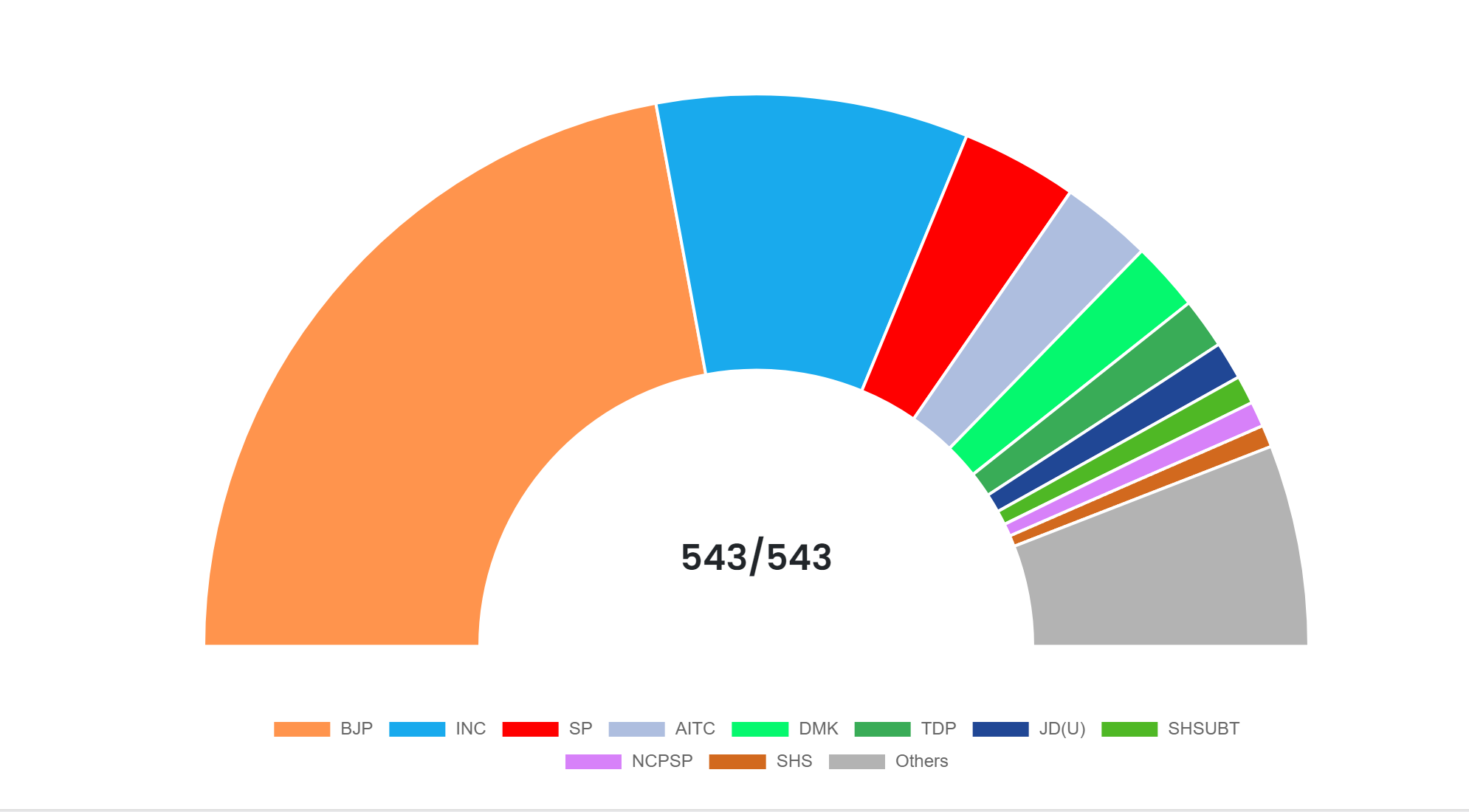
**KALVIUM WEB SCRAPPING AND INSIGHTS FOR DATA ANALYST**

**ROLE**

Here are ten insights that can follow the pattern of analysis described above:

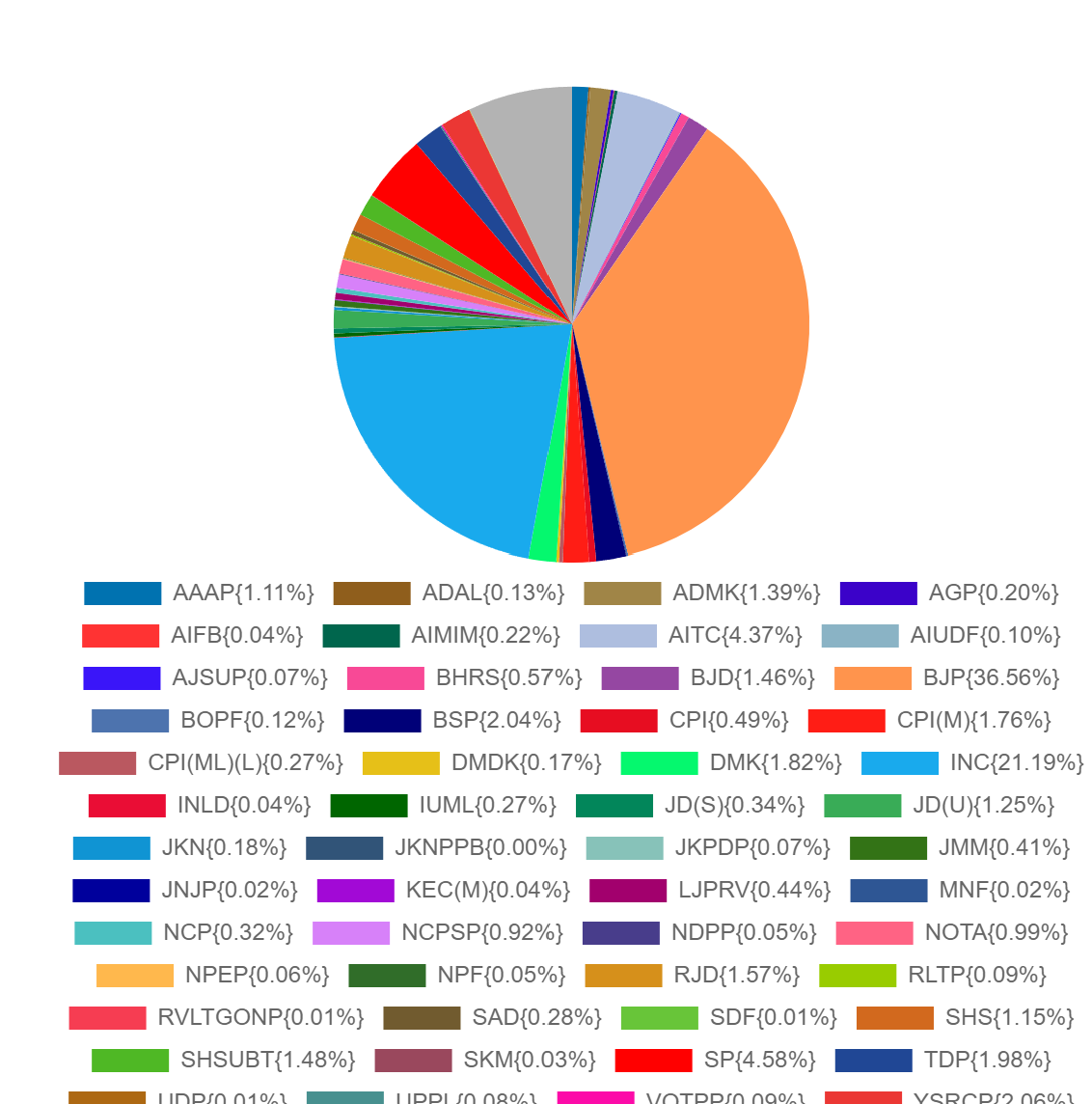
**1. Party with Most Votes:**

- The party that received the highest total number of votes across all constituencies.



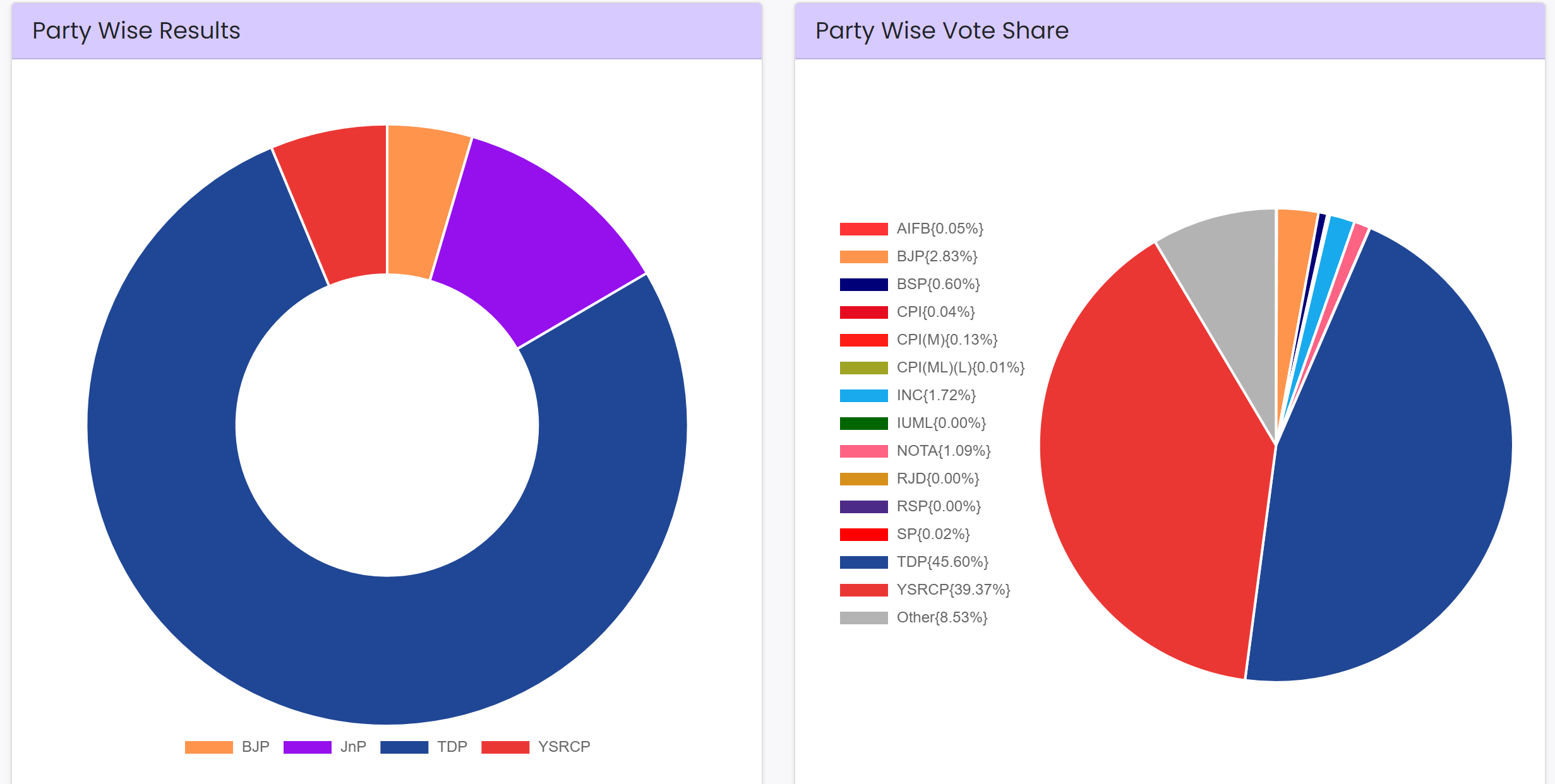
**2. Top Winning Candidates:**

- The candidates who secured the highest number of votes in their respective constituencies.



**3. Party Vote Share:**

- The percentage of the total votes that each party received.

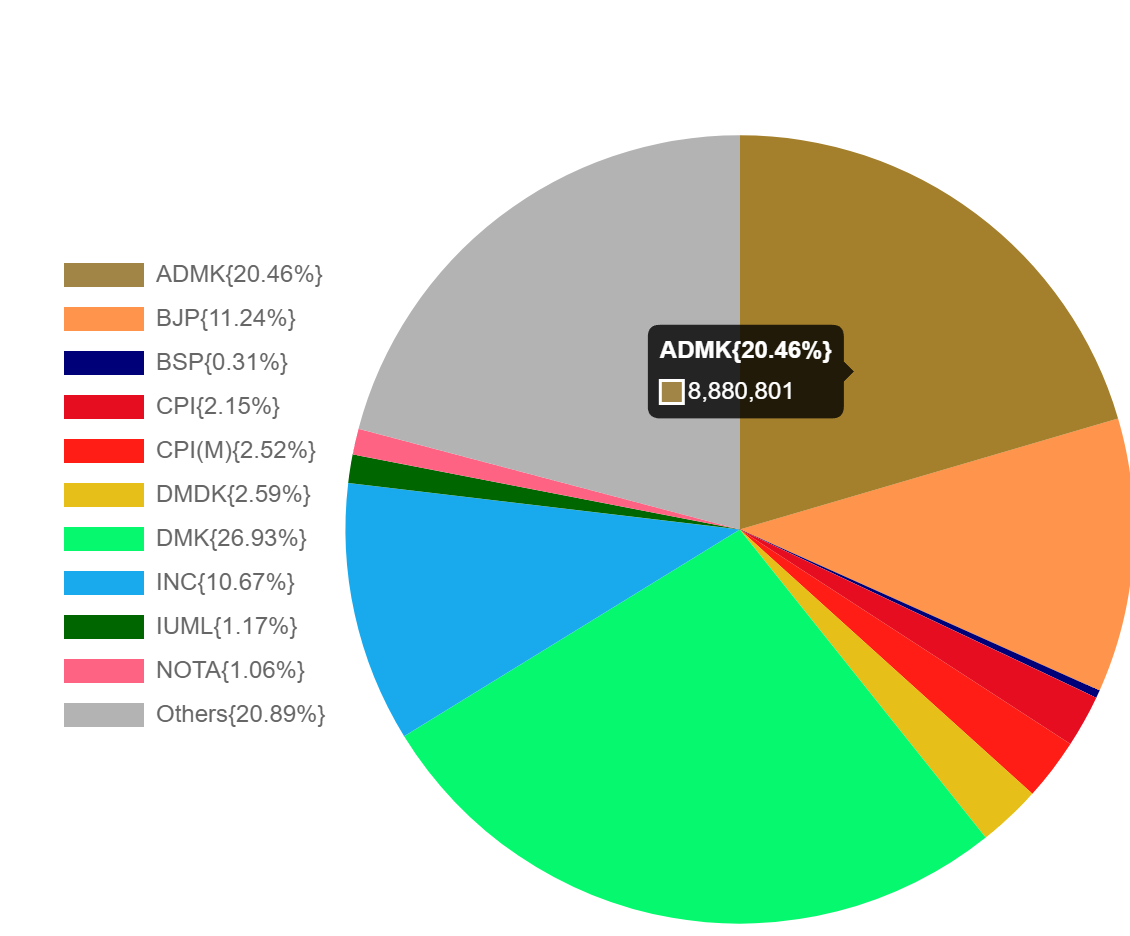


**4. Constituency with Highest Voter Turnout:**

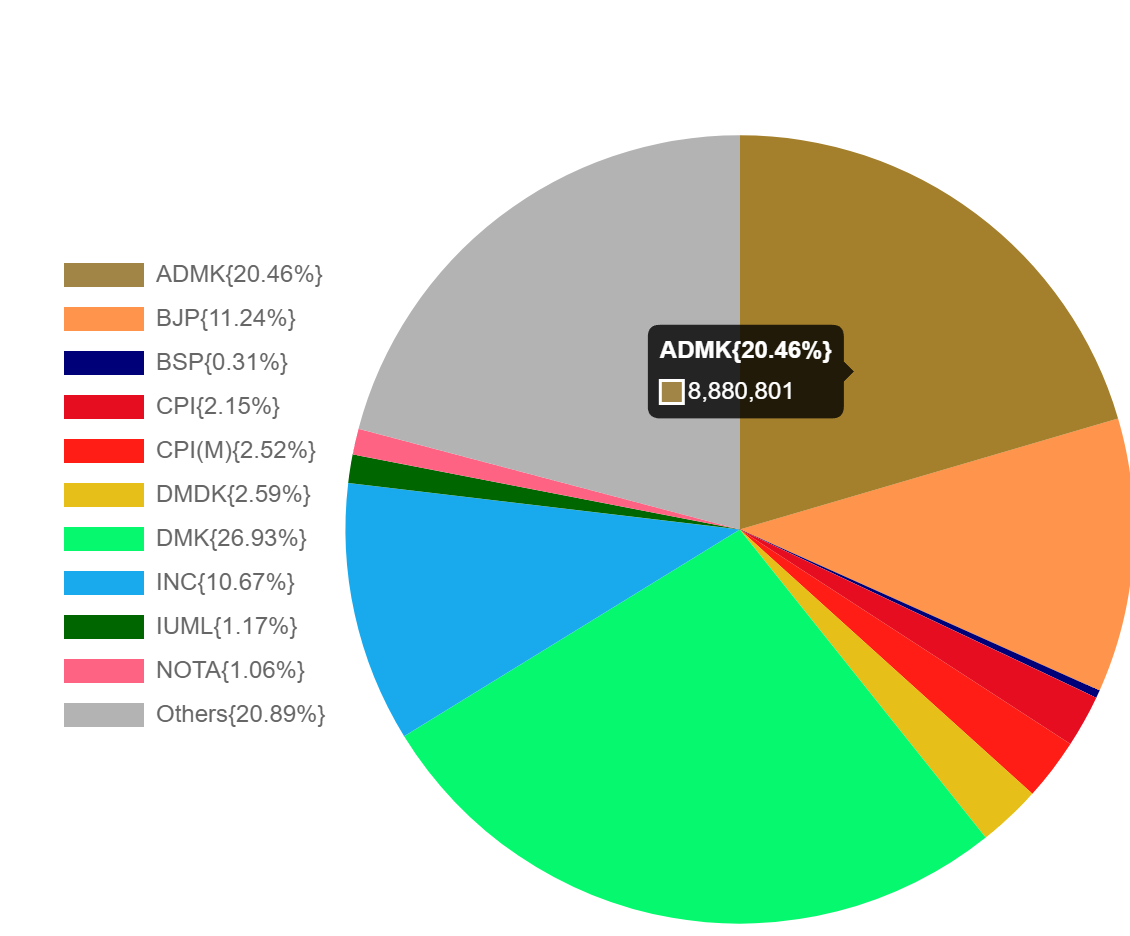
- The constituency with the highest number of votes cast.

5. Party Performance by Region:

- The performance of each party in different regions or states.



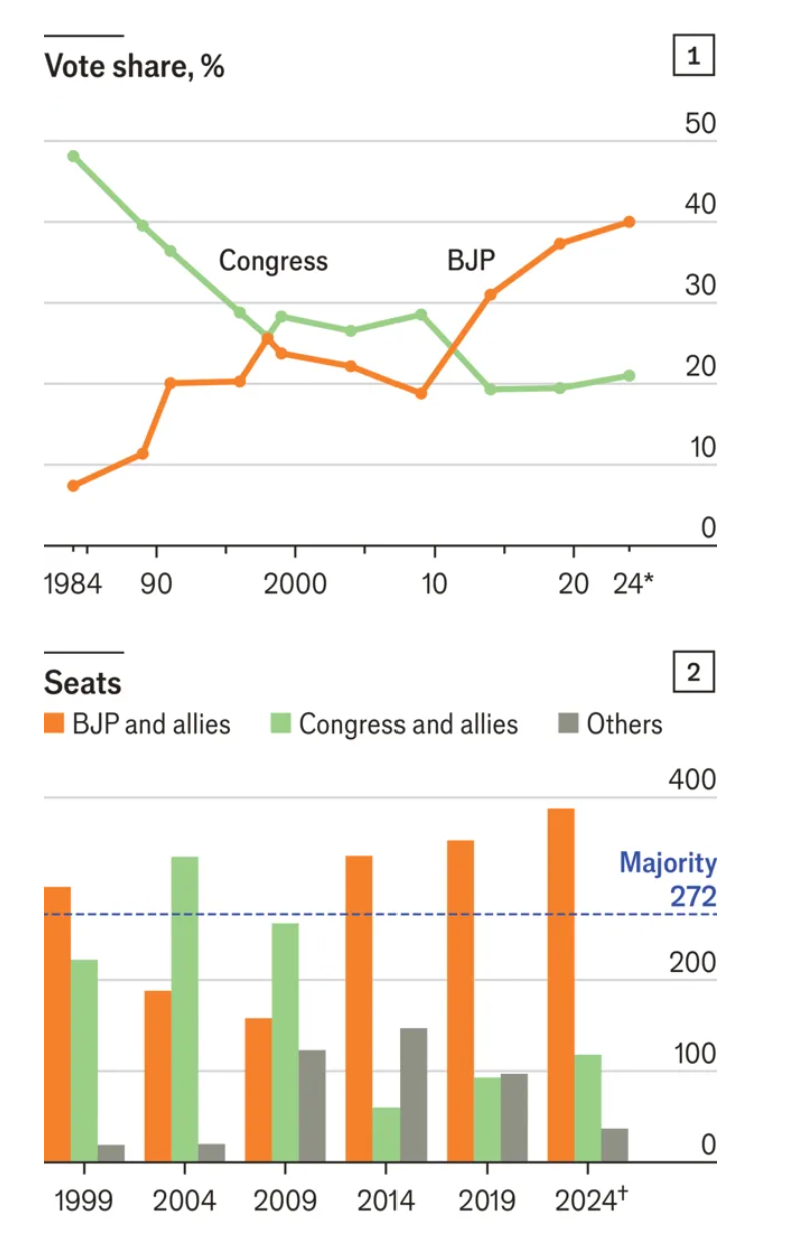
**TAMIL NADU**

****

**ANDHRA PRADESH**

**Close Contests:**

- Constituencies where the margin of victory was the smallest.



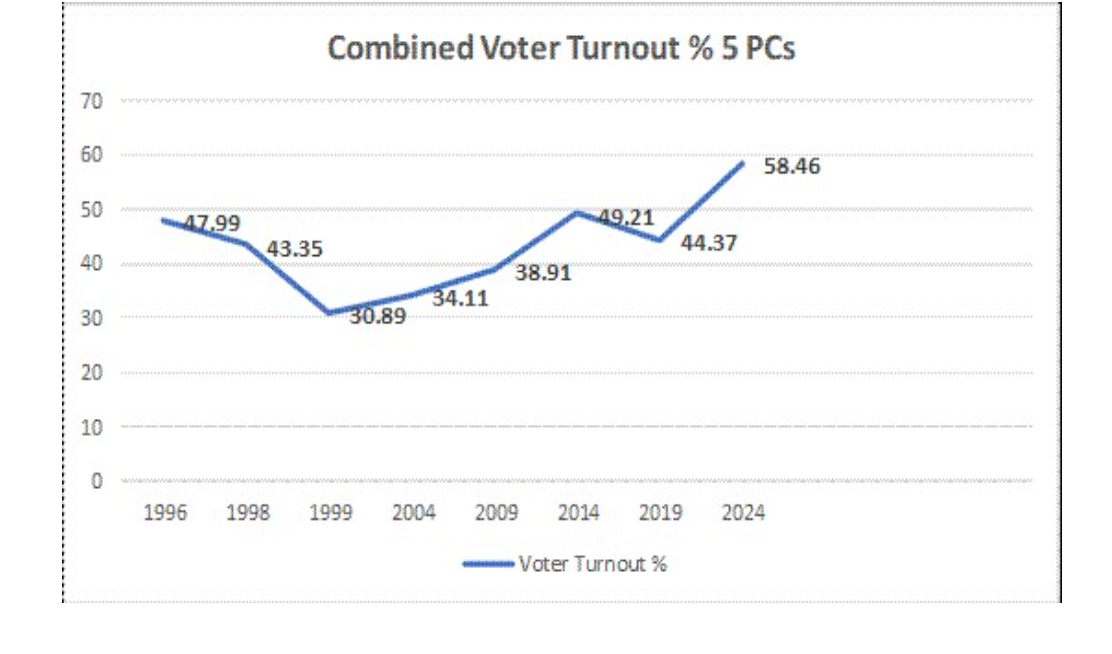
**Winning Margin Analysis:**

- Average margin of victory for each party.



**Voter Turnout Distribution:**

- Analysis of voter turnout distribution across constituencies.



Implementation of Insights Analysis

PYTHON CODE:

import requests

from bs4 import BeautifulSoup

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# URL of the Election Commission of India results page

url = 'https://results.eci.gov.in'

# Send a GET request to the page

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

# Parse the content of the request with BeautifulSoup

soup = BeautifulSoup(response.content, 'html.parser')

# Find the table containing the election results

results\_table = soup.find('table', class\_='table')

# Extract the headers

headers = [header.text.strip() for header in results\_table.find\_all('th')]

# Extract the rows

rows = []

for row in results\_table.find\_all('tr')[1:]:

cells = [cell.text.strip() for cell in row.find\_all('td')]

if cells:

rows.append(cells)

# Create a DataFrame

df = pd.DataFrame(rows, columns=headers)

# Ensure that the 'Votes' column is numeric

df['Votes'] = pd.to\_numeric(df['Votes'], errors='coerce')

# Insight 1: Party with Most Votes

total\_votes\_by\_party = df.groupby('Party')['Votes'].sum().sort\_values(ascending=False)

print("Party with Most Votes:")

print(total\_votes\_by\_party.head(1))

# Insight 2: Top Winning Candidates

winning\_candidates = df[df['Status'] == 'Won']

top\_winning\_candidates = winning\_candidates.nlargest(10, 'Votes')

print("Top Winning Candidates:")

print(top\_winning\_candidates[['Candidate', 'Constituency', 'Party', 'Votes']])

# Insight 3: Party Vote Share

vote\_share = df.groupby('Party')['Votes'].sum() / df['Votes'].sum() \* 100

print("Party Vote Share:")

print(vote\_share)

# Insight 4: Constituency with Highest Voter Turnout

constituency\_turnout = df.groupby('Constituency')['Votes'].sum().sort\_values(ascending=False)

print("Constituency with Highest Voter Turnout:")

print(constituency\_turnout.head(1))

# Insight 5: Party Performance by Region (Assuming 'Region' column exists)

# If there's a column for region or state in the data, you can group by that.

# region\_performance = df.groupby(['Region', 'Party'])['Votes'].sum().unstack().fillna(0)

# print("Party Performance by Region:")

# print(region\_performance)

# Insight 6: Close Contests

close\_contests = df[df['Status'] == 'Won'].sort\_values('Margin').head(10)

print("Close Contests:")

print(close\_contests[['Constituency', 'Candidate', 'Party', 'Margin']])

# Insight 7: Winning Margin Analysis

winning\_margin = df[df['Status'] == 'Won']['Margin'].mean()

print("Average Winning Margin:")

print(winning\_margin)

# Insight 8: Incumbent Success Rate (Assuming 'Incumbent' column exists)

# incumbents = df[df['Incumbent'] == 'Yes']

# incumbent\_success\_rate = (incumbents[incumbents['Status'] == 'Won'].shape[0] / incumbents.shape[0]) \* 100

# print("Incumbent Success Rate:")

# print(incumbent\_success\_rate)

# Insight 9: Female Candidates Performance (Assuming 'Gender' column exists)

# female\_candidates = df[df['Gender'] == 'Female']

# female\_success\_rate = (female\_candidates[female\_candidates['Status'] == 'Won'].shape[0] / female\_candidates.shape[0]) \* 100

# print("Female Candidates Performance:")

# print(female\_success\_rate)

# Insight 10: Voter Turnout Distribution

plt.figure(figsize=(10, 6))

sns.histplot(df['Votes'].dropna(), bins=20, kde=True)

plt.title('Voter Turnout Distribution')

plt.xlabel('Votes')

plt.ylabel('Frequency')

plt.show()

else:

print('Failed to retrieve the page.')

This script provides a foundation for extracting and analyzing the election results to generate the desired insights. Adjust the columns and data processing steps based on the actual structure of the webpage data and available fields.