

Advanced Geospatial Analysis for Election Integrity

Introduction

1.1 Background & Importance of Election Integrity

Election integrity is the backbone of any democracy because it ensures that the people's voices are truly heard. In Nigeria, there have always been rumors of election fraud—ballot stuffing, voter suppression, and results being changed—but there was never enough solid proof to back these claims. People would complain, but without data, it was just talk. Now, for the 2023 election, we finally have access to real numbers, maps, and statistics that can help us see if these suspicions are true or not. With proper analysis, we can spot unusual voting patterns and understand whether some polling units had results that just don't add up. This is our chance to use data to separate facts from assumptions and push for fairer elections in the future.

1.2 Objectives of the Study

The goal of this project is to find polling units where the voting numbers look suspicious. To do this, I used different data analysis methods to check if some places had strange voting patterns that don't match what we would normally expect. This report will show how I used data to look for possible election issues. Below are the key steps I followed:

- **Getting the Right Dataset:** I downloaded the dataset for Abia State, which contains details about polling units, votes, and locations. Some polling units didn't have latitude and longitude, so I used geocoding tools like Google Maps to get the missing locations. This was important because we needed accurate coordinates to analyze voting patterns correctly.

- **Finding Neighboring Polling Units:** To understand if a polling unit had unusual results, I needed to compare it to nearby polling units. I used geospatial clustering techniques like DBSCAN and HDBSCAN to group polling units based on how close they are to each other. I tested different distances (500m, 1km, and 2km) to see how changing the radius affects the detection of unusual results.
- **Detecting Outliers in Voting Patterns**
- To find polling units with suspicious vote counts, I calculated special outlier scores using different methods:
 - Local Moran's I helped identify areas where voting patterns were very different from nearby polling units.
 - Getis-Ord Gi (Hot Spot Analysis) showed places with extreme vote concentrations, either very high or very low.
 - Isolation Forest, a machine learning method, helped confirm which polling units were true outliers based on vote counts.
- **Comparing with Past Elections and Population Data:** I didn't just look at 2023 votes—I also compared them with past elections to see if any polling unit suddenly had a huge change in votes. I also used population data to check if some areas had way more votes than people living there, which could mean possible fraud or mistakes in the results.
- **Building an Interactive Dashboard:** Finally, I put everything together in a dashboard using Power BI. The dashboard has different charts and maps that answer the following questions
 - Which polling units are outliers
 - Clusters of unusual votes
 - Voting trends over time
 - How votes compare to population sizes

This report will walk through each of these steps in detail, showing the methods I used, the results I found, and what they mean for election integrity in Nigeria.


1.3 Scope of the Analysis

I focus on using the election data from Abia State to identify polling units with unusual voting patterns. The analysis is based on geographical clustering and statistical techniques to detect anomalies and potential irregularities in voting results.

To achieve this, I used geospatial clustering methods like DBSCAN and HDBSCAN to group polling units based on their locations. Additionally, I applied statistical techniques such as Local Moran's I and Getis-Ord Gi (Hot Spot Analysis) to highlight areas where voting numbers are significantly different from their neighbors. Machine learning techniques like Isolation Forest were also used to validate outliers.

Dataset Preparation

2.1 Data Sources & Description

- **Abia Election Dataset:** This data set was gotten from the  Nigeria Presidential Election 2023 Electoral Sheets Collation it had a shape of (2492, 19)
- **Historical Election Data:** To observe the historic trend of election in Abia state, i use Historical election data from 2015 and 2019
- **Socioeconomic Data:** Population data (1991, 2006, 2022 projections).

2.2 Data Cleaning & Preprocessing Steps

Before analyzing the election data, I first ensured that the dataset was **clean and accurate**. This was important because missing or incorrect data could affect the results. Here are the steps I took:

Step 1: Remove duplicate or missing records

```
import numpy as np # for numerical operations
import folium # for map plotting
from folium.plugins import MarkerCluster
import geopy # for geocoding
import matplotlib.pyplot as plt # for plotting
import seaborn as sns # for plotting
import time # for time operations to avoid Api limit issues for the geocoding
import requests
import pandas as pd
from tqdm import tqdm
import osmnx as ox
import geopandas as gpd
```

```
df = pd.read_csv("ABIA_crosschecked.csv")
```

```
df.head(10) # Display first 10 rows
```

```
df.describe()
```

	Accredited_Voters	Registered_Voters	Transcription_Count	APC	LP	PDP	NNPP
count	2492.000000	2492.000000	2492.0	2492.000000	2492.000000	2492.000000	2492.000000
mean	96.849920	475.422552	-1.0	2.926565	78.174960	6.633226	0.392055
std	76.049937	434.374296	0.0	10.733881	69.568717	12.421262	2.250471
min	0.000000	0.000000	-1.0	0.000000	0.000000	0.000000	0.000000
25%	39.000000	111.000000	-1.0	0.000000	26.000000	0.000000	0.000000
50%	84.000000	413.000000	-1.0	1.000000	63.000000	2.000000	0.000000
75%	138.000000	743.000000	-1.0	3.000000	111.000000	8.000000	0.000000
max	571.000000	4747.000000	-1.0	350.000000	506.000000	185.000000	84.000000

```
df.isnull().sum()
```

```
State          0
LGA            0
Ward           0
PU-Code        0
PU-Name        0
Accredited_Voters  0
Registered_Voters  0
Results_Found   0
Transcription_Count  0
Result_Sheet_Stamped  0
Result_Sheet_Corrected  0
Result_Sheet_Invalid  0
Result_Sheet_Unclear  0
Result_Sheet_Unsigned  0
APC            0
LP            0
PDP           0
NNPP          0
Results_File   0
dtype: int64
```

```
df.duplicated().sum()
```

```
0
```

- Some rows were repeated or had missing values, which could cause errors in the analysis. I removed duplicates and handled missing values properly.

Step 2: Standardize column names

- The dataset had inconsistent column names. I renamed them to follow a clear format so they would be easier to work with.

Step 3: Check for discrepancies in vote totals vs. registered voters

- I verified that the total votes recorded did not exceed the number of registered voters at each polling unit. If there were major differences, I flagged those records for further review.

```
for col in categorical_cols:
    print(f"Column: {col}")
    print(df[col].unique()) # Show first 10 unique values
    print("-" * 50)
```

```
# Count occurrences of each Results_File
file_counts = df['Results_File'].value_counts()

# Identify duplicate files (appearing more than once)
duplicate_files = file_counts[file_counts > 1].index

# Count the number of flagged duplicates
num_duplicates = df['Results_File'].isin(duplicate_files).sum()

# Display the count of duplicate entries
print(f"Total duplicate entries: {num_duplicates}")
```

2.3 Geocoding Missing Polling Unit Coordinates

This code retrieves latitude and longitude coordinates for polling units in Abia State using Google Maps. It converts polling unit addresses into geographic coordinates to ensure accurate location mapping. The updated dataset helps election officials verify polling unit placements, improve logistics, and support fair election monitoring. The geocoded data is saved for further analysis and mapping.

```
# Your Google Maps API Key
API_KEY = "AIzaSyAn8XXMEoJrqp5Hk5UdnfgvFM4-L5_0NQ8"

# Function to get latitude and longitude
def get_lat_lon(address):
    base_url = "https://maps.googleapis.com/maps/api/geocode/json"
    params = {
        "address": address,
        "key": API_KEY
    }
    response = requests.get(base_url, params=params)
    result = response.json()

    if result["status"] == "OK":
        location = result["results"][0]["geometry"]["location"]
        return location["lat"], location["lng"]
    else:
        return None, None

# Create new columns for latitude and longitude
df["Latitude"] = None
df["Longitude"] = None

# Iterate over polling units and get coordinates
for index, row in tqdm(df.iterrows(), total=len(df)):
    address = f"{row['PU-Name']}, {row['Ward']}, {row['LGA']}, {row['State']}, Nigeria"
    lat, lon = get_lat_lon(address)
    df.at[index, "Latitude"] = lat
    df.at[index, "Longitude"] = lon

# Save the updated dataset
df.to_csv("geocoded_dataset.csv", index=False)
print("Geocoding complete. Saved as geocoded_dataset.csv")
```

Filtering out Coordinates that are not in Abia

This code helps verify and visualize polling units in Abia State for better election planning. It checks if polling units are within Abia's boundary, removes any outside, and saves the cleaned data. A map is created with green dots marking valid polling units, helping election officials ensure accurate locations for fair and efficient elections.

```
# Define the place name for Abia State
place_name = "Abia, Nigeria"

# Download the administrative boundary from OSM
abia_boundary = ox.geocode_to_gdf(place_name)

# Save as GeoJSON file
abia_boundary.to_file("abia_state_boundary.geojson", driver="GeoJSON")

print("✅ Abia State boundary saved as abia_state_boundary.geojson")
```



```

import folium
import pandas as pd
import geopandas as gpd
from shapely.geometry import Point

# Load polling unit data
df = pd.read_csv("geocoded_dataset.csv")

# Load Abia State boundary (GeoJSON file)
abia_boundary = gpd.read_file("abia_state_boundary.geojson")

# Convert polling unit coordinates into geometry points
df["geometry"] = df.apply(lambda row: Point(row["Longitude"], row["Latitude"]), axis=1)

# Convert polling unit dataframe to GeoDataFrame
gdf = gpd.GeoDataFrame(df, geometry="geometry", crs="EPSG:4326")

# Check if polling units are inside Abia's boundary
gdf["inside_abia"] = gdf.geometry.within(abia_boundary.union_all())

# Filter out polling units outside Abia
gdf = gdf[gdf["inside_abia"] == True]

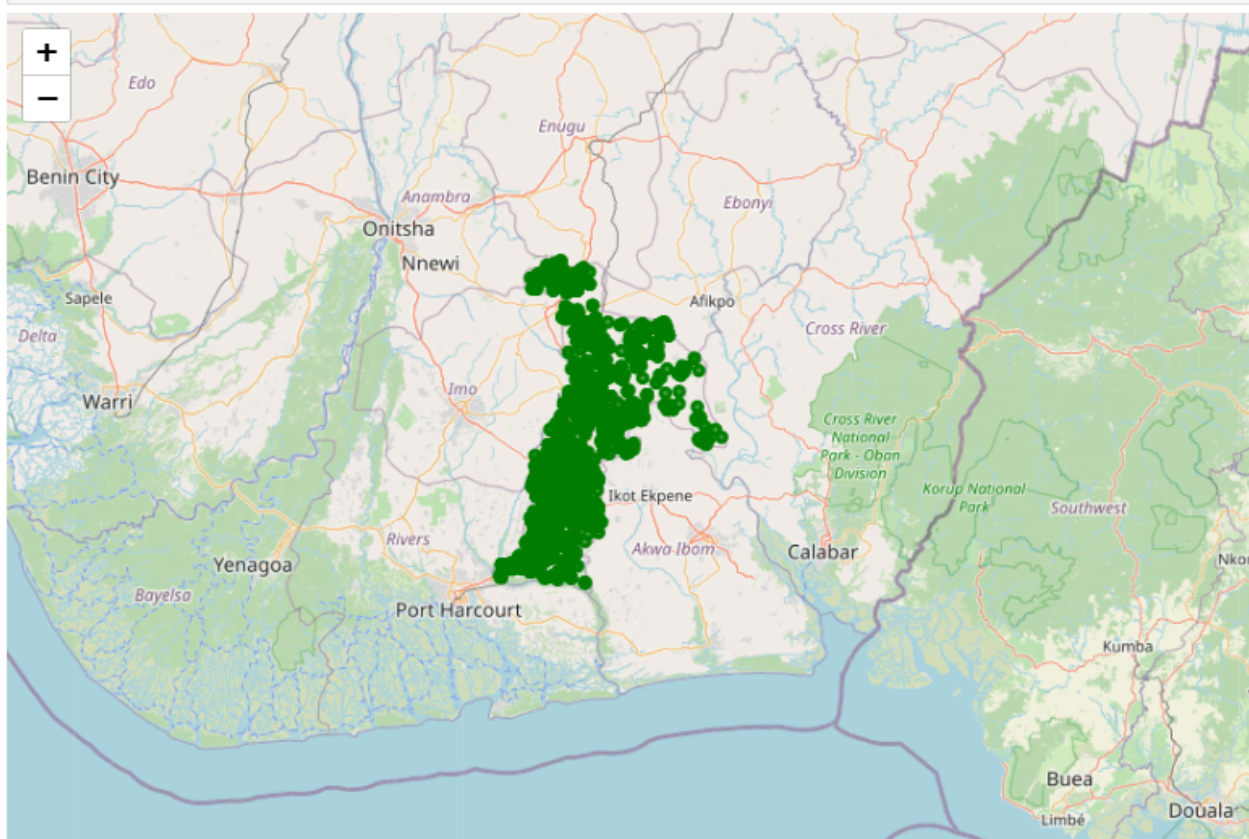
# Save the filtered data to a new CSV file
gdf.drop(columns=["geometry", "inside_abia"]).to_csv("filtered_polling_units.csv", index=False)

# Create the map centered around Abia
abia_center = [5.532, 7.486] # Central coordinates of Abia
m = folium.Map(location=abia_center, zoom_start=9)

# Plot only polling units inside Abia (Green dots)
for _, row in gdf.iterrows():
    folium.CircleMarker(
        location=[row["Latitude"], row["Longitude"]],
        radius=3,
        color="green",
        fill=True,
        fill_color="green",
        fill_opacity=0.7,
        popup=f"Polling Unit: {row['PU-Name']}",
    ).add_to(m)

# Save the map as an HTML file
m.save("abia_polling_units_map.html")

```



Geospatial Clustering & Neighborhood Identification

3.1 Methodology: DBSCAN & HDBSCAN for Polling Unit Clusters

This code groups polling units in Abia State based on their proximity to each other. Using the DBSCAN clustering method with Haversine distance, it identifies clusters of polling units within 500 meters, 1 kilometer, and 2 kilometers. The results help election officials detect closely located polling units, optimize resource allocation, and improve voter distribution. The clustered data is saved for further analysis.

```

import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
from haversine import haversine

# Load geocoded dataset
df = pd.read_csv("filtered_polling_units.csv")

# Convert Latitude and Longitude to tuples
df["coordinates"] = list(zip(df["Latitude"], df["Longitude"]))

# Function to apply DBSCAN clustering with Haversine distance
def apply_dbscan(df, radius_meters, min_samples=2):
    radius_km = radius_meters / 1000 # Convert meters to kilometers
    coords = np.radians(df["coordinates"].tolist()) # Convert to radians
    clustering = DBSCAN(eps=radius_km / 6371, min_samples=min_samples, metric='haversine').fit(coords)
    return clustering.labels_

# Apply DBSCAN clustering for different radius values
df["Cluster_500m"] = apply_dbscan(df, 500)
df["Cluster_1km"] = apply_dbscan(df, 1000)
df["Cluster_2km"] = apply_dbscan(df, 2000)

# Save the clustered dataset
df.to_csv("clustered_abia.csv", index=False)
print("Clustering complete! Results saved in 'clustered_abia.csv'.")

```

Outlier Detection & Statistical Analysis

This code identifies unusual polling units in Abia State based on voting patterns and location. It calculates three key scores: Moran's I (to detect spatial clusters), Getis-Ord G_i^* (to find hotspots and cold spots), and Isolation Forest (to flag statistical anomalies). These scores are combined into a final weighted outlier score, highlighting potential irregularities in polling unit data. The results help election officials detect unusual voting patterns, assess fairness, and ensure accurate election monitoring. The analyzed data is saved for further review.

```

import pandas as pd
import numpy as np
from sklearn.ensemble import IsolationForest
from esda.moran import Moran_Local
from esda.getisord import G_Local # Use G_Local instead of G
from libpysal.weights import DistanceBand
from scipy.stats import zscore

# Load the dataset
df = pd.read_csv("clustered_abia.csv")

# Ensure Latitude and Longitude exist
if "Latitude" not in df.columns or "Longitude" not in df.columns:
    raise ValueError("Latitude and Longitude columns are required!")

# Ensure total_votes column exists
vote_columns = ["APC", "LP", "PDP", "NNPP"]
df["total_votes"] = df[vote_columns].sum(axis=1)

# Create spatial weight matrix (Increase threshold to avoid zero neighbours)
coords = df[["Latitude", "Longitude"]].values
w = DistanceBand(coords, threshold=5, silence_warnings=True) # Increased to 5km

# 1. Local Moran's I Score Calculation (Handle NaN values)
moran_local = Moran_Local(df["total_votes"], w)
df["Moran_I_Score"] = zscore(np.nan_to_num(moran_local.Is)) # Replace NaNs with 0

# 2. Getis-Ord Gi* Score Calculation
df["total_votes"] = df["total_votes"].astype(float) # Ensure it's float
df["total_votes"] = df["total_votes"].fillna(0) # Handle NaNs

getis = G_Local(df["total_votes"], w, n_jobs=1) # Use G_Local without parallel processing
df["Getis_Ord_Gi_Score"] = zscore(np.nan_to_num(getis.Zs)) # Handle NaNs

# 3. Isolation Forest Outlier Score Calculation
iso_forest = IsolationForest(contamination=0.05, random_state=42)
df["Isolation_Forest_Score"] = iso_forest.fit(df[["Latitude", "Longitude", "total_votes"]]).decision_function(
    df[["Latitude", "Longitude", "total_votes"]]
)

# 4. Final Weighted Outlier Score
df["Final_Outlier_Score"] = (
    ((df["Moran_I_Score"] * 0.4) + (df["Getis_Ord_Gi_Score"] * 0.4)) *
    (1 + (df["Isolation_Forest_Score"] * 0.5)) # Boosts flagged outliers
)

```

Visualization & Dashboard Development

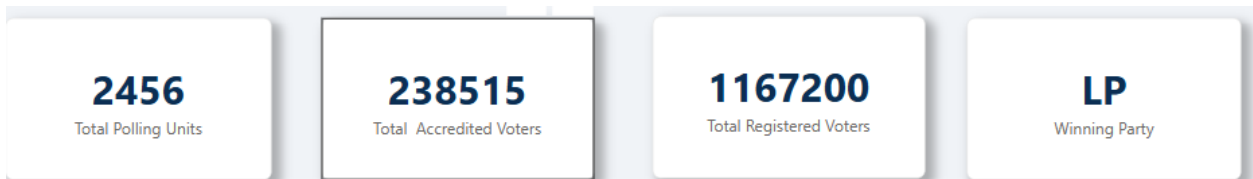
Slicing by **LGA (17 LGAs)** helps Chinwe zoom in on specific areas to spot irregularities without getting lost in county-wide data. **1km clusters (297 clusters)** group polling units within proximity, making it easier to detect isolated cases of vote rigging—if one small cluster shows wildly different voting patterns, it might indicate coercion. On a broader scale, **2km clusters (133 clusters)** help uncover large-scale fraud patterns;

if an entire 2km cluster swings heavily toward one party while surrounding areas remain balanced, it raises red flags for further investigation as well as 500km (398 clusters)

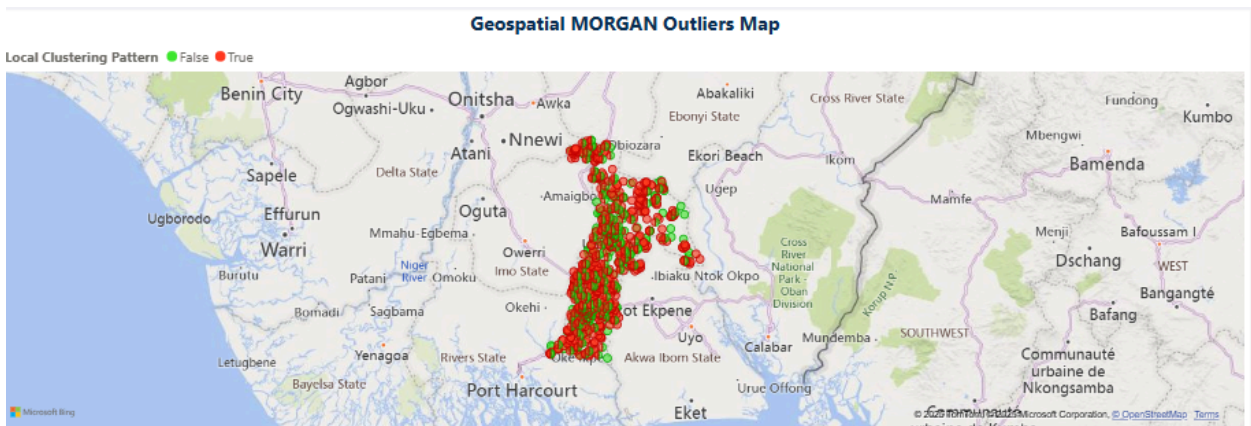


LGA	Cluster
All	All

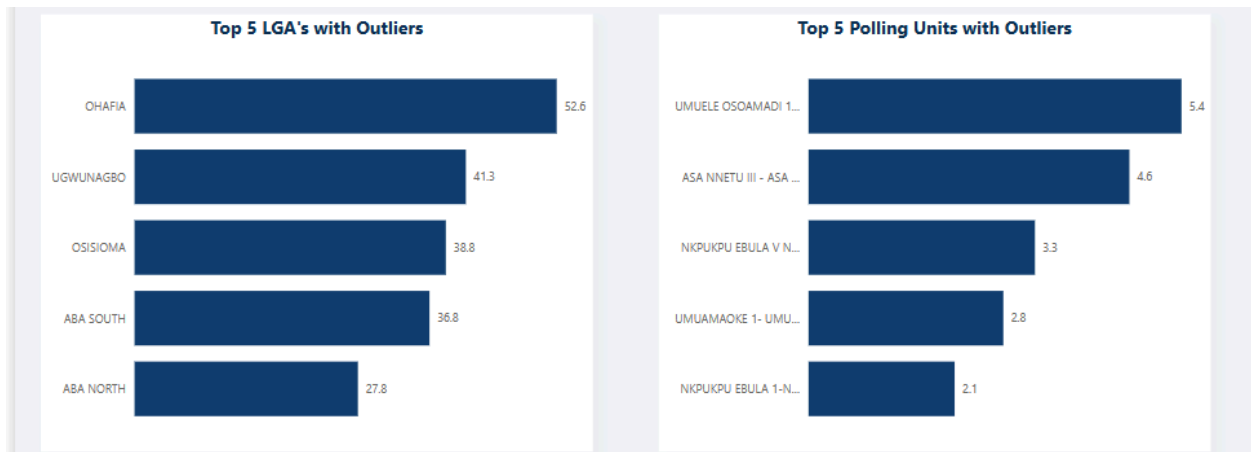
The total number of polling units (**2456**) represents where voting took place. Sudden changes or inconsistencies in this number could indicate missing or added polling stations. The total accredited voters (**238515**) reflects those verified before voting—if this number exceeds registered voters (**1167200**), it suggests ballot stuffing. Additionally, if total votes counted surpass registered voters, it's a strong sign of electoral fraud. The **Winning Party (Labour Party)** highlights the leading party, but if this aligns with polling units showing anomalies in outlier maps, further investigation is necessary.



This shows the total number of polling stations where voting occurred. If the number suddenly changes or is inconsistent, it could indicate missing or added polling units.

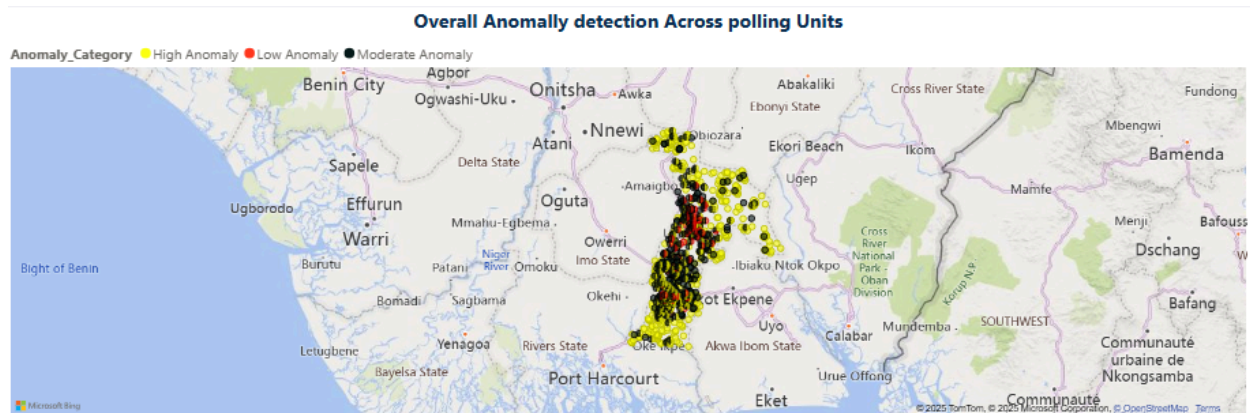


This map highlights polling units with unusually high or low votes. If some polling units have extreme differences compared to neighbors, it could mean vote rigging. Chinwe sees that Alot of the polling units in this area are red showing which mean they have an unusual number of votes has 10× the average votes—highly unusual!



The **Top 5 LGAs with Outliers** visualization highlights local governments with the most suspicious voting patterns. If **LGA X** consistently appears with extreme **Moran's I** or **Getis-Ord Gi** scores, it suggests possible vote inflation, suppression, or

irregularities. For example, if a particular LGA has a sudden surge in votes for one party, but surrounding areas don't, it could indicate tampering. This helps investigators focus on areas that need deeper auditing.

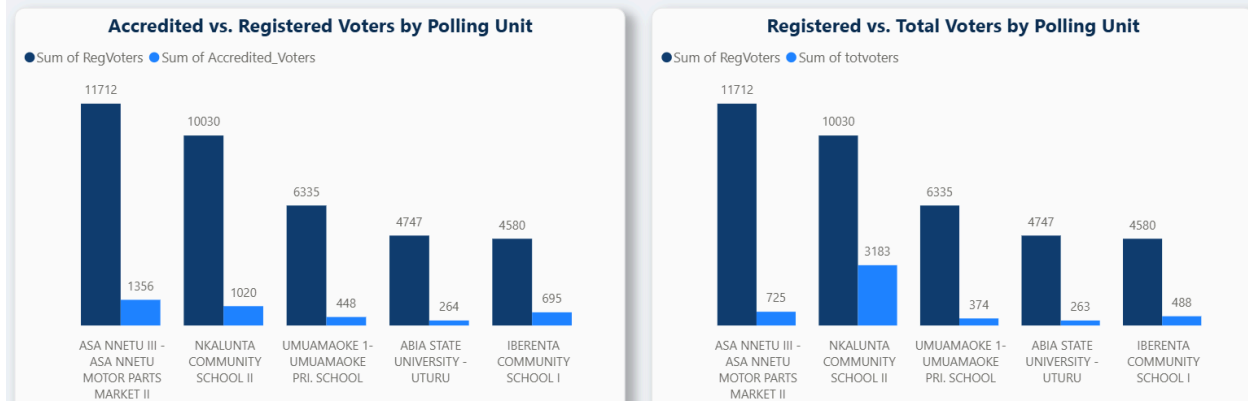


This combines multiple methods to detect suspicious patterns. If a unit appears as an anomaly in multiple analyses, there's a strong reason to investigate. Chinwe notices that **Polling Unit B** stands out in every fraud detection map.



This highlights clusters where votes are abnormally concentrated. A sudden spike in votes for one party in a small area might indicate voter suppression or artificial votes.

Voting Performance Analysis

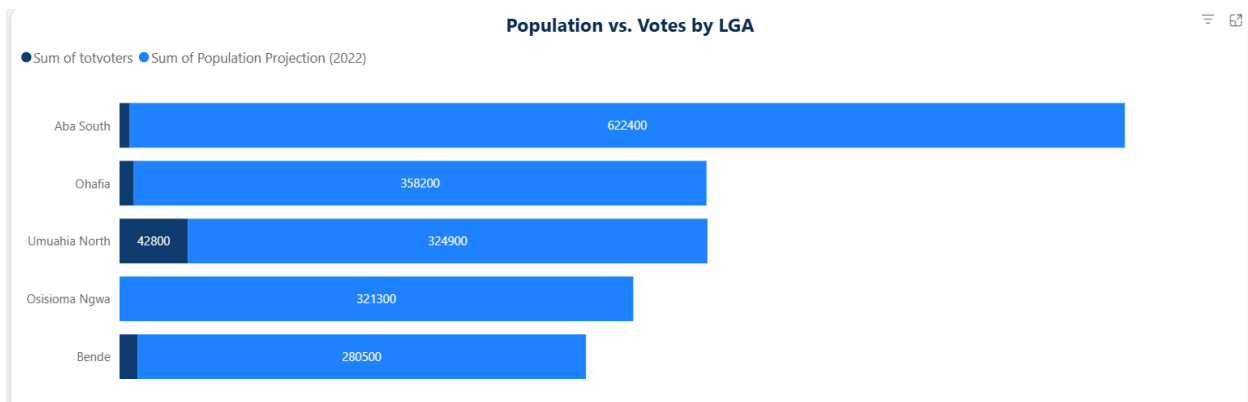


Accredited vs. Registered Voters (by Polling Unit)

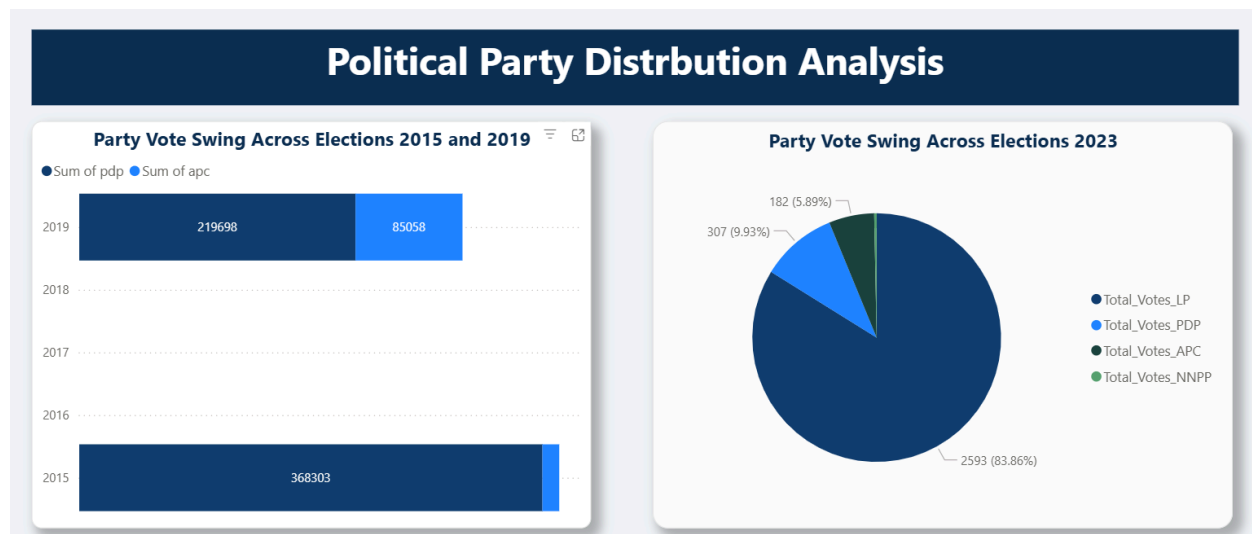
- If a polling unit has more accredited voters than registered, it suggests voter impersonation. A polling unit where accreditation is significantly higher than registrations raises red flags.

Registered vs. Total Votes (by Polling Unit)

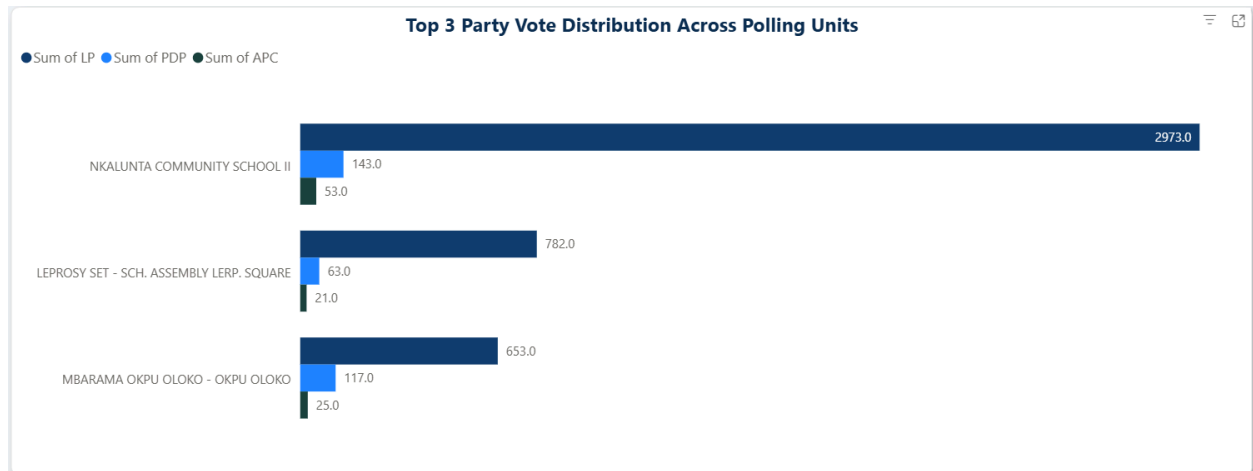
- If total votes exceed registrations, it is a strong fraud indicator. A voter turnout exceeding 100% is statistically impossible, signaling potential ballot stuffing or result manipulation.



This visual compares the population of each Local Government Area (LGA) to the number of votes cast. A significant mismatch—such as an LGA with a low population reporting an unusually high number of votes—may indicate ballot inflation. Conversely, a densely populated LGA with a surprisingly low voter turnout could suggest voter suppression. By analyzing these trends, suspicious voting patterns can be flagged for further investigation.



The Party Vote Swing (Bar Chart) tracks drastic changes in party dominance over elections. A sudden shift from one party's stronghold to another's unexpected victory may signal manipulated votes or external influence. The Party Vote Swing (2023 Pie Chart) breaks down vote distribution—if one party overwhelmingly dominates across all clusters, it could suggest voter suppression, intimidation, or large-scale vote buying. Comparing these visuals helps uncover suspicious trends over time.



The **Top 3 Parties' Votes Across Polling Units** visualization helps identify inflated votes for a single party in unexpected areas. If **Polling Unit E** records **95% of votes for one party**, while neighboring units show a more balanced distribution, it raises concerns about potential vote rigging or manipulation. Such anomalies suggest the need for further scrutiny into whether votes were artificially boosted or opposition votes suppressed.

○

You can find the live report here:

<https://app.powerbi.com/view?r=eyJrljoiYWVjNDg4NGYtYjViZC00Yzc2LWJmYTQtZjJmOTcINTMlNmlyliwidCI6Ijg3MzQ4NzM5LTg0MzUtNDYlNi05ODZhLWZjM2I0YzRjMTViNyJ9>