Election Analysis Report: Anambra State Voting Data

Introduction

This report provides an analysis of the voting data from Anambra State, sourced unofficially by individual election observers. The state exhibits a significant demographic support for the Labour Party (LP). This analysis aims to identify voting patterns, potential anomalies, and outliers using geospatial and statistical methods.

Methodology

1. Data Preparation
    - The dataset was loaded and cleaned to ensure accuracy and consistency.
    - Geocoding was performed using the open source QGIS project python plugin to obtain latitude and longitude for each polling unit, enabling geospatial analysis.
    - Voting numbers for the main parties (APC, LP, PDP, NNPP) were used for statistical analysis.
2. Outlier Score Calculation
    - For each polling unit, the voting numbers were compared to those of neighboring units within a 1 km radius.
    - The Z-score method was used to calculate the outlier score for each voting column, representing how many standard deviations a data point is from the mean of its neighbors.
    - A positive score indicates votes above the mean, while a negative score indicates votes below the mean.

Formula for Outlier Score: $Z = \frac{(X - \mu)}{\sigma}$

Where:

X is the voting number for a party in a polling unit.

μ is the mean voting number for that party in neighboring polling units.

σ is the standard deviation of the voting numbers in neighboring units.

Significance of an Outlier Score of 2.04

An outlier score of 2.04 means the polling unit's voting numbers are 2.04 standard deviations above the mean of its neighboring polling units. In a normal distribution, this corresponds to being higher than approximately 96% of the data, indicating a significant deviation.

Key Findings and Insights

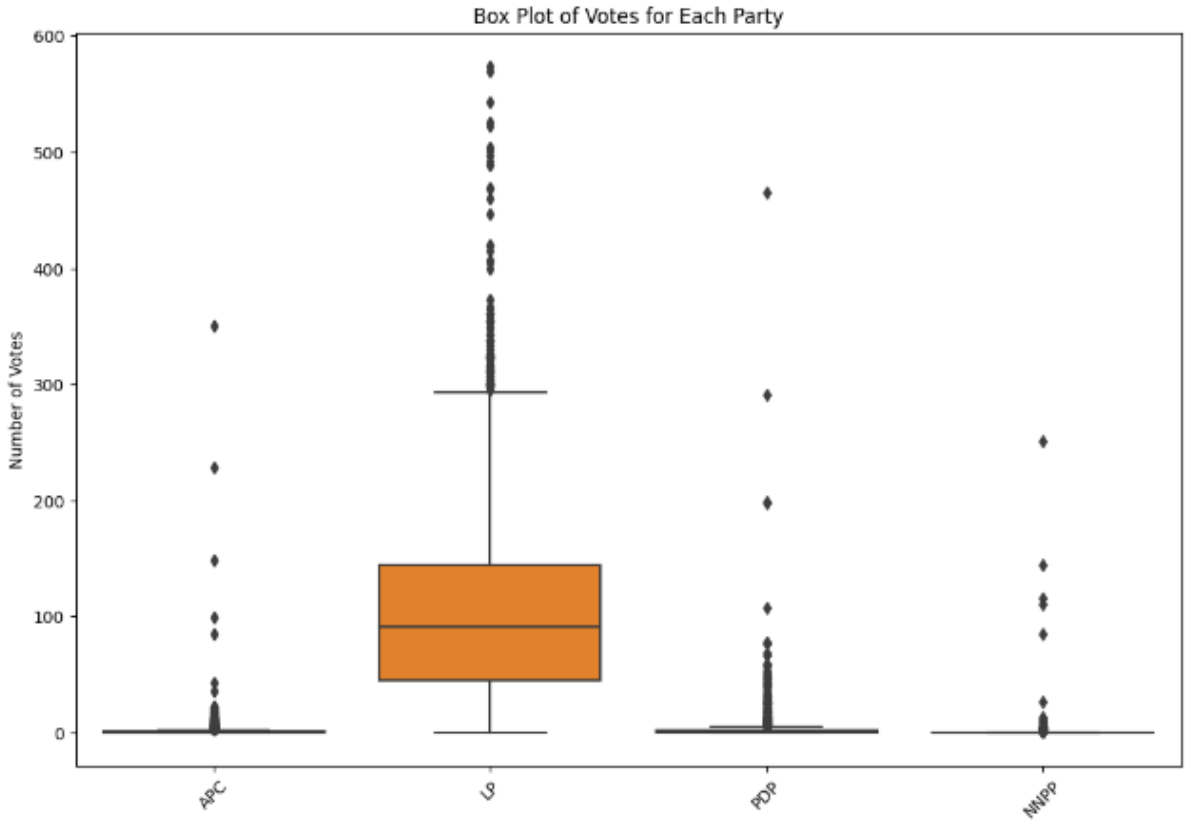
1. Voting Patterns
    a. LP Dominance: The Labour Party (LP) has a wider distribution of votes and a larger number of outliers compared to other parties. This is likely due to the demographic dominance supporting LP in Anambra State.
    b. APC and PDP Outliers: Outlier scores for APC and PDP are the most skewed, suggesting possible malpractice or irregularities, although LP's dominance makes it difficult to conclude definitively.

2. Statistical Analysis
    a. Vote Clustering: Most votes for APC are clustered around lower values, while LP votes are more widely distributed. This indicates a strong support base for LP compared to APC.
    b. Linear Relationship: There is no strong linear relationship between APC and LP votes, as indicated by the almost flat regression line in the scatter plot.
3. Geospatial Analysis
    a. Computational Challenges: The number of polling units in close proximity makes the geospatial data computation intensive, requiring significant CPU power and time. Optimizing the computational process or using high-performance computing resources could improve efficiency.

Visualizations

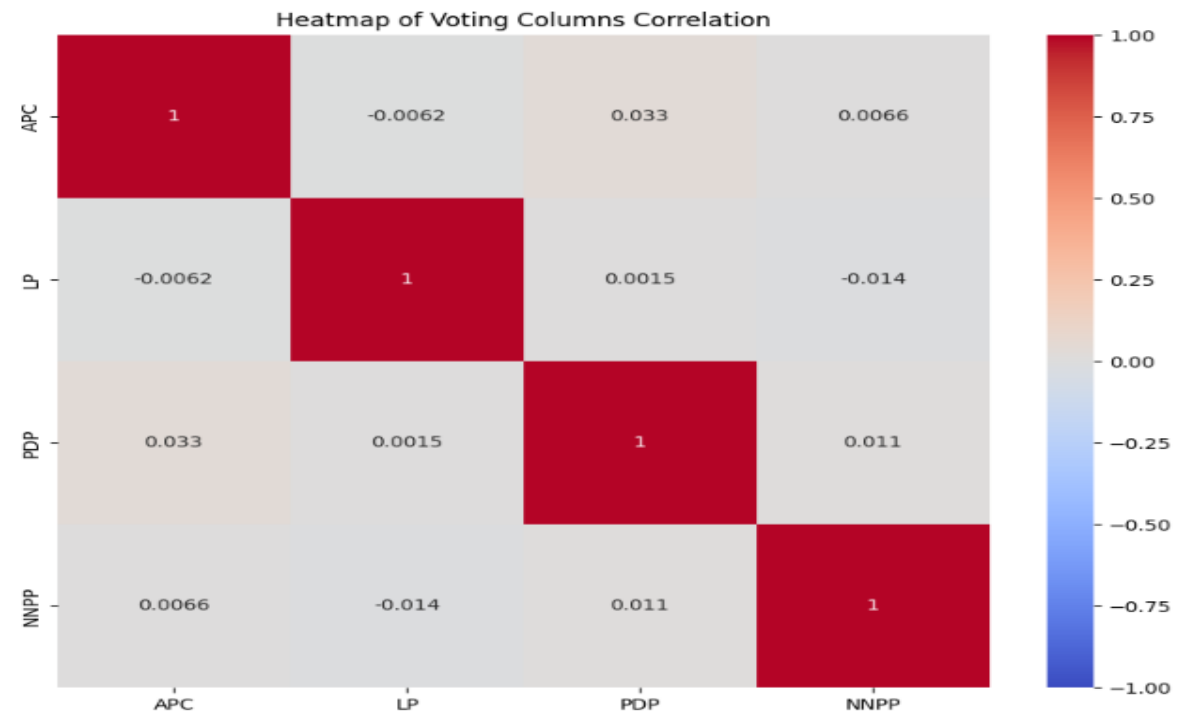
Box Plot of Votes for Each Party

- This plot provides a summary of the distribution of votes for each party, highlighting the median, interquartile range, and outliers.



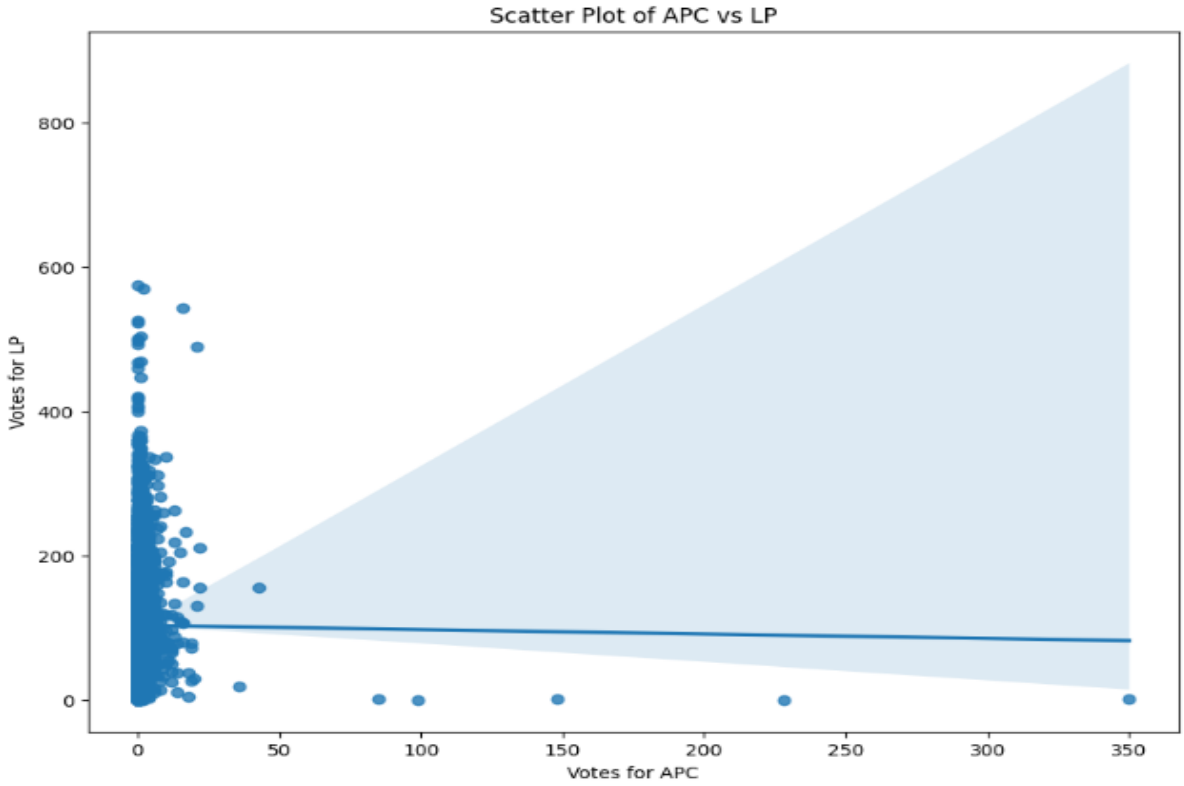
Heatmap of Voting Columns Correlation

- The heatmap shows the correlation between the votes of different parties, indicating very low or near-zero correlation between them.



Scatter Plot of APC vs LP Votes

- The scatter plot visualizes the relationship between APC and LP votes, with a regression line indicating the trend.



Conclusion

The analysis reveals significant insights into the voting patterns in Anambra State, with a clear dominance of the Labour Party. The outlier scores for APC and PDP suggest potential irregularities, though further investigation is needed to draw definitive conclusions. The geospatial analysis underscores the computational challenges due to the high density of polling units.

APPENDIX

Python code for neighbor detection and outlier score calculation;

```
!pip install pandas geopandas shapely geopy scipy
import pandas as pd
import numpy as np
from scipy.spatial import KDTree
from geopy.distance import geodesic
import geopandas as gpd
from shapely.geometry import Point
import folium
import matplotlib.pyplot as plt
import seaborn as sns
# Load your data
file_path = '/kaggle/input/anambra-with-geo/anambra_with_geo.csv'
data = pd.read_csv(file_path)
# Ensure Latitude and Longitude columns are numeric
data['Latitude'] = pd.to_numeric(data['Latitude'], errors='coerce')
data['Longitude'] = pd.to_numeric(data['Longitude'], errors='coerce')

# Drop rows with missing coordinates
data = data.dropna(subset=['Latitude', 'Longitude'])

# Create a GeoDataFrame
gdf = gpd.GeoDataFrame(data,
geometry=gpd.points_from_xy(data.Longitude, data.Latitude))
```

```
# Convert latitude and longitude to radians
gdf['Latitude_rad'] = np.radians(gdf['Latitude'])
gdf['Longitude_rad'] = np.radians(gdf['Longitude'])

# Create KDTree for fast spatial indexing
tree = KDTree(gdf[['Latitude_rad', 'Longitude_rad']].values)

# Function to find neighbors within a radius
def find_neighbors_within_radius(tree, gdf, radius_km):
    neighbors_dict = {}
    radius_rad = radius_km / 6371.0  # Convert radius to radians
    for idx, row in gdf.iterrows():
        center_point = np.array([row['Latitude_rad'],
row['Longitude_rad']])
        indices = tree.query_ball_point(center_point, radius_rad)
        neighbors = gdf.iloc[indices]['PU-Name'].tolist()
        neighbors.remove(row['PU-Name'])  # Remove self from neighbors
        neighbors_dict[row['PU-Name']] = neighbors
    return neighbors_dict

# Find neighbors within 1km radius
neighbors_within_1km = find_neighbors_within_radius(tree, gdf, 1)

# Inspect the dictionary to understand its structure
print(f"Sample neighbors_within_1km:
{list(neighbors_within_1km.items())[:5]}")

# Convert neighbors dictionary to DataFrame
neighbors_df = pd.DataFrame(list(neighbors_within_1km.items()),
columns=['PU-Name', 'Neighbors'])
```

```
# Merge neighbors with original DataFrame
data_with_neighbors = pd.merge(data, neighbors_df, on='PU-Name', how='left')
# Function to find outliers and calculate outlier scores
def calculate_outlier_scores(row, data, columns, threshold=2):
    if row['Neighbors']:
        neighbors = data[data['PU-Name'].isin(row['Neighbors'])]
        scores = {}
        for col in columns:
            mean = neighbors[col].mean()
            std = neighbors[col].std()
            if std > 0:
                score = abs(row[col] - mean) / std
                if score > threshold:
                    scores[col] = round(score, 2)  # Round to 2 significant figures
        return scores
    return None

# Apply the function to calculate outlier scores in voting numbers
voting_columns = ['APC', 'LP', 'PDP', 'NNPP']
data_with_neighbors['Outlier_Scores'] = data_with_neighbors.apply(
    lambda row: calculate_outlier_scores(row, data_with_neighbors, voting_columns), axis=1)

# Filter rows with outlier scores that are dictionaries
outliers_df = data_with_neighbors[data_with_neighbors['Outlier_Scores'].apply(lambda x: isinstance(x, dict))]

# Explode the outlier scores dictionary into separate rows
outliers_exploded = outliers_df.explode('Outlier_Scores')

# Create a DataFrame from the exploded outlier scores
```

```
outlier_details = pd.DataFrame([
    (pu_name, col, score, neighbors)
    for pu_name, scores, neighbors in zip(outliers_exploded['PU-Name'], outliers_exploded['Outlier_Scores'], outliers_exploded['Neighbors'])
    for col, score in (scores.items() if isinstance(scores, dict) else [])
], columns=['PU-Name', 'Voting_Column', 'Outlier_Score', 'Neighbors'])

# Sort the outlier details by Outlier_Score in descending order
sorted_outliers = outlier_details.sort_values(by='Outlier_Score', ascending=False)

# Merge the outlier scores back into the original DataFrame
data_with_outliers = pd.merge(data_with_neighbors, sorted_outliers, on='PU-Name', how='left')
# Save the updated DataFrame to a new CSV file
updated_file_path = '/kaggle/working/anambra_with_outliers.csv'
data_with_outliers.to_csv(updated_file_path, index=False)

# Print the updated DataFrame with outlier scores
print(data_with_outliers.head())
```