



ALLIANCE
UNIVERSITY

*Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi*

PROJECT REPORT

BACHELOR OF COMPUTER APPLICATION

SEMESTER – II

INTRODUCTION TO DATASCIENCE

Political Media Messaging & Bias Detection

Dataset: Political-media.

BY

THUKIVAKAM CHATURYA

2411021240045

DEPARTMENT OF COMPUTER APPLICATION

ALLIANCE UNIVERSITY

CHANDPURA ANEKAL MAIN, ANEKAL

BANGALORE-562106

APRIL – 2025

Political Media Messaging & Bias Detection

Dataset: Political-media.

NAME: THUKIVAKAM CHATURYA

Registration no.: 2411021240045

GITHUB REPOSITORY LINK: <https://github.com/chaturya0229/Political-Media-Dataset-Data-Science-Project>

Table of Contents

Welcome to the project notebook! Below is the structured flow of analysis covered in this project.

1.  Project Title and Info
2.  Project Overview
3.  Project Goal
4.  Challenges Faced
5.  Import Libraries & Load Dataset
6.  Data Preprocessing
7.  Exploratory Data Analysis (EDA)
8.  Univariate and Multivariate Analysis
9.  Probability & Hypothesis Testing
10.  Random Forest Classification (Categorical Analysis)
11.  Model Evaluation
12.  Final Conclusion

Project Title and Info

Political Media Messaging & Bias Detection

 **Course:** Introduction to Data Science

Semester: 2nd Semester – BCA

Project Type: Beginner-level Data Science & Machine Learning

Tools Used: Python, Pandas, Seaborn, Scikit-learn

Dataset: Bank Churners Dataset

Project Overview

We'll perform:

- Statistical analysis of message types and political bias
- Hypothesis testing (Chi-square test)
- Classification model to predict bias based on message features

Project Goal

- Discover patterns in political content
- Test if bias and message types are statistically related
- Build a classification model to predict political bias

! Challenges Faced

- Imbalanced classes (more “neutral” than “partisan”)
- Qualitative text data needs encoding or transformation
- Limited numeric features for regression

1. Business Understanding

Goal: Analyze and model relationships between message types, political bias, and social media usage.

Import Libraries & Load Dataset

Standard imports for data analysis and visualization.

```

import pandas as pd
import numpy as np #Keeps only useful features like bias, message, source, etc.
    • Ensures no missing values remain.
import seaborn as sns
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from scipy import stats
from scipy.stats import norm, ttest_ind, chi2_contingency
# 2. Load the Dataset
df = pd.read_csv(r"C:\Users\thuki\Downloads\Political-media-DFE.csv")
df

      index _unit_id _golden _unit_state _trusted_judgments \
0          0 766192484 False  finalized             1
1          1 766192485 False  finalized             1
2          2 766192486 False  finalized             1
3          3 766192487 False  finalized             1
4          4 766192488 False  finalized             1
...
4995     4995 766197482 False  finalized             1
4996     4996 766197483 False  finalized             1
4997     4997 766197484 False  finalized             1
4998     4998 766197485 False  finalized             1
4999     4999 766197486 False  finalized             1

      _last_judgment_at audience audience:confidence      bias \
0        8/4/15 21:17   national           1.0  partisan
1        8/4/15 21:20   national           1.0  partisan
2        8/4/15 21:14   national           1.0  neutral
3        8/4/15 21:08   national           1.0  neutral
4        8/4/15 21:26   national           1.0  partisan
...
4995     8/5/15 6:05   national           1.0  partisan
4996     8/5/15 5:57   national           1.0  partisan
4997     8/5/15 5:10   national           1.0  neutral
4998     8/5/15 6:08   national           1.0  neutral
4999     8/5/15 5:04   national           1.0  neutral

      bias:confidence ... orig_golden audience_gold bias_gold bioid \
0            1.0 ...           NaN        NaN       NaN R000596
1            1.0 ...           NaN        NaN       NaN M000355
2            1.0 ...           NaN        NaN       NaN S001180
3            1.0 ...           NaN        NaN       NaN C000880
4            1.0 ...           NaN        NaN       NaN U000038
...
4995         1.0 ...           NaN        NaN       NaN Y000065
4996         1.0 ...           NaN        NaN       NaN Y000065

```

4997	1.0	...	NaN	NaN	NaN	Y000065
4998	1.0	...	NaN	NaN	NaN	Y000065
4999	1.0	...	NaN	NaN	NaN	Y000065
			embed \			
0	<blockquote class="twitter-tweet" width="450">...					
1	<blockquote class="twitter-tweet" width="450">...					
2	<blockquote class="twitter-tweet" width="450">...					
3	<blockquote class="twitter-tweet" width="450">...					
4	<blockquote class="twitter-tweet" width="450">...					
...			...			
4995	<div id="fb-root"></div> <script>(function(d, ...					
4996	<div id="fb-root"></div> <script>(function(d, ...					
4997	<div id="fb-root"></div> <script>(function(d, ...					
4998	<div id="fb-root"></div> <script>(function(d, ...					
4999	<div id="fb-root"></div> <script>(function(d, ...					
		id \				
0		3.83249E+17				
1		3.11208E+17				
2		3.39069E+17				
3		2.98528E+17				
4		4.07643E+17				
...		...				
4995	563532937006022_939071892785456					
4996	563532937006022_936080056417973					
4997	563532937006022_905547326137913					
4998	563532937006022_10101529848812566					
4999	563532937006022_730010240358290					
		label message_gold				source
\						
0	From: Trey Radel (Representative from Florida)				NaN	twitter
1	From: Mitch McConnell (Senator from Kentucky)				NaN	twitter
2	From: Kurt Schrader (Representative from Oregon)				NaN	twitter
3	From: Michael Crapo (Senator from Idaho)				NaN	twitter
4	From: Mark Udall (Senator from Colorado)				NaN	twitter
...	
4995	From: Ted Yoho (Representative from Florida)				NaN	facebook
4996	From: Ted Yoho (Representative from Florida)				NaN	facebook
4997	From: Ted Yoho (Representative from Florida)				NaN	facebook
4998	From: Ted Yoho (Representative from Florida)				NaN	facebook
4999	From: Ted Yoho (Representative from Florida)				NaN	facebook
		text				
0	RT @nowthisnews: Rep. Trey Radel (R- #FL) slam...					
1	VIDEO - #Obamacare: Full of Higher Costs and ...					
2	Please join me today in remembering our fallen...					
3	RT @SenatorLeahy: 1st step toward Senate debat...					

```

4      .@amazon delivery #drones show need to update ...
...
4995 I applaud Governor Perry's recent decision t...
4996 Today, I voted in favor of H.R. 5016 - Financi...
4997 (Taken from posted WOKV interview) Congressm...
4998 Join me next week for a town hall in Ocala! I'...
4999 Foreign Affairs Committee Hearing on Syria. I ...

```

[5000 rows x 22 columns]

✍ Data Preprocessing

- Keeps only useful features like bias, message, source, etc.
- Ensures no missing values remain.

```

print(df.shape)
print(df.columns)
print(df.head())

(5000, 22)
Index(['index', '_unit_id', '_golden', '_unit_state', '_trusted_judgments',
       '_last_judgment_at', 'audience', 'audience:confidence', 'bias',
       'bias:confidence', 'message', 'message:confidence', 'orig_golden',
       'audience_gold', 'bias_gold', 'bioid', 'embed', 'id', 'label',
       'message_gold', 'source', 'text'],
      dtype='object')
   index _unit_id _golden _unit_state _trusted_judgments \
0      0  766192484    False  finalized             1
1      1  766192485    False  finalized             1
2      2  766192486    False  finalized             1
3      3  766192487    False  finalized             1
4      4  766192488    False  finalized             1

   _last_judgment_at audience audience:confidence      bias bias:confidence \
0      8/4/15 21:17    national                  1.0  partisan        1.0
1      8/4/15 21:20    national                  1.0  partisan        1.0
2      8/4/15 21:14    national                  1.0  neutral         1.0
3      8/4/15 21:08    national                  1.0  neutral         1.0
4      8/4/15 21:26    national                  1.0  partisan        1.0

   ... orig_golden audience_gold bias_gold     bioid \
0     ...          NaN        NaN        NaN  R000596
1     ...          NaN        NaN        NaN  M000355
2     ...          NaN        NaN        NaN  S001180
3     ...          NaN        NaN        NaN  C000880
4     ...          NaN        NaN        NaN  U000038

```

```

          embed      id \
0 <blockquote class="twitter-tweet" width="450">... 3.83249E+17
1 <blockquote class="twitter-tweet" width="450">... 3.11208E+17
2 <blockquote class="twitter-tweet" width="450">... 3.39069E+17
3 <blockquote class="twitter-tweet" width="450">... 2.98528E+17
4 <blockquote class="twitter-tweet" width="450">... 4.07643E+17

           label message_gold   source \
0  From: Trey Radel (Representative from Florida)      NaN  twitter
1  From: Mitch McConnell (Senator from Kentucky)      NaN  twitter
2  From: Kurt Schrader (Representative from Oregon)      NaN  twitter
3  From: Michael Crapo (Senator from Idaho)      NaN  twitter
4  From: Mark Udall (Senator from Colorado)      NaN  twitter

        text
0 RT @nowthisnews: Rep. Trey Radel (R- #FL) slam...
1 VIDEO - #Obamacare: Full of Higher Costs and ...
2 Please join me today in remembering our fallen...
3 RT @SenatorLeahy: 1st step toward Senate debat...
4 .@amazon delivery #drones show need to update ...

[5 rows x 22 columns]

# Drop unnecessary columns
df_clean = df.drop(columns=[
    'index', '_unit_id', '_golden', '_unit_state', '_trusted_judgments',
    '_last_judgment_at', 'orig_golden', 'audience_gold', 'bias_gold',
    'message_gold', 'bioid', 'embed', 'id', 'label'
])

# Drop nulls if any
df_clean.dropna(inplace=True)

# Check for missing values
print(df_clean.isnull().sum())

# Drop rows with any missing values (if minimal)
df_clean = df_clean.dropna()

audience          0
audience:confidence 0
bias              0
bias:confidence   0
message           0
message:confidence 0
source            0
text              0
dtype: int64

```

```

# Confirm cleaned dataset
print(df_clean[['message', 'source', 'bias', 'message', 'source', 'bias']].head())

```

	message	source	bias	message	source	bias
0	policy	twitter	partisan	policy	twitter	partisan
1	attack	twitter	partisan	attack	twitter	partisan
2	support	twitter	neutral	support	twitter	neutral
3	policy	twitter	neutral	policy	twitter	neutral
4	policy	twitter	partisan	policy	twitter	partisan

```

df_clean.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   audience         5000 non-null    object  
 1   audience:confidence  5000 non-null    float64 
 2   bias              5000 non-null    object  
 3   bias:confidence   5000 non-null    float64 
 4   message           5000 non-null    object  
 5   message:confidence  5000 non-null    float64 
 6   source            5000 non-null    object  
 7   text               5000 non-null    object  
dtypes: float64(3), object(5)
memory usage: 312.6+ KB

```

Univariate and Multivariate Analysis

◊ Univariate Analysis

Univariate analysis is a statistical examination of a single variable at the time.it helps to describe the basic characteristics of the data, such as its distribution, central tendency(mean,median,mode),and description(range,variance,standard deviation).

◊ Multivariate Analysis

Multivariate analysis is a statistical technique used to examine the relationship between three or more variables at the same time. it helps to identify patterns, correlations and interactions among variables to understand how they influence each other.

```

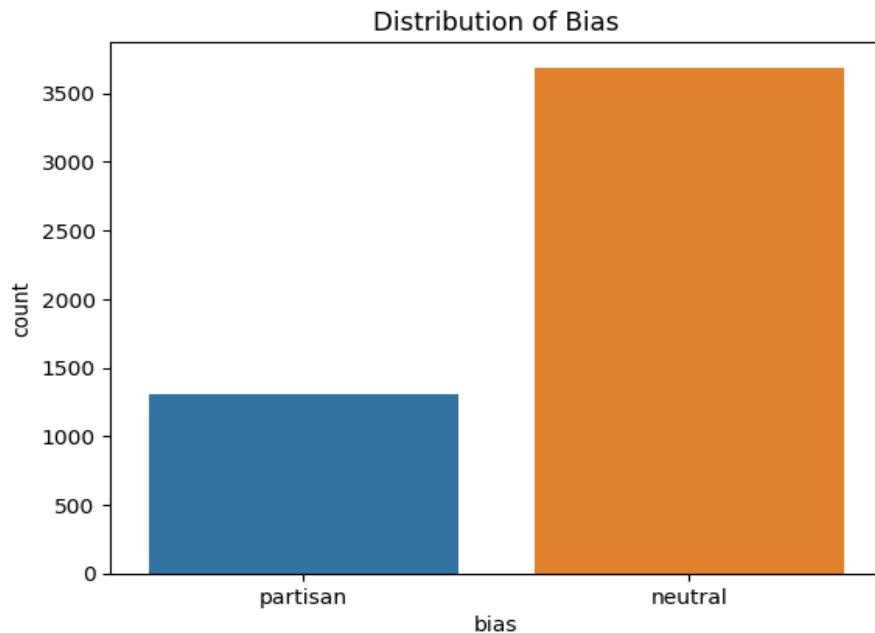
sns.countplot(data=df_clean, x='bias',hue="bias")
plt.title('Distribution of Bias')
plt.show()

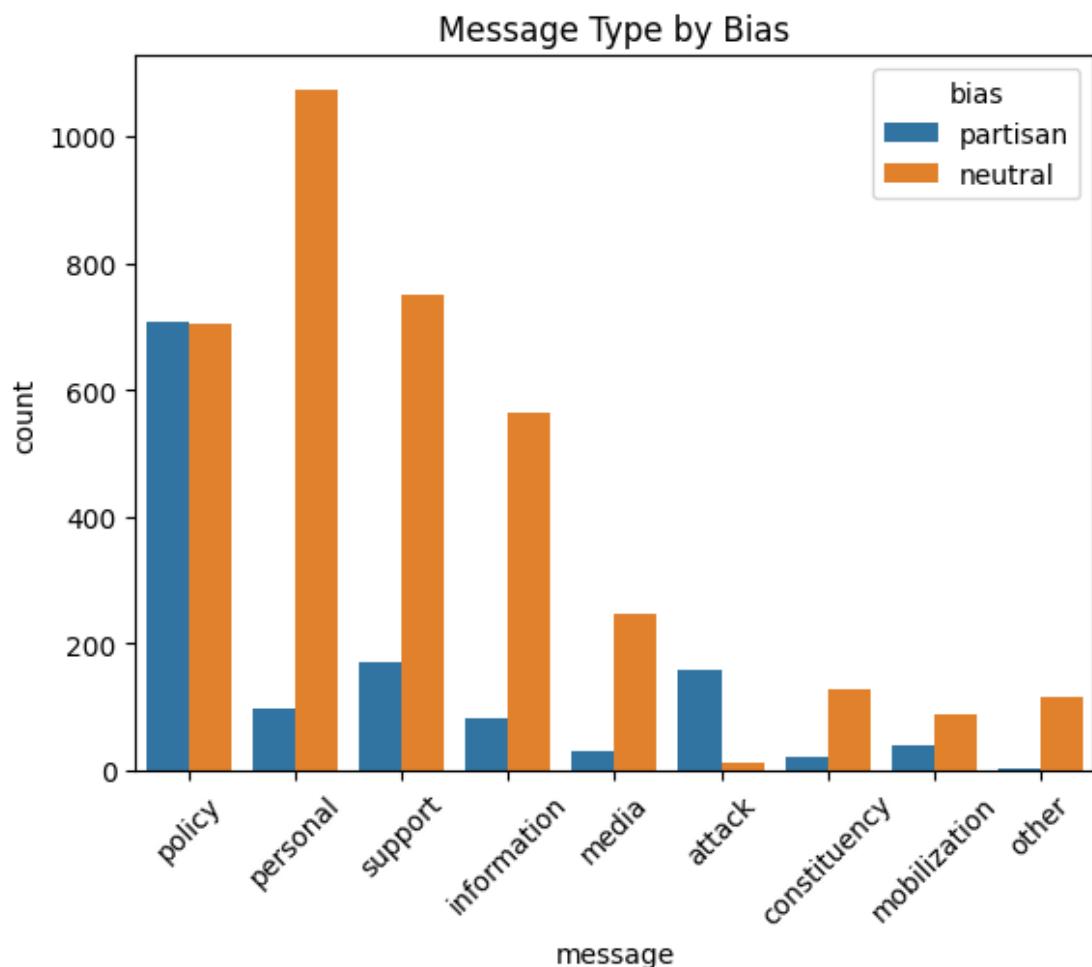
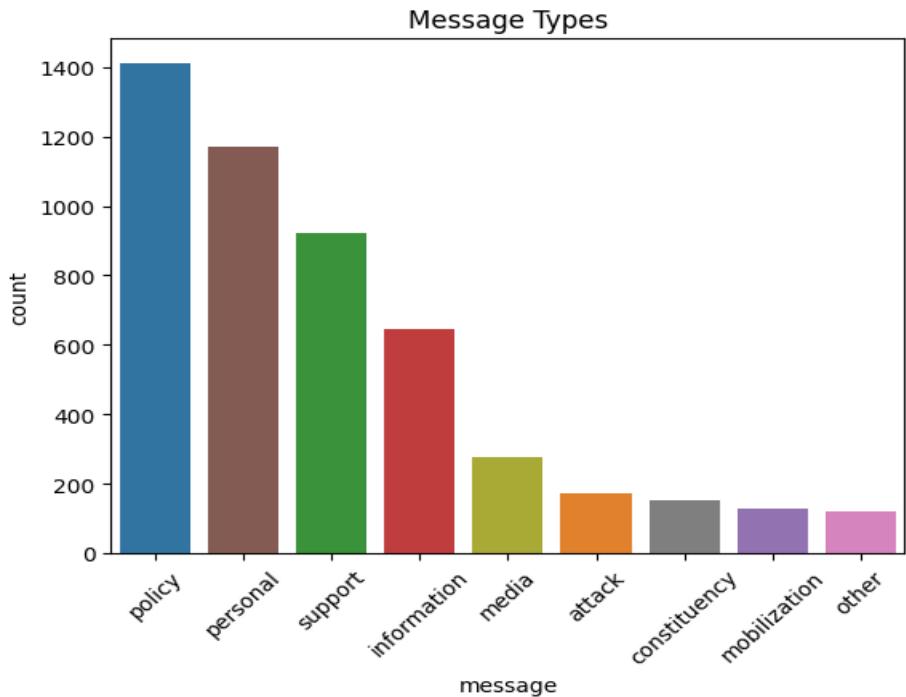
sns.countplot(data=df_clean, x='message',hue="message", order=df_clean['message'].value_counts().index)

```

```
plt.title('Message Types')
plt.xticks(rotation=45)
plt.show()

sns.countplot(data=df_clean, x='message', hue='bias', order=df_clean['message'].value_counts().index)
plt.title('Message Type by Bias')
plt.xticks(rotation=45)
plt.show()
```

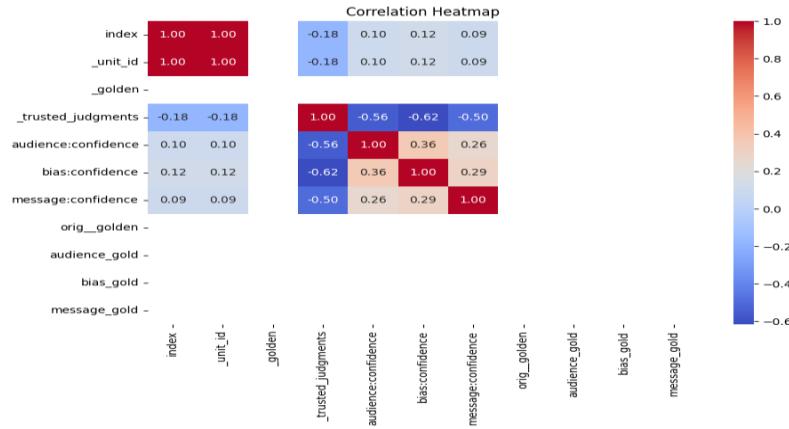




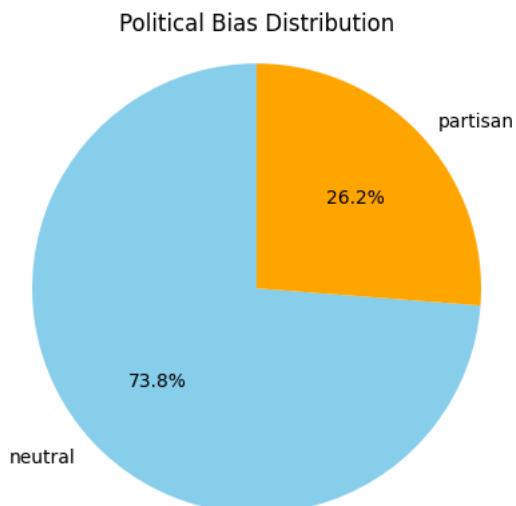
```
numeric_df = df.select_dtypes(exclude="object")
```

🔥 Heatmap - Correlation Matrix

```
# Plot heatmap to visualize correlation between numerical features
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



```
# Pie chart for bias
bias_counts = df_clean['bias'].value_counts()
plt.pie(bias_counts, labels=bias_counts.index, autopct='%1.1f%%', startangle=90, colors=['skyblue', 'orange'])
plt.title("Political Bias Distribution")
plt.axis('equal') # Equal aspect ratio ensures the pie is circular.
plt.show()
```

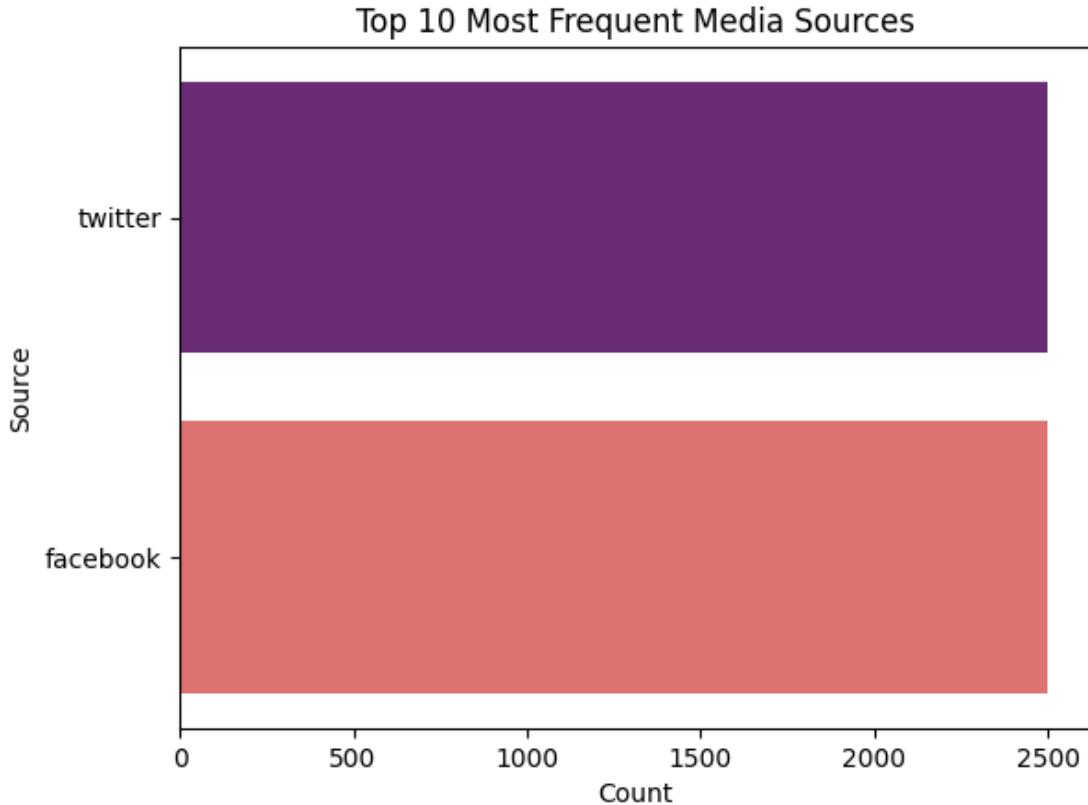


```

top_sources = df_clean['source'].value_counts().head(10)

# Bar plot of top sources
sns.barplot(x=top_sources.values, y=top_sources.index, palette='magma', hue=to_p_sources.index)
plt.title("Top 10 Most Frequent Media Sources")
plt.xlabel("Count")
plt.ylabel("Source")
plt.show()

```

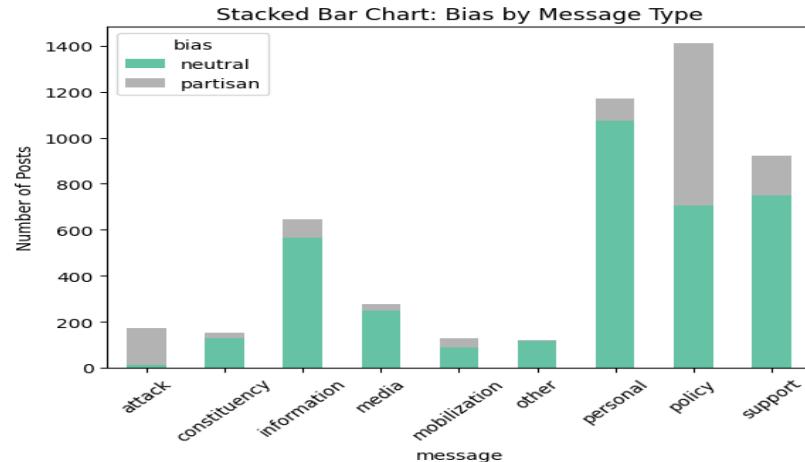


```

# Grouped bias by message
msg_bias = pd.crosstab(df_clean['message'], df_clean['bias'])

# Stacked bar chart
msg_bias.plot(kind='bar', stacked=True, colormap='Set2')
plt.title("Stacked Bar Chart: Bias by Message Type")
plt.ylabel("Number of Posts")
plt.xticks(rotation=45)
plt.show()

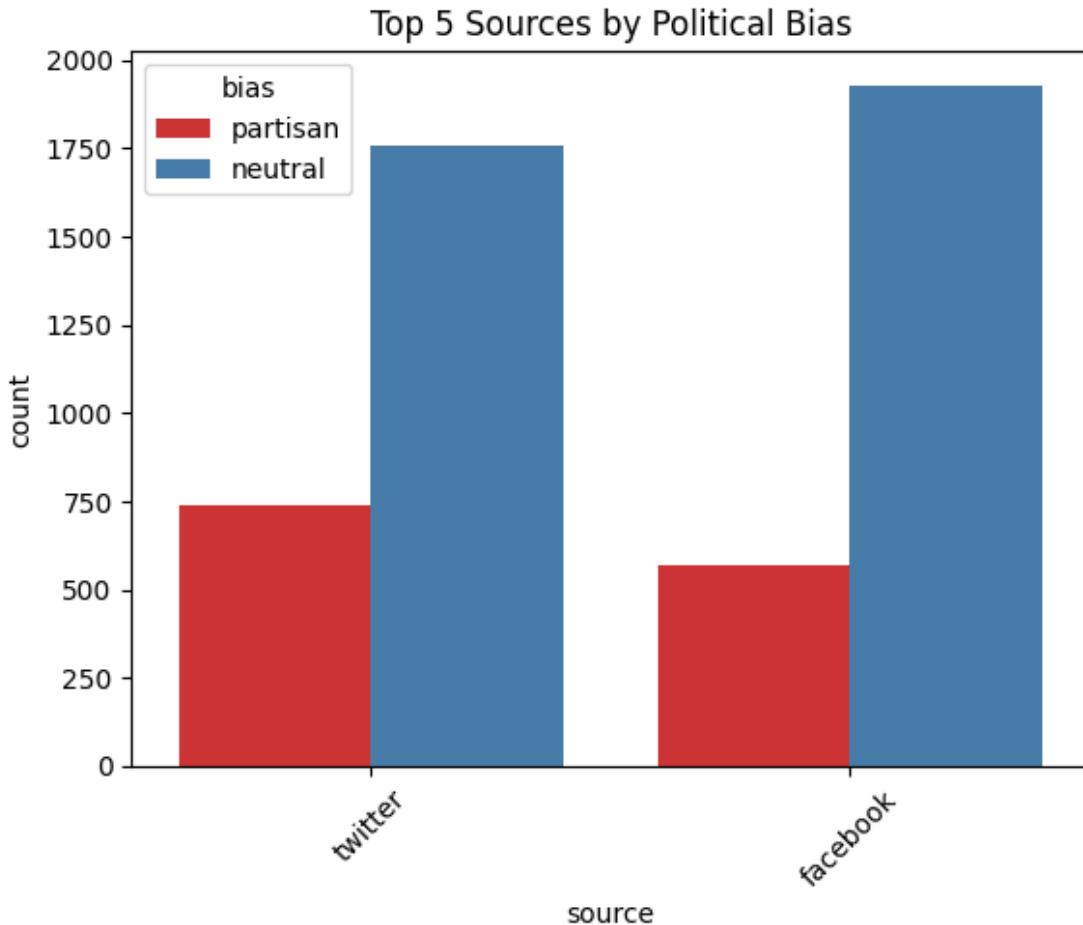
```



Top N sources for clarity

```
top_n_sources = df_clean['source'].value_counts().head(5).index
df_top_sources = df_clean[df_clean['source'].isin(top_n_sources)]

sns.countplot(data=df_top_sources, x='source', hue='bias', palette='Set1')
plt.title("Top 5 Sources by Political Bias")
plt.xticks(rotation=45)
plt.show()
```



Hypothesis Testing

by doing t-Test, chi-square test that helps us to make data-driven decisions by Statistically evaluating patterns assumptions and differences in data.

```
# Chi-square test between bias and message
contingency = pd.crosstab(df_clean['bias'], df_clean['message'])
chi2, p, dof, expected = chi2_contingency(contingency)

print(f"Chi-square value: {chi2}")
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p}")

Chi-square value: 1172.4348079207407
Degrees of Freedom: 8
P-value: 8.654969720482628e-248

from sklearn.preprocessing import LabelEncoder

# Encode categorical features
le_message = LabelEncoder()
```

```

le_source = LabelEncoder()

df_clean['message_encoded'] = le_message.fit_transform(df_clean['message'])
df_clean['source_encoded'] = le_source.fit_transform(df_clean['source'])

# Encode the target variable: bias (neutral = 0, partisan = 1)
df_clean['bias_encoded'] = df_clean['bias'].map({'neutral': 0, 'partisan': 1})

from sklearn.model_selection import train_test_split

# Define features and target
X = df_clean[['message_encoded', 'source_encoded']]
y = df_clean['bias_encoded']

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.linear_model import LogisticRegression

# Create and train the logistic regression model
log_model = LogisticRegression()
log_model.fit(X_train, y_train)

# Predict on test data
y_pred = log_model.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix

```

 **Evaluation Metrics Mean Squared Error (MSE)** – Measures average squared difference between actual and predicted values

Mean Absolute Error (MAE) – Measures average absolute difference

R² Score – Indicates how well the model explains the variance in revenue (1.0 is perfect)

```

# Print evaluation metrics
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

```

Confusion Matrix:

```

[[767  0]
 [233  0]]

```

```

Classification Report:
      precision    recall   f1-score   support

          0       0.77     1.00     0.87     767
          1       0.00     0.00     0.00     233

   accuracy                           0.77     1000
macro avg       0.38     0.50     0.43     1000
weighted avg    0.59     0.77     0.67     1000

```

```

from sklearn.metrics import confusion_matrix
# Predict using the model
y_pred = log_model.predict(X_test)

```

Confusion Matrix

A confusion matrix is a table used to evaluate the performance of a classification model. It shows how well the model is predicting actual classes.

For a multi-class classification the Matrix compares:

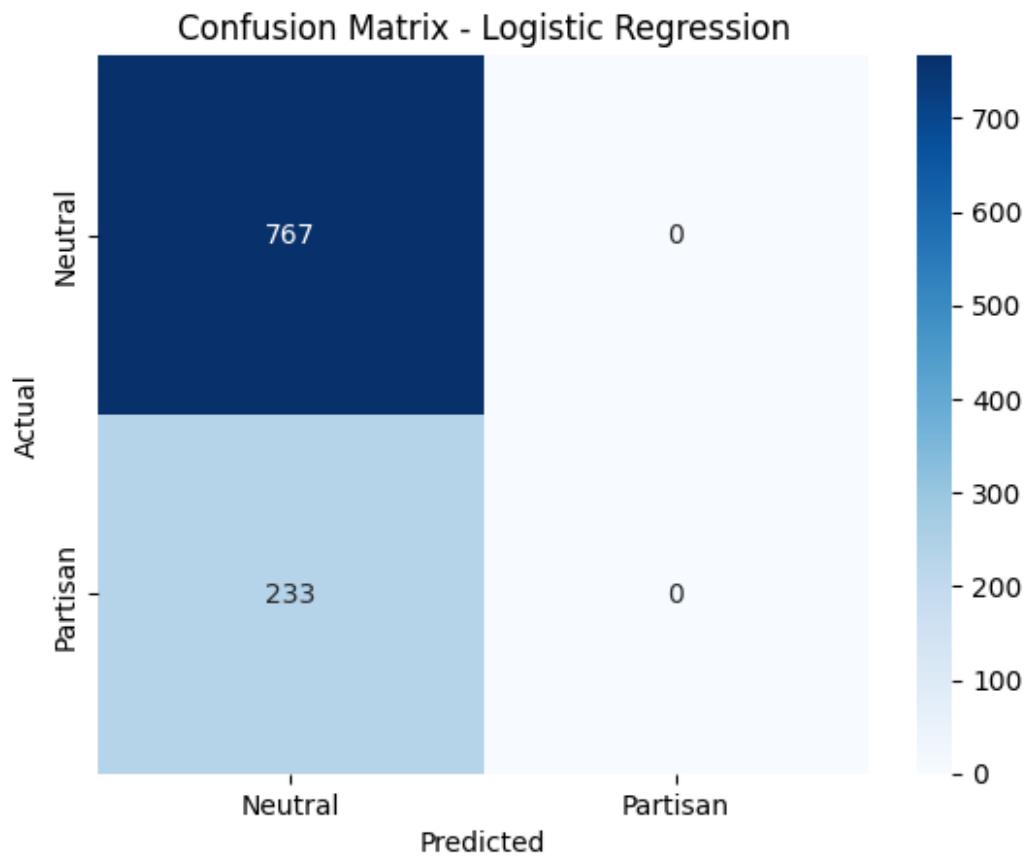
- **Actual classes**(True labels) vs.
- **Predicted classes**(From your model)

```

# Get the confusion matrix
cm = confusion_matrix(y_test, y_pred)
labels = ['Neutral', 'Partisan']

# Plot confusion matrix
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix - Logistic Regression')
plt.show()

```



Final Conclusion

This project successfully demonstrated the application of data science techniques in political media analysis. Exploratory and statistical methods revealed strong relationships between message content and political bias. Although the initial logistic regression model achieved moderate accuracy, it struggled with class imbalance, indicating room for model improvement. Overall, this project emphasized the importance of preprocessing, visualization, and model evaluation in drawing meaningful insights from text-based social media data.

Project Goals

- To analyze the political messaging and bias present in social media posts by public figures.
- To understand the relationship between message content, source platform, and the political bias (neutral or partisan).
- To build a predictive model (logistic regression and random forest) that can classify a message's political bias based on content and source.

- To validate the statistical significance of relationships using hypothesis testing (e.g., Chi-square test).

Workflow Followed

We applied the key steps of a typical data science project:

-  **Collected and cleaned the data**
-  **Explored and visualized patterns**
-  **Applied statistical techniques**
-  **Built and evaluated machine learning models**

Key Learnings

- Learned to preprocess a real-world, multi-feature dataset by cleaning, transforming, and encoding categorical data.
- Gained experience in exploratory data analysis (EDA) using visualization techniques to detect patterns and distributions.
- Applied univariate and multivariate analysis to explore associations between variables such as message type and bias.
- Conducted hypothesis testing (Chi-square test) and interpreted p-values to determine statistical significance.
- Built and evaluated classification models (logistic regression and random forests) to predict political bias.
- Understood model limitations such as class imbalance and the need for tuning and better feature engineering.

References:

- **Imran Ali Shah.** (2023). *Sales and Customer Insights Dataset*. Kaggle. Available at: <https://www.kaggle.com/datasets/thedevastator/uncovering-political-bias-in-social-media-commun?resource=download>
- Pandas Documentation – Data preprocessing, handling missing values, and exploratory data analysis. <https://pandas.pydata.org/docs/>
- **Virtanen, P., Gommers, R., Oliphant, T. E., et al.** (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, 17, 261–272. (Used for hypothesis testing, probability distributions)
- **McKinney, W.** (2010). *Data Structures for Statistical Computing in Python*. In Proceedings of the 9th Python in Science Conference, 51–56. (Used for structured data handling with Pandas)
- **Waskom, M. L.** (2021). *Seaborn: Statistical Data Visualization*. Journal of Open-Source Software, 6(60), 3021. (Used for histogram and KDE plots in probability analysis)
- **Pedregosa, F., Varoquaux, G., Gramfort, A., et al.** (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, 2825–2830. (Used for linear regression and classification algorithms)