifically those that occurred the previous night.

"thousands people": Again, emphasizing the large-scale involvement of people.

"police said": This suggests that statements or reports from the police are frequently mentioned, indicating a focus on official responses or accounts.

"general strike": This bigram points to discussions about widespread labor strikes, which are significant forms of protest.

"10 000": Likely part of phrases indicating the number of people involved in the events.

"anti government": This indicates that many notes discuss anti-government sentiments or actions.

"year old": This could refer to the ages of individuals mentioned in the notes, possibly highlighting the demographics involved.

"took part": Suggests discussions about participation in events or activities.

"people arrested": Indicates that the arrest of individuals is a frequent topic.

"european union": Points to discussions involving the European Union, suggesting a broader geopolitical context.

"across country": Indicates that events are happening nationwide, highlighting the widespread nature of the events.

"protest government": Suggests that many notes discuss protests against the government.

"people marched": Highlights that marching is a common form of protest mentioned in the notes.

"police used": Likely extends to "police used tear gas" or similar phrases, emphasizing police actions.

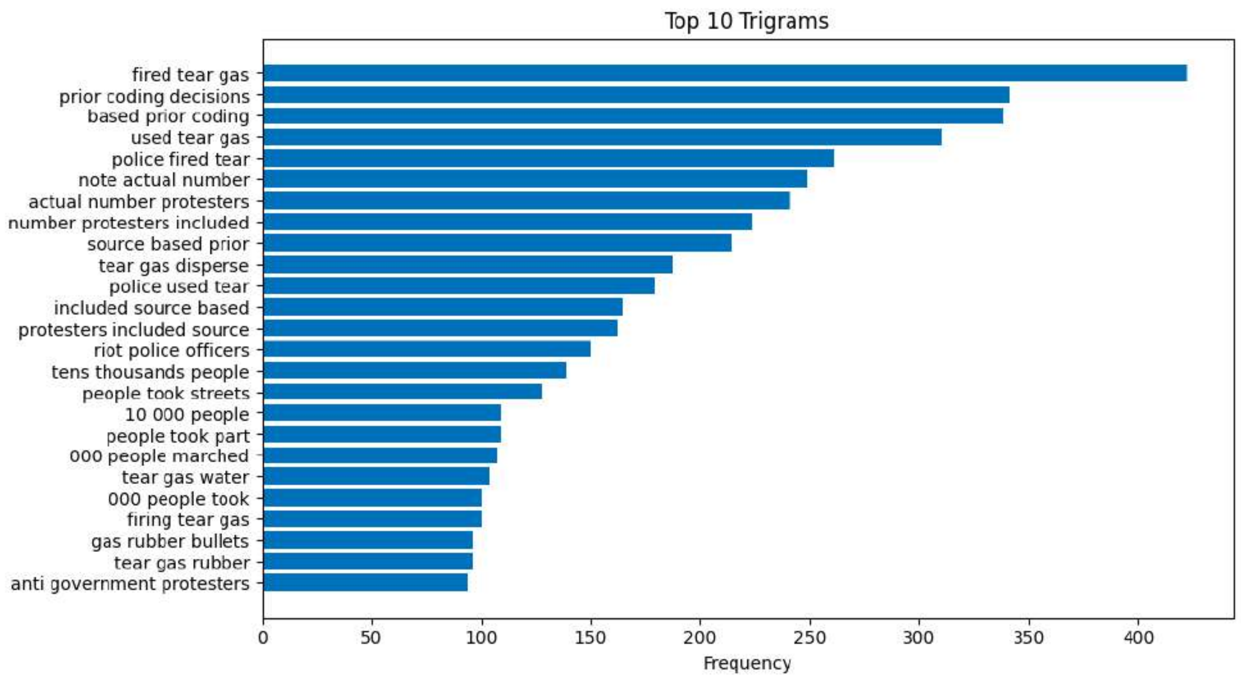
"yesterday protest": Indicates that protests occurring recently (yesterday) are frequently mentioned.

"people injured": Highlights that injuries sustained by people during events are a significant topic.

"per cent": Likely part of statistical discussions, possibly about participation rates, public opinion, or other relevant metrics.

Overall, the bigrams highlight themes of protest, police actions, large-scale public involvement, political leadership, and recent events. These frequent phrases help identify the primary subjects and concerns within the dataset

```
# Plotting the top 10 trigrams
plt.figure(figsize=(10, 6))
plt.barh(top_10_trigrams['trigram'], top_10_trigrams['frequency'],
color='#0072bc')
plt.xlabel('Frequency')
plt.title('Top 10 Trigrams')
plt.gca().invert_yaxis()
plt.show()
```



The chart shows the most frequent trigrams (three-word combinations) in the dataset, along with their frequencies. The trigrams provide insight into common phrases and themes present in the text data. Here's a detailed interpretation of the results:

"fired tear gas": This trigram is the most frequent, indicating that many notes discuss incidents where tear gas was fired. The high frequency suggests that this is a significant topic in the dataset.

"used tear gas": Similar to the previous trigram, this one also points to discussions about the use of tear gas, showing that it’s a prevalent subject.

"police used tear": This phrase likely extends to "police used tear gas," emphasizing the involvement of police in using tear gas during events.

"police fired tear": This likely extends to "police fired tear gas," reinforcing the theme of police actions involving tear gas.

"port au prince": This trigram indicates that Port-au-Prince, the capital of Haiti, is frequently mentioned, suggesting that many notes might be discussing events occurring there.

"president hugo chavez": Mentions of President Hugo Chavez suggest discussions related to his actions or influence, possibly in a historical or political context.

"000 people marched": This is likely part of phrases like "thousands of people marched," indicating significant protest events involving large crowds.

"10 000 people": Similar to the previous point, this suggests discussions about large groups of people, possibly in the context of protests or demonstrations.

"people took streets": This trigram indicates that people taking to the streets, a common form of protest, is a frequent topic.

"tens thousands people": This also points to discussions involving large groups of people, emphasizing the scale of the events being described.

Overall, the trigrams highlight themes of police action, the use of tear gas, large-scale protests, and specific geographical and political contexts (such as Port-au-Prince and Hugo Chavez). These frequent phrases help identify the primary subjects and concerns within the dataset.

```python
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from collections import defaultdict
from nltk import ngrams
from sklearn.decomposition import LatentDirichletAllocation
from sklearn.feature_extraction.text import CountVectorizer

# Download necessary NLTK data
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\Magda\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

```
True
```

```python
# Load the tweets file
tweets_df = pd.read_csv('tweets.csv')

# Load the CSV files for Swahili stopwords and slang
swahili_stopwords_df = pd.read_csv('Common Swahili Stop-words.csv')
swahili_slang_df = pd.read_csv('Common Swahili Slangs.csv')

# Preview the first 3 columns of the tweets file.
tweets_df.head(3)
```

```
                                                link  \
0  https://twitter.com/KennedyMuling94/status/181...
1  https://twitter.com/KennedyMuling94/status/181...
```

```
2  https://twitter.com/KennedyMuling94/status/181...

                                                text  \
0  #OccupyParliament meant the power is back to t...
1  #OccupyParliament meant the power is back to t...
2  #OccupyParliament meant the power is back to t...

                        date  no_of_likes  no_of_comments
0  Jul 21, 2024 · 5:02 PM UTC            0               0
1  Jul 21, 2024 · 4:54 PM UTC            0               0
2  Jul 21, 2024 · 4:53 PM UTC            0               0
```

The dataset consists of tweets, with each row containing information about an individual tweet.

The columns include:

link: The URL to the specific tweet.

text: The content of the tweet.

date: The timestamp of when the tweet was posted, including the date and time.

no_of_likes: The number of likes the tweet received.

no_of_comments: The number of comments the tweet received.

```
# View the columns
tweets_df.columns

Index(['link', 'text', 'date', 'no_of_likes', 'no_of_comments'],
dtype='object')

# Copy the data frane
original_copy = tweets_df.copy()

# Drop the columns that will not be needed for this analysis
tweets_df.drop(columns=['link', 'no_of_likes', 'no_of_comments'],
inplace=True)

# Check if the columns have been dropped
tweets_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15265 entries, 0 to 15264
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    15255 non-null  object
 1   date    15265 non-null  object
dtypes: object(2)
memory usage: 238.6+ KB
```

```python
# Check for duplicates and drop them.
tweets_df.drop_duplicates(inplace=True)

# Check if the duplicates have been dropped
tweets_df
```

```
                                                    text  \
0      #OccupyParliament meant the power is back to t...
1      #OccupyParliament meant the power is back to t...
2      #OccupyParliament meant the power is back to t...
3      #OccupyParliament meant the power is back to t...
4      #OccupyParliament meant the power is back to t...
...                                                  ...
15253  Get eid of the new funding model completely if...
15254  When people think they don't have a say in gov...
15255  US Secret Service Director resigns within 24hr...
15256  The goal of The Butcher was to just win 2022 e...
15257                                        #RUTOMUSTGO

                             date
0      Jul 21, 2024 · 5:02 PM UTC
1      Jul 21, 2024 · 4:54 PM UTC
2      Jul 21, 2024 · 4:53 PM UTC
3      Jul 21, 2024 · 4:52 PM UTC
4      Jul 21, 2024 · 4:50 PM UTC
...                           ...
15253  Jul 24, 2024 · 5:40 AM UTC
15254  Jul 24, 2024 · 5:39 AM UTC
15255  Jul 24, 2024 · 5:39 AM UTC
15256  Jul 24, 2024 · 5:39 AM UTC
15257  Jul 24, 2024 · 5:39 AM UTC

[13802 rows x 2 columns]
```

This dataset contains a series of tweets with two columns:

text: The content of each tweet.

date: The timestamp of when each tweet was posted.

Key observations:

The dataset includes 13,802 rows, indicating a large collection of tweets.

The tweets span multiple days, starting from July 21, 2024, and continuing at least until July 24, 2024.

The content of the tweets varies, with some including hashtags like #OccupyParliament and #RUTOMUSTGO. These suggest that the tweets may be related to social or political movements, possibly in response to specific events or decisions.

Some tweets appear to be part of a continuous stream with minimal time differences between them, such as those between 4:50 PM and 5:02 PM on July 21, 2024.

```
# View the swahili stopwords that will be used for the analysis
swahili_stopwords_df

    StopWords
0          na
1      lakini
2      ingawa
3    ingawaje
4         kwa
..        ...
250      nini
251      hasa
252       huu
253      zako
254      mimi

[255 rows x 1 columns]

# View the swahili_slang words
swahili_slang_df

          Slang                           Meaning Meaning1 Meaning2
\
0         manzi                          msichana      NaN      NaN

1          slay                          msichana      NaN      NaN

2         queen                          msichana      NaN      NaN

3          mshi                          msichana      NaN      NaN

4         chick                          msichana      NaN      NaN

..          ...                               ...      ...      ...

182         arv  dawa za kufubaza virusi vya ukimwi      NaN      NaN

183        arvs  dawa za kufubaza virusi vya ukimwi      NaN      NaN

184  tunacorona                       tuna corona      NaN      NaN

185        lais                             raisi      NaN      NaN

186     nyumban                          nyumbani      NaN      NaN


    Meaning3
0        NaN
```

```
1          NaN
2          NaN
3          NaN
4          NaN
..         ...
182        NaN
183        NaN
184        NaN
185        NaN
186        NaN

[187 rows x 5 columns]

# Convert the stopwords and slang words to sets
swahili_stopwords = set(swahili_stopwords_df['StopWords'].tolist())
swahili_slang = set(swahili_slang_df['Slang'].tolist())

# Define stopwords including English, Swahili, and Swahili slang
stop_words =
set(stopwords.words('english')).union(swahili_stopwords).union(swahili
_slang)

# Function to clean the text
def clean_text(text):
    text = re.sub(r"http\S+|www\S+|https\S+", '', text,
flags=re.MULTILINE)  # Remove URLs
    text = re.sub(r'\@\w+|\#','', text)  # Remove @ and # characters
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
    return text

# Function to preprocess the text
def preprocess_text(text):
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    swahili_stop_words = set (swahili_stopwords.words('swahili'))
    swahili_slang = set (swahili_slang.words('slang'))
    tokens = [word for word in tokens if word not in stop_words]
    stop_words =
set(stopwords.words('english')).union(swahili_stop_words,
swahili_slang)
    tokens = [word for word in tokens if word not in stop_words]
    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return ' '.join(tokens)
```

```
# Function to clean the text
def clean_text(text):
    if isinstance(text, str):  # Check if text is a string
        text = re.sub(r"http\S+|www\S+|https\S+", '', text,
flags=re.MULTILINE)  # Remove URLs
        text = re.sub(r'\@\w+|\#','', text)  # Remove @ and #
characters
        text = re.sub(r'\d+', '', text)  # Remove numbers
        text = text.lower()  # Convert to lowercase
        text = re.sub(r'[^\w\s]', '', text)  # Remove punctuation
        return text
    else:
        return ''  # Return an empty string if text is not a string

# Clean the text column
tweets_df['cleaned_text'] = tweets_df['text'].apply(clean_text)

tweets_df
```

```
                                                    text  \
0      #OccupyParliament meant the power is back to t...
1      #OccupyParliament meant the power is back to t...
2      #OccupyParliament meant the power is back to t...
3      #OccupyParliament meant the power is back to t...
4      #OccupyParliament meant the power is back to t...
...                                                  ...
15253  Get eid of the new funding model completely if...
15254  When people think they don't have a say in gov...
15255  US Secret Service Director resigns within 24hr...
15256  The goal of The Butcher was to just win 2022 e...
15257                                        #RUTOMUSTGO

                              date  \
0      Jul 21, 2024 · 5:02 PM UTC
1      Jul 21, 2024 · 4:54 PM UTC
2      Jul 21, 2024 · 4:53 PM UTC
3      Jul 21, 2024 · 4:52 PM UTC
4      Jul 21, 2024 · 4:50 PM UTC
...                            ...
15253  Jul 24, 2024 · 5:40 AM UTC
15254  Jul 24, 2024 · 5:39 AM UTC
15255  Jul 24, 2024 · 5:39 AM UTC
15256  Jul 24, 2024 · 5:39 AM UTC
15257  Jul 24, 2024 · 5:39 AM UTC

                                            cleaned_text
0      occupyparliament meant the power is back to th...
1      occupyparliament meant the power is back to th...
2      occupyparliament meant the power is back to th...
3      occupyparliament meant the power is back to th...
```

```
4      occupyparliament meant the power is back to th...
...                                                 ...
15253  get eid of the new funding model completely if...
15254  when people think they dont have a say in gove...
15255  us secret service director resigns within hrs ...
15256  the goal of the butcher was to just win  elect...
15257                                        rutomustgo

[13802 rows x 3 columns]
```

```
tweets_df.drop(columns=['text'], inplace=True)

tweets_df
```

```
                             date  \
0      Jul 21, 2024 · 5:02 PM UTC
1      Jul 21, 2024 · 4:54 PM UTC
2      Jul 21, 2024 · 4:53 PM UTC
3      Jul 21, 2024 · 4:52 PM UTC
4      Jul 21, 2024 · 4:50 PM UTC
...                           ...
15253  Jul 24, 2024 · 5:40 AM UTC
15254  Jul 24, 2024 · 5:39 AM UTC
15255  Jul 24, 2024 · 5:39 AM UTC
15256  Jul 24, 2024 · 5:39 AM UTC
15257  Jul 24, 2024 · 5:39 AM UTC

                                           cleaned_text
0      occupyparliament meant the power is back to th...
1      occupyparliament meant the power is back to th...
2      occupyparliament meant the power is back to th...
3      occupyparliament meant the power is back to th...
4      occupyparliament meant the power is back to th...
...                                                 ...
15253  get eid of the new funding model completely if...
15254  when people think they dont have a say in gove...
15255  us secret service director resigns within hrs ...
15256  the goal of the butcher was to just win  elect...
15257                                        rutomustgo

[13802 rows x 2 columns]
```

```
# Function to preprocess the text
def preprocess_text(text):
    # Tokenize the text
    tokens = word_tokenize(text)
    # Remove stopwords
    tokens = [word for word in tokens if word not in stop_words]
    tokens = [word for word in tokens if word not in swahili_slang] #
Iterate over 'tokens' to remove swahili slang
```

```
    tokens = [word for word in tokens if word not in
swahili_stopwords] # Iterate over 'tokens' to remove swahili stop
words
    # Lemmatize the tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    return ' '.join(tokens) # Return the single preprocessed text

# Preprocess the text column
tweets_df['preprocessed_text'] =
tweets_df['cleaned_text'].apply(preprocess_text)

# Display the cleaned and preprocessed data
cleaned_df = tweets_df[['cleaned_text', 'preprocessed_text', 'date']]
cleaned_df

                                            cleaned_text  \
0      occupyparliament meant the power is back to th...
1      occupyparliament meant the power is back to th...
2      occupyparliament meant the power is back to th...
3      occupyparliament meant the power is back to th...
4      occupyparliament meant the power is back to th...
...                                                  ...
15253  get eid of the new funding model completely if...
15254  when people think they dont have a say in gove...
15255  us secret service director resigns within hrs ...
15256  the goal of the butcher was to just win  elect...
15257                                         rutomustgo

                                       preprocessed_text  \
0      occupyparliament meant power back people keep ...
1      occupyparliament meant power back people keep ...
2      occupyparliament meant power back people keep ...
3      occupyparliament meant power back people keep ...
4      occupyparliament meant power back people keep ...
...                                                  ...
15253  get eid new funding model completely want u st...
15254  people think dont say government spark revolut...
15255  u secret service director resigns within hr co...
15256  goal butcher win election called president tha...
15257                                         rutomustgo

                             date
0      Jul 21, 2024 · 5:02 PM UTC
1      Jul 21, 2024 · 4:54 PM UTC
2      Jul 21, 2024 · 4:53 PM UTC
3      Jul 21, 2024 · 4:52 PM UTC
4      Jul 21, 2024 · 4:50 PM UTC
...                           ...
15253  Jul 24, 2024 · 5:40 AM UTC
```

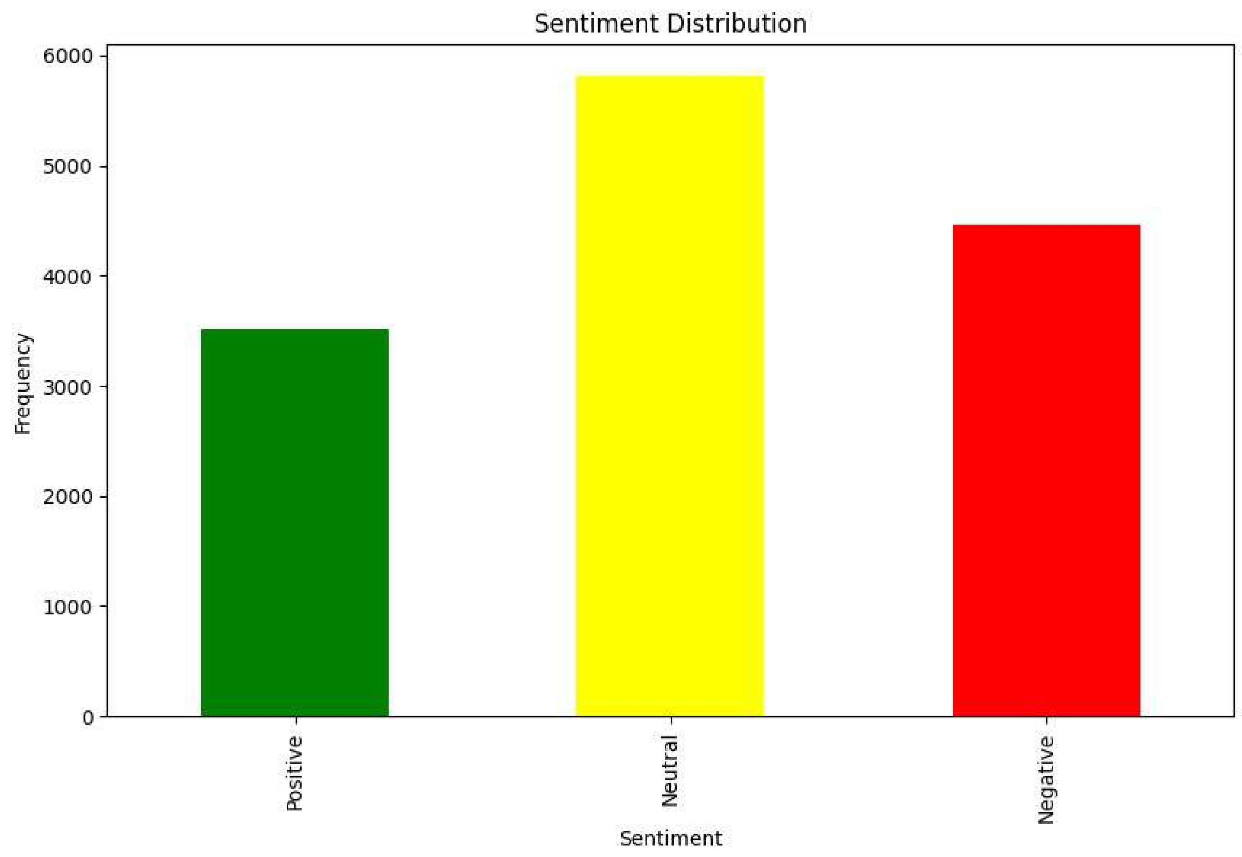
```
15254  Jul 24, 2024 · 5:39 AM UTC
15255  Jul 24, 2024 · 5:39 AM UTC
15256  Jul 24, 2024 · 5:39 AM UTC
15257  Jul 24, 2024 · 5:39 AM UTC

[13802 rows x 3 columns]

# Save the DataFrame to a CSV file
cleaned_df.to_csv('cleaned_data.csv', index=False)

data = pd.read_csv('cleaned_data.csv')

data

                                             cleaned_text  \
0      occupyparliament meant the power is back to th...
1      occupyparliament meant the power is back to th...
2      occupyparliament meant the power is back to th...
3      occupyparliament meant the power is back to th...
4      occupyparliament meant the power is back to th...
...                                                  ...
13797  get eid of the new funding model completely if...
13798  when people think they dont have a say in gove...
13799  us secret service director resigns within hrs ...
13800  the goal of the butcher was to just win  elect...
13801                                          rutomustgo

                                        preprocessed_text  \
0      occupyparliament meant power back people keep ...
1      occupyparliament meant power back people keep ...
2      occupyparliament meant power back people keep ...
3      occupyparliament meant power back people keep ...
4      occupyparliament meant power back people keep ...
...                                                  ...
13797  get eid new funding model completely want u st...
13798  people think dont say government spark revolut...
13799  u secret service director resigns within hr co...
13800  goal butcher win election called president tha...
13801                                          rutomustgo

                             date
0      Jul 21, 2024 · 5:02 PM UTC
1      Jul 21, 2024 · 4:54 PM UTC
2      Jul 21, 2024 · 4:53 PM UTC
3      Jul 21, 2024 · 4:52 PM UTC
4      Jul 21, 2024 · 4:50 PM UTC
...                           ...
13797  Jul 24, 2024 · 5:40 AM UTC
13798  Jul 24, 2024 · 5:39 AM UTC
13799  Jul 24, 2024 · 5:39 AM UTC
```

```
13800  Jul 24, 2024 · 5:39 AM UTC
13801  Jul 24, 2024 · 5:39 AM UTC

[13802 rows x 3 columns]

# Initialize sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Convert 'preprocessed_text' column to string type
data['preprocessed_text'] = data['preprocessed_text'].astype(str)

# 5. Sentiment Analysis
data['sentiments'] = data['preprocessed_text'].apply(lambda x:
sid.polarity_scores(x)['compound'])
data['sentiment_category'] = data['sentiments'].apply(lambda x:
'Positive' if x > 0 else ('Negative' if x < 0 else 'Neutral'))

data.head()

                                        cleaned_text  \
0  occupyparliament meant the power is back to th...
1  occupyparliament meant the power is back to th...
2  occupyparliament meant the power is back to th...
3  occupyparliament meant the power is back to th...
4  occupyparliament meant the power is back to th...

                                   preprocessed_text  \
0  occupyparliament meant power back people keep ...
1  occupyparliament meant power back people keep ...
2  occupyparliament meant power back people keep ...
3  occupyparliament meant power back people keep ...
4  occupyparliament meant power back people keep ...

                         date  sentiments sentiment_category
0  Jul 21, 2024 · 5:02 PM UTC         0.0            Neutral
1  Jul 21, 2024 · 4:54 PM UTC         0.0            Neutral
2  Jul 21, 2024 · 4:53 PM UTC         0.0            Neutral
3  Jul 21, 2024 · 4:52 PM UTC         0.0            Neutral
4  Jul 21, 2024 · 4:50 PM UTC         0.0            Neutral
```

- -1.0 represents very negative sentiment.
- 0.0 represents neutral sentiment.
- 1.0 represents very positive sentiment.

```
data['sentiment_category'].head(20)

0      Neutral
1      Neutral
2      Neutral
3      Neutral
4      Neutral
```

```
5      Neutral
6      Neutral
7      Neutral
8      Neutral
9      Neutral
10     Neutral
11    Positive
12     Neutral
13    Negative
14     Neutral
15    Positive
16     Neutral
17     Neutral
18    Negative
19    Positive
Name: sentiment_category, dtype: object

# 6. Visualization of sentiments
sentiments = ['Positive', 'Neutral', 'Negative']

# Plotting the sentiment distribution
plt.figure(figsize=(10, 6))

# Reorder the sentiment categories before plotting
data['sentiment_category'] =
pd.Categorical(data['sentiment_category'], categories=sentiments,
ordered=True)

# Plot the bar chart
data['sentiment_category'].value_counts().reindex(sentiments).plot(kin
d='bar', color=['green', 'yellow', 'red'])
plt.title('Sentiment Distribution')
plt.xlabel('Sentiment')
plt.ylabel('Frequency')

# Set the x-ticks to match the order
plt.xticks(range(len(sentiments)), sentiments)
plt.show()
```



The bar chart shows the distribution of sentiments (Neutral, Negative, and Positive) within the dataset. Here's a detailed interpretation:

Positive Sentiment:

- Positive sentiment is the least frequent, with a frequency of about 3,000. This indicates that positive emotions and favorable opinions are less common in the dataset compared to neutral and negative sentiments.

Neutral Sentiment:

- The majority of the data points have a neutral sentiment, with a frequency of over 6,000. This indicates that most of the content in the dataset does not express strong positive or negative emotions.

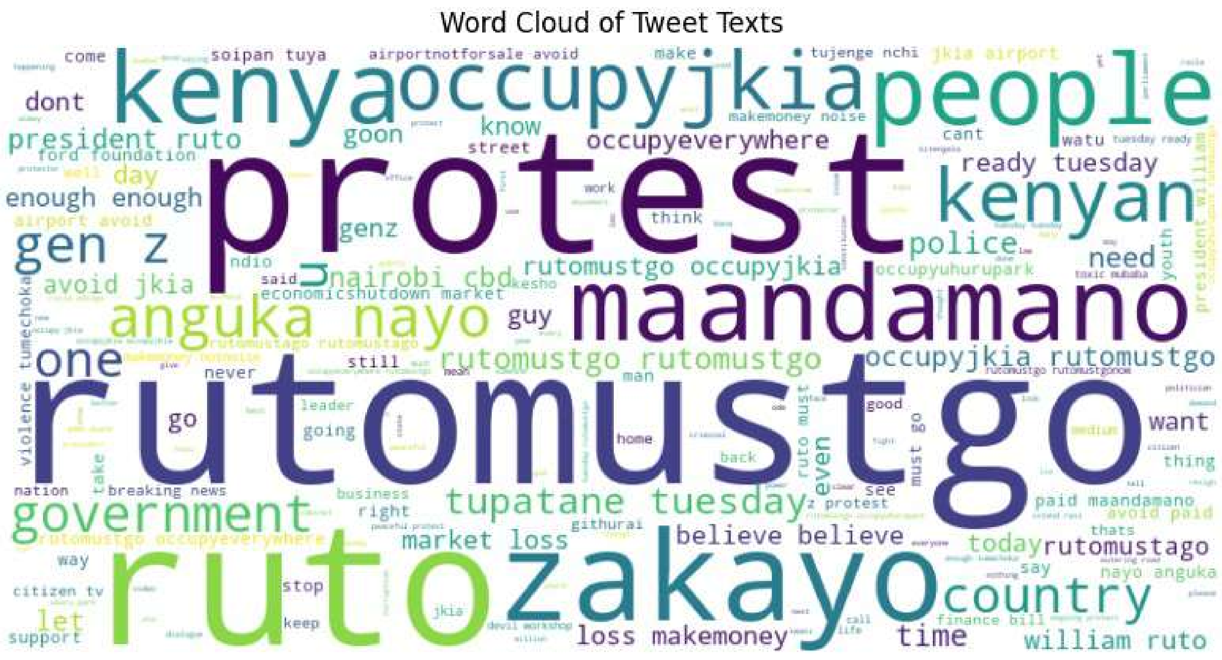
Negative Sentiment:

- The second most frequent sentiment is negative, with a frequency of around 5,000. This suggests that a significant portion of the data contains negative sentiments, reflecting dissatisfaction, criticism, or negative reactions.

#Wordcloud

```
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

```python
# Combine all text data into a single string
text_data = ' '.join(data['preprocessed_text'])

# Create the word cloud
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(text_data)

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Word Cloud of Tweet Texts")
plt.show()
```



The word cloud visualization reveals the most prominent words in the tweet texts related to the protests.

"protest": The most prominent word, indicating that protests are the main topic of discussion.

"rutomustgo" and "ruto": These terms are frequently mentioned, suggesting a significant focus on opposition to a person named Ruto.

"zakayo": Another frequently mentioned term, potentially a key figure or term related to the protests.

"occupyjkia": This term indicates a specific protest or movement focused on the Jomo Kenyatta International Airport (JKIA).

"kenya" and "kenyan": These words suggest that the protests and discussions are related to events in Kenya.

"maandamano": A Swahili word for protests, reinforcing the focus on protest activities.

"people": Indicates the involvement or focus on the people in the protest activities.

"police": Suggests discussions about law enforcement in the context of the protests.

"government": Indicates that the government is a significant topic in the context of the protests.

Contextual Words:

"occupy": Related to various locations, suggesting a strategy of occupying key areas.

"genz" and "youth": Indicating that younger generations are involved or are a topic of discussion.

"tuesday": A specific day that might be significant for the protest activities.

"time": Refers to timing, either of events or the urgency of the protest.

"let", "go", "want", "need": Words indicating demands or actions related to the protests.

Sentiment and Actions:

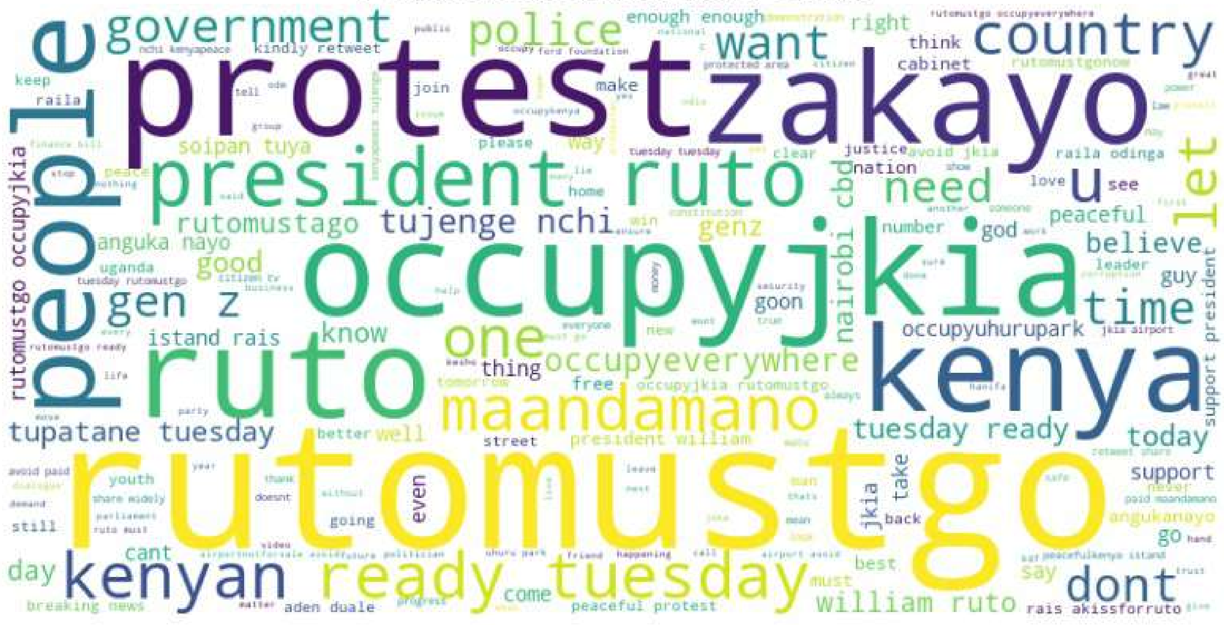
"believe", "even", "still": Reflect sentiment and the state of mind of the participants or observers.

"avoid", "stop": Indicate actions or recommendations related to the protests.

Overall, the word cloud highlights a significant amount of protest-related discussion focused on key figures, locations, and actions within Kenya, with prominent involvement of younger generations and calls for specific actions against certain entities.

```python
# Separate texts by sentiment category
positive_texts = ' '.join(data[data['sentiment_category'] == 
'Positive']['preprocessed_text'])
neutral_texts = ' '.join(data[data['sentiment_category'] == 'Neutral']
['preprocessed_text'])
negative_texts = ' '.join(data[data['sentiment_category'] == 
'Negative']['preprocessed_text'])

# Function to generate word cloud
def generate_wordcloud(text, title):
    wordcloud = WordCloud(width=800, height=400, 
background_color='white').generate(text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.title(title, fontsize=15)
    plt.show()

# Generate word clouds for each sentiment category
generate_wordcloud(positive_texts, 'Positive Sentiment Word Cloud')
```

Positive Sentiment Word Cloud



The word cloud predominantly conveys a sense of optimism, hope, and unity. The presence of words like "good", "peace", "justice", "believe", "support", and "love" reinforces this positive sentiment.

Key Themes:

Protest and Change: Words like "protest", "occupyjkia", "rutomustgo", "change", and "justice" indicate a strong focus on social and political change.

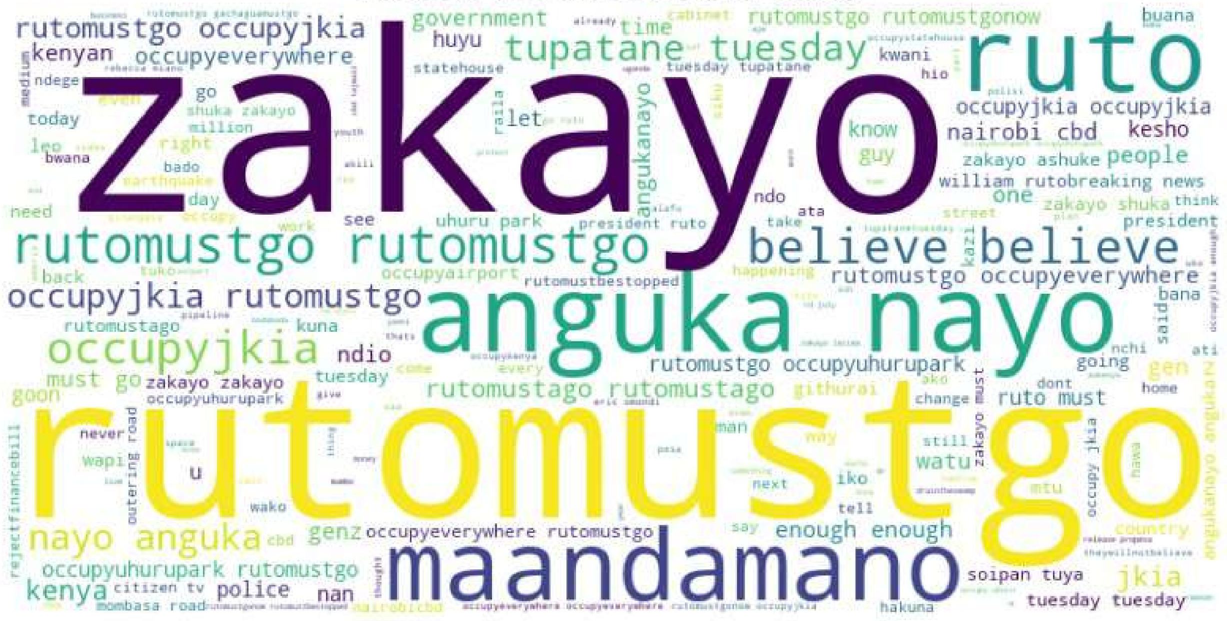
Unity and Togetherness: Terms like "people", "together", "support", "stand", and "occupyeverywhere" emphasize a sense of collective action and solidarity.

Hope and Optimism: Words like "good", "better", "peace", "hope", and "believe" reflect a positive outlook for the future.

Specific Demands: The frequent appearance of "rutomustgo" and "occupyjkia" suggests targeted demands for political change.

```
generate_wordcloud(neutral_texts, 'Neutral Sentiment Word Cloud')
```

Neutral Sentiment Word Cloud



Dominant Themes:

Protest and Discontent: Words like "occupyjkia", "rutomustgo", "maandamano" (demonstrations), "government", "president", "tuko" (we are), and "anguka nayo" (fall down) are prominent, indicating a strong focus on protest, dissent, and dissatisfaction with the government.

Time and Place: Words like "tuesday", "today", "now", "time", "nairobi", "cbd", "kenya" suggest a focus on current events and specific locations.

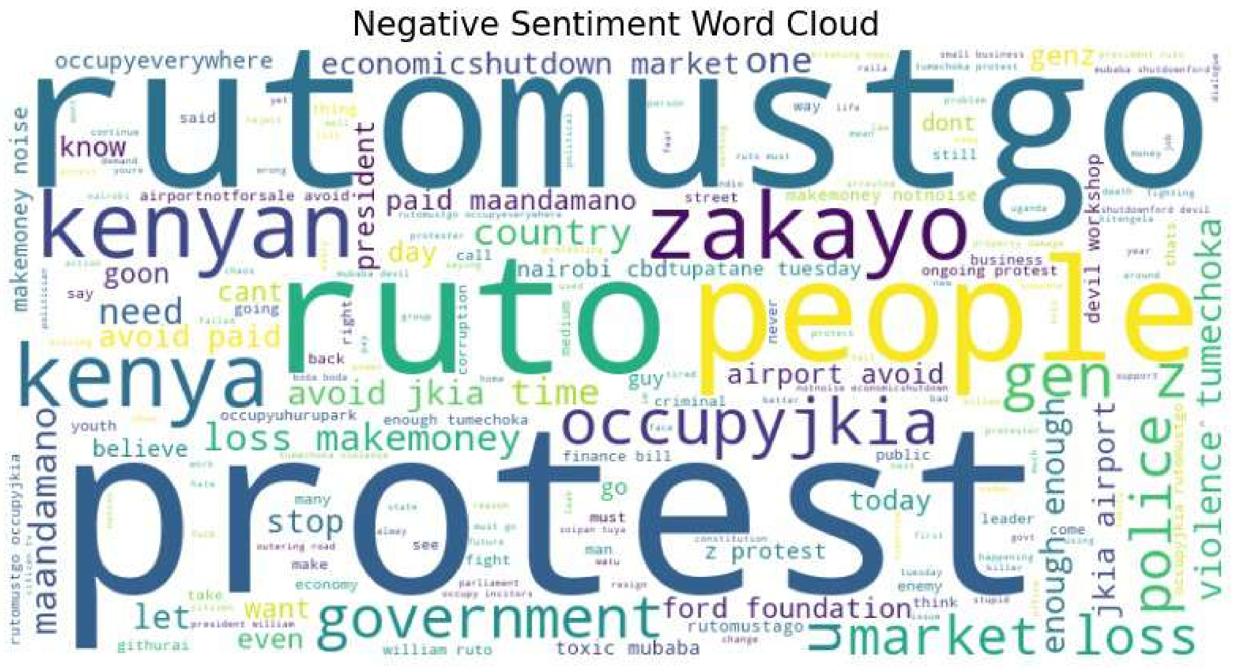
People and Unity: Terms like "people", "together", "one", "we", and "occupyeverywhere" imply a sense of collective action and unity among protesters.

Specific Demands: The repeated phrase "rutomustgo" highlights a central demand of the protests.

Overall Sentiment:

While labeled as "neutral," the word cloud leans towards a negative sentiment due to the prevalence of words associated with protest, discontent, and criticism of the government. However, the absence of overtly negative or aggressive language suggests a more measured tone compared to potentially negative or positive sentiment word clouds.

```
generate_wordcloud(negative_texts, 'Negative Sentiment Word Cloud')
```



Dominant Themes:

The word cloud overwhelmingly reflects a negative sentiment, centered around discontent, protest, and economic hardship.

Protest and Discontent: Words like "maandamano" (protests), "occupyjkia", "rutomustgo", "tumechoka" (we are tired), "fight", "violence", "enough", and "stop" are prominent, indicating a strong sense of frustration and opposition.

Economic Hardship: Terms such as "market loss", "makemoney", "economic shutdown", "loss", "finance bill", and "job" highlight the economic struggles faced by many.

Government Criticism: Words like "government", "president", "corruption", "paid", and "police" suggest a critical view of the government and its policies.

Calls to Action: Phrases like "occupyjkia", "rutomustgo", and "tupatane" (let's meet) emphasize a collective desire for change and mobilization.

Specific Issues:

The prominence of "ruto" indicate he is a central figure driving the negative sentiment.

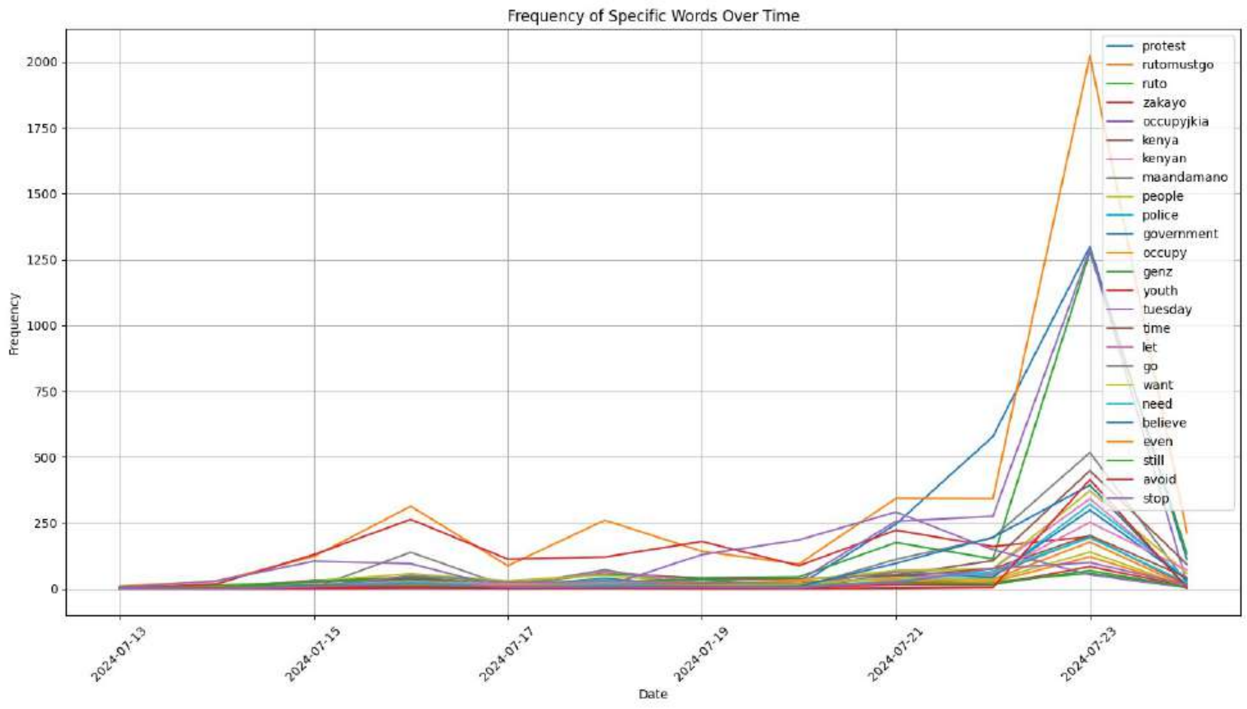
The mention of "finance bill" suggests economic policies are a significant point of contention.

Words like "violence", "goon", and "enemy" highlight the perceived threat or aggression associated with the situation. Overall Sentiment:

The word cloud paints a picture of widespread dissatisfaction, economic hardship, and a strong desire for political change. The tone is largely negative and confrontational, with a clear call to action against perceived injustices.

```
from collections import Counter
from datetime import datetime

# Convert 'date' to datetime format
data['date'] = pd.to_datetime(data['date'], format='%b %d, %Y · %I:%M
%p UTC')

# Define words of interest
words_of_interest = [
    'protest', 'rutomustgo', 'ruto', 'zakayo', 'occupyjkia', 'kenya',
'kenyan',
    'maandamano', 'people', 'police', 'government', 'occupy', 'genz',
'youth',
    'tuesday', 'time', 'let', 'go', 'want', 'need', 'believe', 'even',
'still',
    'avoid', 'stop'
]

# Function to count occurrences of words
def count_words(text, words):
    word_counts = Counter(word for word in text.split() if word in
words)
    return dict(word_counts)

# Count occurrences of words for each row
data['word_counts'] = data['preprocessed_text'].apply(lambda x:
count_words(x, words_of_interest))

# Create a DataFrame from the word counts
word_counts_df = pd.DataFrame(data['word_counts'].tolist(),
index=data.index).fillna(0)

# Merge the word counts DataFrame with the original data
data = pd.concat([data, word_counts_df], axis=1)

# Aggregate data by date with explicit numeric_only parameter
daily_word_counts =
data.groupby(data['date'].dt.date).sum(numeric_only=True).reset_index(
)

# Plotting the data
plt.figure(figsize=(14, 8))

for word in words_of_interest:
    if word in daily_word_counts.columns:
        plt.plot(daily_word_counts['date'], daily_word_counts[word],
label=word)

plt.title('Frequency of Specific Words Over Time')
plt.xlabel('Date')
plt.ylabel('Frequency')
```

```python
plt.legend(loc='upper right')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



The line plot visualizes the frequency of specific words over a period from approximately July 13th to July 23rd, 2024.

Key Findings

Dominant Words: The words "protest," "rutomustgo," "ruto," and "occupyjkia" exhibit significantly higher frequencies compared to other words, suggesting they were central topics during this period.

Fluctuating Frequencies: Most words exhibit fluctuations in frequency over time, indicating varying levels of usage.

Peak Usage: Some words, like "protest" and "rutomustgo," experienced sharp peaks, suggesting specific events or trends drove their increased usage.

Correlated Trends: Certain words, such as "rutomustgo" and "occupyjkia," tend to move together, indicating a potential relationship or shared context.

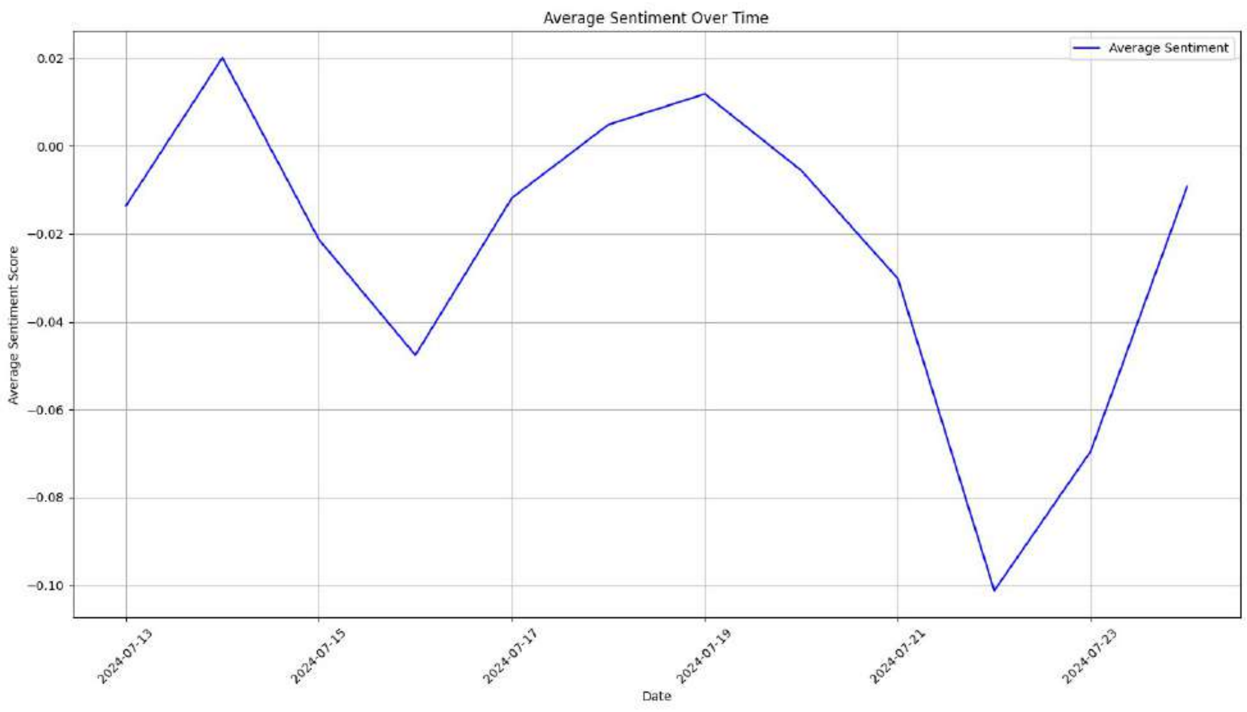
Emerging Trends: Words like "genz" and "youth" show an upward trend, suggesting growing relevance or participation of younger demographics.

```python
# Aggregate sentiment scores by date
daily_sentiments = data.groupby(data['date'].dt.date)
```

```
['sentiments'].mean().reset_index()

# Check the resulting DataFrame structure
print(daily_sentiments.head())
```

```
         date  sentiments
0  2024-07-13   -0.013648
1  2024-07-14    0.020110
2  2024-07-15   -0.021281
3  2024-07-16   -0.047621
4  2024-07-17   -0.011833
```

```
# Plotting the sentiment scores over time
plt.figure(figsize=(14, 8))

plt.plot(daily_sentiments['date'], daily_sentiments['sentiments'],
label='Average Sentiment', color='blue')

plt.title('Average Sentiment Over Time')
plt.xlabel('Date')
plt.ylabel('Average Sentiment Score')
plt.legend(loc='upper right')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



The line plot illustrates the fluctuation of average sentiment over a specific period, ranging from approximately July 13th to July 23rd, 2024. The sentiment score appears to be a numerical value, with positive values indicating more positive sentiment and negative values representing more negative sentiment.

Key Observations:

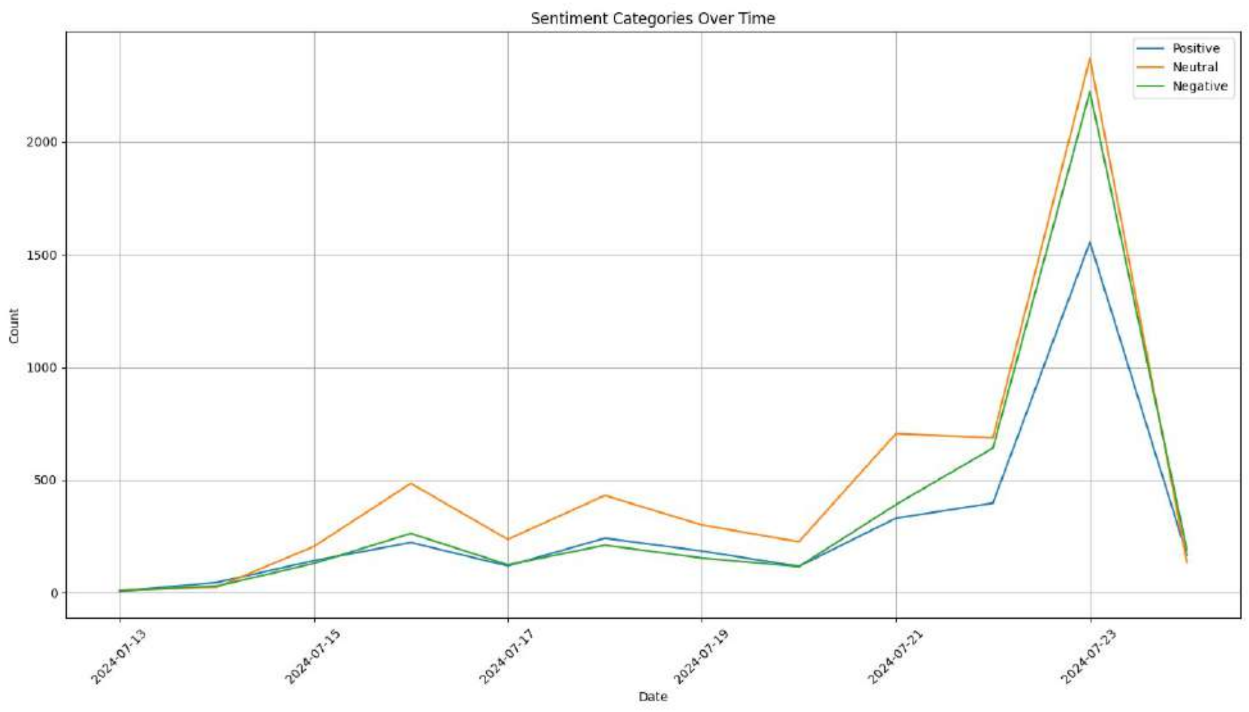
Fluctuating Sentiment: The average sentiment score exhibits significant fluctuations over the analyzed period, indicating shifts in overall sentiment.

Negative Bias: The majority of the data points lie below the zero line, suggesting a generally negative sentiment during this time.

Extreme Points: There are instances of both highly positive and highly negative sentiment scores, indicating periods of strong emotional reactions.

General Trend: While there are fluctuations, there doesn't seem to be a clear overall upward or downward trend in sentiment.

```
# Aggregate counts of sentiment categories by date
daily_sentiment_counts = data.groupby([data['date'].dt.date,
'sentiment_category']).size().unstack(fill_value=0).reset_index()

# Check the resulting DataFrame structure
print(daily_sentiment_counts.head())

sentiment_category        date  Positive  Neutral  Negative
0                   2024-07-13         6        9         8
1                   2024-07-14        45       24        29
2                   2024-07-15       141      204       129
3                   2024-07-16       222      484       262
4                   2024-07-17       118      236       123

# Plotting the sentiment categories over time
plt.figure(figsize=(14, 8))

# Plot each sentiment category
for sentiment in ['Positive', 'Neutral', 'Negative']:
    if sentiment in daily_sentiment_counts.columns:
        plt.plot(daily_sentiment_counts['date'],
daily_sentiment_counts[sentiment], label=sentiment)

plt.title('Sentiment Categories Over Time')
plt.xlabel('Date')
plt.ylabel('Count')
plt.legend(loc='upper right')
plt.grid(True)
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



The graph illustrates the distribution of sentiment categories (positive, neutral, and negative) over a specific time period, ranging from July 13th to July 23rd, 2024. The y-axis represents the count of sentiments, while the x-axis shows the date.

Key Findings:

Dominance of Negative Sentiment: The "Negative" line consistently occupies the highest position on the graph, indicating that negative sentiment was the most prevalent throughout the analyzed period.

Fluctuating Trends: All three sentiment categories exhibit fluctuations over time, with peaks and troughs in their respective counts.

Sharp Increase Around July 22nd: A notable spike is observed in all sentiment categories, particularly negative, around July 22nd, suggesting a significant event or trend impacted sentiment during this time.

Relative Proportions: The relative proportions of positive, neutral, and negative sentiments vary over time. While negative sentiment is dominant, there are periods where neutral or even positive sentiment increases.

Topic Modeling

```
# Vectorize the text data
vectorizer = CountVectorizer(max_df=0.95, min_df=2,
stop_words=['english', 'swahili_stopwords', 'swahili_slang'])
doc_term_matrix = vectorizer.fit_transform(data['preprocessed_text'])
```

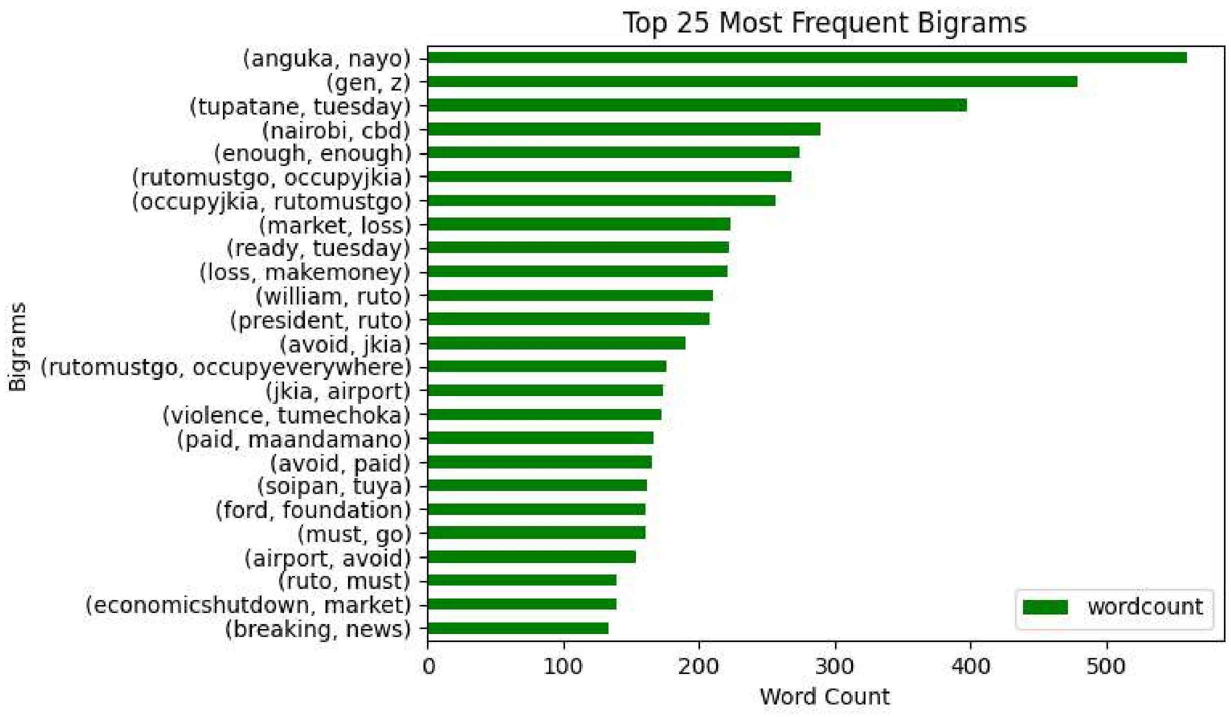
```
# Initialize LDA model
lda = LatentDirichletAllocation(n_components=5, random_state=42)
lda.fit(doc_term_matrix)
```

```
LatentDirichletAllocation(n_components=5, random_state=42)
```

```
# Function to display topics
def display_topics(model, feature_names, no_top_words):
    for topic_idx, topic in enumerate(model.components_):
        print(f"Topic {topic_idx+1}:")
        print(" ".join([feature_names[i] for i in topic.argsort()[:-
no_top_words - 1:-1]]))

no_top_words = 10
display_topics(lda, vectorizer.get_feature_names_out(), no_top_words)
```

```
Topic 1:
zakayo president kenya ruto let occupyuhurupark protest raila peace
rutos
Topic 2:
rutomustgo believe occupyjkia people occupyeverywhere zakayo protest
kenyan ruto one
Topic 3:
ruto tuesday rutomustgo nayo anguka rutomustago tupatane occupyjkia
ready enough
Topic 4:
protest occupyjkia loss road nairobi economicshutdown market makemoney
cbd economic
Topic 5:
protest jkia maandamano avoid paid enough airport government police
area
```

```
# N-grams Analysis
# Function to generate n-grams
def generate_ngrams(clean_text, n):
    words = clean_text.split()
    return list(ngrams(words, n))

# Bigrams
freq_dict = defaultdict(int)
for sent in data["preprocessed_text"]:
    for word in generate_ngrams(sent, 2):
        freq_dict[word] += 1

fd_sorted_bigrams = pd.DataFrame(sorted(freq_dict.items(), key=lambda
x: x[1], reverse=True))
fd_sorted_bigrams.columns = ["word", "wordcount"]

# Plot the top 25 most frequent bigrams
def horizontal_bar_chart(data, color, title):
    data.plot(kind='barh', x='word', y='wordcount', color=color)
    plt.xlabel('Word Count')
    plt.ylabel('Bigrams')
    plt.title(title)
    plt.gca().invert_yaxis()  # Invert y-axis to have the highest
count on top
    plt.show()

horizontal_bar_chart(fd_sorted_bigrams.head(25), 'green', 'Top 25 Most
Frequent Bigrams')
```



The bar chart shows the top 25 most frequent bigrams (two-word combinations) in the dataset. Here’s a detailed interpretation:

High Frequency Bigrams:

- The most frequent bigram is "(anguka, nayo)" with around 600 occurrences. This suggests that this phrase is very commonly discussed in the dataset.
- Other high-frequency bigrams include "(gen, z)", "(tupatane, tuesday)", and "(occupyjkia, rutomustgo)", each with significant counts, indicating these terms are prevalent in the discussions.

Protest and Movement Keywords:

- Bigrams like "(tupatane, tuesday)" and "(occupyjkia, rutomustgo)" suggest organized protest movements and specific days for action.
- "(rutomustgo, occupyjkia)" and "(rutomustgo, occupyeverywhere)" indicate coordinated efforts under the banner "Ruto Must Go," focusing on occupying specific

places like JKIA (Jomo Kenyatta International Airport). Political Figures and Contexts:

- Bigrams such as "(william, ruto)", "(president, ruto)", and "(president, william)" point to discussions about President William Ruto.
- "(soipan, tuya)" is another political figure frequently mentioned. Economic and Social Issues:
- "(market, loss)" and "(loss, makemoney)" indicate ongoing discussions about economic issues and financial losses.
- "(violence, tumechoka)" (translated to "violence, we are tired") reflects social discontent and fatigue with ongoing violence.

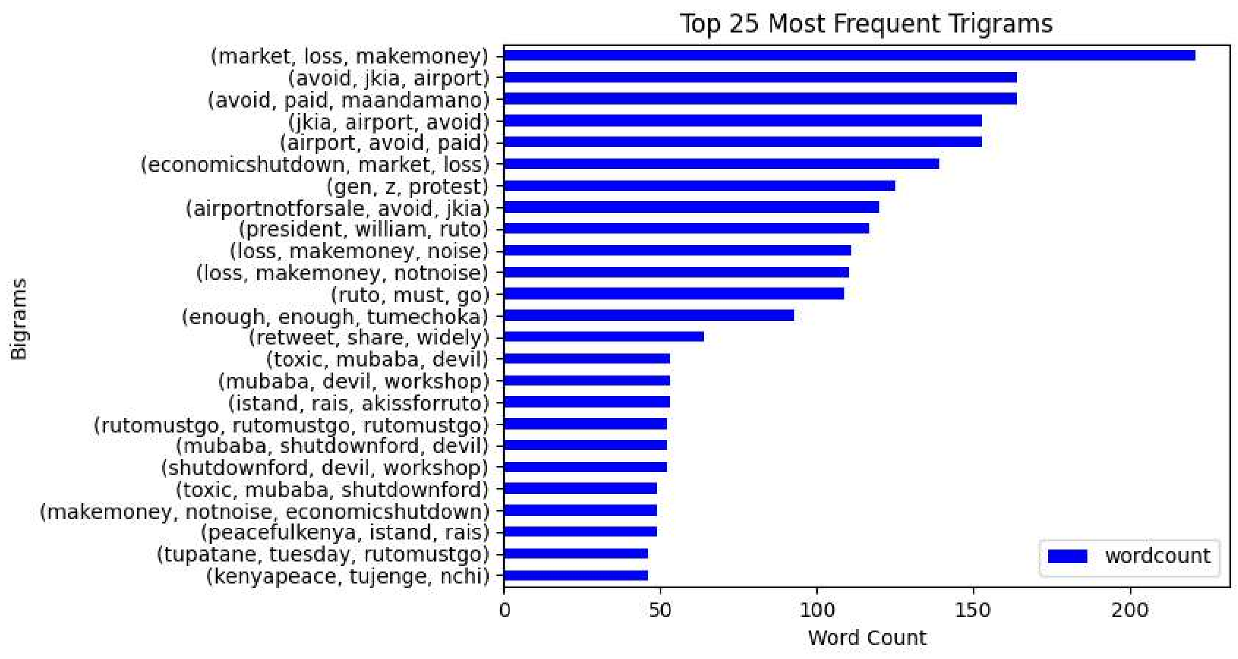
Location-Specific Mentions:

- "(nairobi, cbd)" and "(jkia, airport)" highlight specific locations frequently mentioned in the dataset.

Coordinated Messaging and Reactions:

- Repetitive bigrams like "(anguka, nayo)", "(tupatane, tuesday)", and "(ready, tuesday)" suggest coordinated messaging around specific events or actions.
- "(breaking, news)" implies frequent mentions of recent or significant events.

```
# Trigrams
freq_dict = defaultdict(int)
for sent in data["preprocessed_text"]:
    for word in generate_ngrams(sent, 3):
        freq_dict[word] += 1

fd_sorted_trigrams = pd.DataFrame(sorted(freq_dict.items(), key=lambda
x: x[1], reverse=True))
fd_sorted_trigrams.columns = ["word", "wordcount"]

# Plot the top 25 most frequent trigrams
horizontal_bar_chart(fd_sorted_trigrams.head(25), 'blue', 'Top 25 Most
Frequent Trigrams')
```



The bar chart shows the top 25 most frequent trigrams (three-word combinations) in the dataset. Here's a detailed interpretation:

Interpretation High Frequency Trigrams:

1. The most frequent trigram is "(market, loss, makemoney)" with over 200 occurrences. This suggests that discussions about market losses and making money are prevalent in the dataset.
2. Other high-frequency trigrams include "(avoid, jkia, airport)", "(avoid, paid, maandamano)", and "(jkia, airport, avoid)", each with over 150 occurrences. This indicates a significant amount of discussion around avoiding the JKIA airport and associated events.

Economic and Political Discussions:

Trigrams like "(economicshutdown, market, loss)", "(president, william, ruto)", and "(gen, z, protest)" point to discussions about economic shutdowns, market losses, the president (William Ruto), and protests involving Generation Z.

Specific Issues and Movements:

1. The presence of trigrams like "(rutomustgo, ruto, must, go)" and "(retweet, share, widely)" suggests organized movements and calls to action, likely against political figures or policies.
2. "(airportnotforsale, avoid, jkia)" indicates a specific campaign or protest related to the sale of an airport. Sentiment and Reaction:

- Trigrams such as "(enough, enough, tumechoka)" (translated to "enough, enough, we are tired") and "(kindly, retweet, share)" reflect a collective sentiment of exhaustion and calls for widespread sharing and support. Repeated and Coordinated Messaging:

- Repeated trigrams like "(rutomustgo, rutomustgo, rutomustgo)" and "(toxic, mubaba, devil)" suggest coordinated messaging and repetitive emphasis on certain themes.