OUTLIER DETECTION IN AKWA-IBOM STATE ELECTION DATA USING GEOSPATIAL ANALYSIS

Methodology

Geocoding:
Concatenated the `State`, `LGA` and `PU-Name` columns to create a new column `Full Address` which gives a precise description as to its location for each polling unit recorded. Then obtained the latitude and longitude for each polling unit using a tool `Geocode by awesome tables` on google sheets

Neighbour Identification:
Akwa-Ibom which is located in the south of Nigeria, is one of the smallest states having around 7081km2 of land. There are 12 `LGA` recorded and in one of them namely `ABAK` there are 103 polling units. It is roughly the same range or more than, in the other 11 LGA. So, the polling units are closely geographically located, hence I defined a 1 km radius to identify neighbouring units.
Using geodesic distance to calculate the distance between each polling unit is quite slow due to its O(n^2) time complexity, where n is the number of polling units. So, I used `geopandas` + `rtree` which is fast for large datasets and uses spatial indexing. The function I defined using these Libraries return a list of tuples containing the neighbouring unit, which I converted to a dataframe.

```
import geopandas as gpd
from shapely.geometry import Point

def get_neighbours(polling_units_df, radius_km=1):
    # Convert DataFrame to GeoDataFrame
    gdf = gpd.GeoDataFrame(
        polling_units_df,
        geometry=gpd.points_from_xy(polling_units_df['Longitude'], polling_units_df['Latitude']),
        crs="EPSG:4326"  # WGS84 Latitude/Longitude
    )

    # Create a spatial index
    gdf_sindex = gdf.sindex

    neighbours = []

    for idx, unit in gdf.iterrows():
        # Get the bounding box for the search area
```

```
        point = unit['geometry']
        possible_matches_index = list(gdf_sindex.intersection(point.buffer(radius_km / 111.32).bounds))  # Convert km to degrees
        possible_matches = gdf.iloc[possible_matches_index]
        precise_matches = possible_matches[possible_matches['geometry'].distance(point) <= radius_km / 111.32]

        for _, other_unit in precise_matches.iterrows():
            if unit['id'] != other_unit['id']:  # Ensure not comparing with itself
                neighbours.append((unit['id'], other_unit['id']))  # Assuming 'id' is present

    return neighbours
```

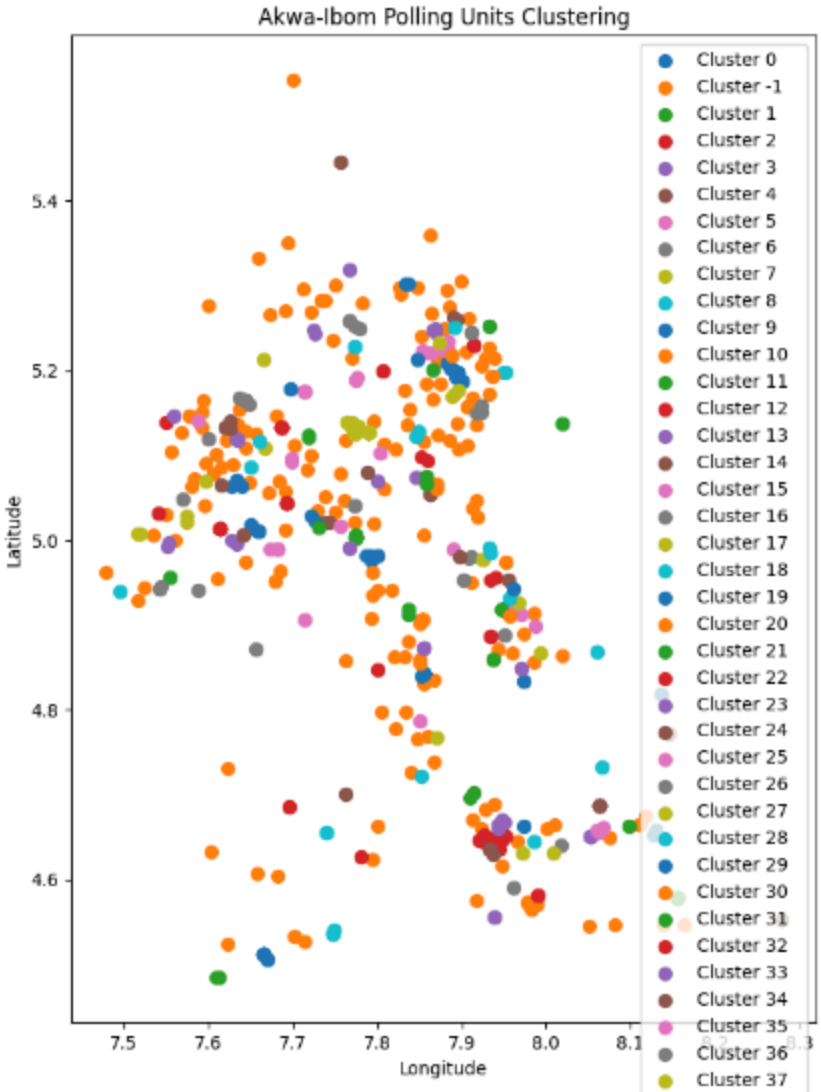
```
transformed_ndf.head()
```

[123]:

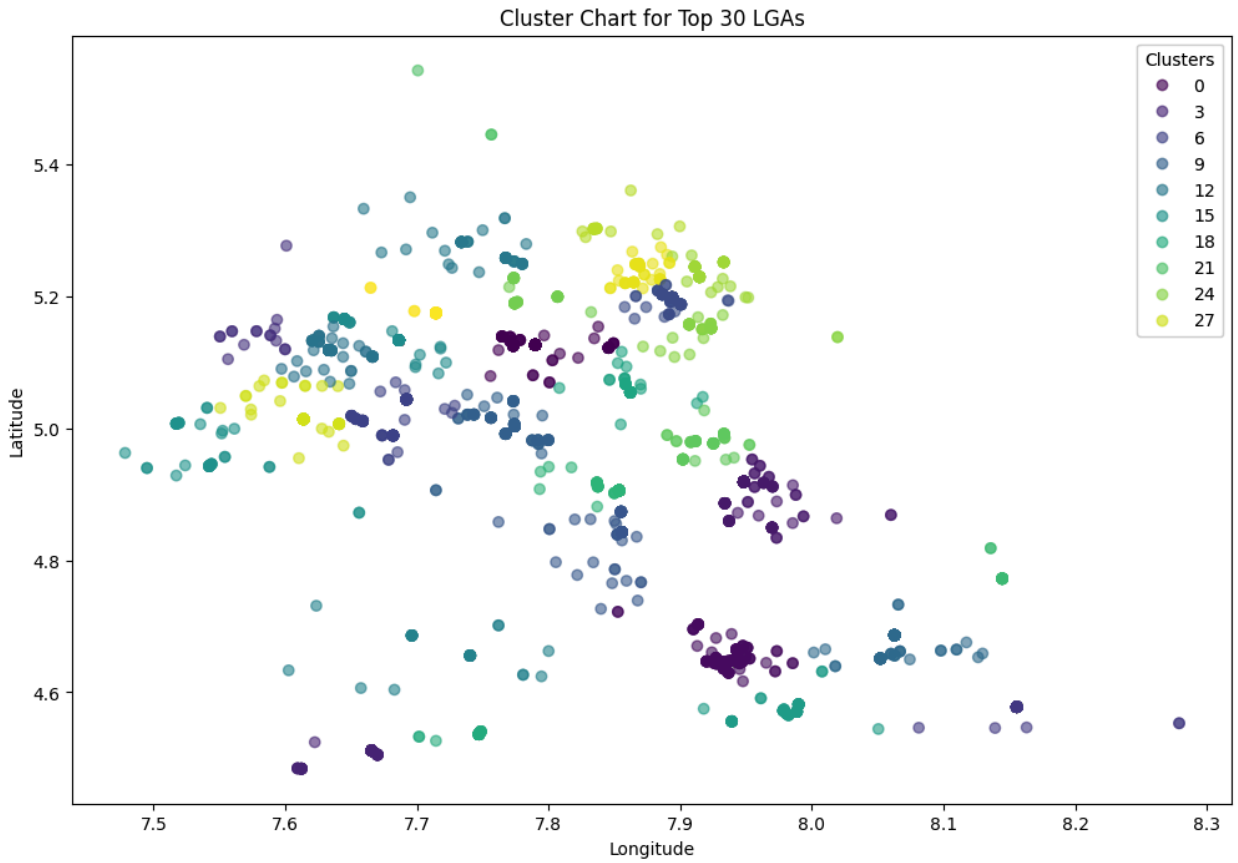
	Unit_ID	Neighbour_ID
0	0	[92, 22, 3, 2]
1	2	[0, 3]
2	3	[0, 2]
3	4	[11, 10, 73, 82, 26, 41, 39, 35, 65, 75, 42, 3...
4	5	[885, 886, 902, 40, 908, 36, 78, 904, 77, 892,...

On the Original dataframe I performed a DBSCAN clustering and plotted it. It better visualises how most polling units are closely geographically located, within their different clusters. I added a new column `Unit_id`, which is a serial index uniquely identifying each row. I used the `Unit_id` as an identifier for neighbouring polling units in the cleaned dataset.

These are the cluster for the polling units and their neighbouring units within the 1km radius, the coordinates of each unit converted to radians for Harversine distance


Akwa-Ibom Polling Units Clustering
Latitude
Longitude
5.4
5.2
5.0
4.8
4.6
7.5
7.6
7.7
7.8
7.9
8.0
Cluster 0
Cluster -1
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Cluster 10
Cluster 11
Cluster 12
Cluster 13
Cluster 14
Cluster 15
Cluster 16
Cluster 17
Cluster 18
Cluster 19
Cluster 20
Cluster 21
Cluster 22
Cluster 23
Cluster 24
Cluster 25
Cluster 26
Cluster 27
Cluster 28
Cluster 29
Cluster 30
Cluster 31
Cluster 32
Cluster 33
Cluster 34
Cluster 35
Cluster 36
Cluster 37
Cluster 38
Cluster 39
Cluster 40
Cluster 41
Cluster 42
Cluster 43
Cluster 44
Cluster 45
Cluster 46
Cluster 47
Cluster 48
Cluster 49
Cluster 50
Cluster 51
Cluster 52
Cluster 53
Cluster 54
Cluster 55
Cluster 56
Cluster 57
Cluster 58
Cluster 59
Cluster 60
Cluster 61
Cluster 62
Cluster 63
Cluster 64
Cluster 65
Cluster 66
Cluster 67

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Function to create and plot clusters for the top 30 LGAs
def plot_lga_clusters(sorted_outliers, n_clusters=30):
    # Filter data to include only relevant columns
    lga_data = sorted_outliers[['LGA', 'Latitude', 'Longitude']]

    # Get the top 30 LGAs based on frequency of occurrences
    top_30_lgas = lga_data['LGA'].value_counts().nlargest(30).index
    filtered_lga_data = lga_data[lga_data['LGA'].isin(top_30_lgas)]

    # Apply KMeans clustering
    kmeans = KMeans(n_clusters=n_clusters, random_state=0)
    filtered_lga_data['Cluster'] = kmeans.fit_predict(filtered_lga_data[['Latitude', 'Longitude']])

    # Plotting
    plt.figure(figsize=(12, 8))
    scatter = plt.scatter(filtered_lga_data['Longitude'], filtered_lga_data['Latitude'],
                          c=filtered_lga_data['Cluster'], cmap='viridis', alpha=0.6)

    # Create a legend with unique clusters
    legend1 = plt.legend(*scatter.legend_elements(), title="Clusters")
    plt.gca().add_artist(legend1)

    # Set plot title and labels
    plt.title('Cluster Chart for Top 30 LGAs')
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')

    # Show plot
    plt.show()

plot_lga_clusters(sorted_outliers)
```



Outlier Score Calculation:

The outlier score of a polling unit is the z-score for the votes of the current unit relative to its neighbouring units.

Formula:

$Z = (X - \mu) / \sigma$

```
# Function to calculate outlier scores and return a comprehensive DataFrame
def calculate_outlier_score(df, neighbour_df, threshold = 3):
    outlier_data = []
```

```
for index, unit in df.iterrows():
    unit_id = unit['id']

    # Find neighbors for the current unit
    neighbours = neighbour_df[neighbour_df['Unit_ID'] == unit_id]['Neighbour_ID']

    if neighbours.empty:
        continue

    neighbours_list = neighbours.iloc[0]  # Get the list of neighbor IDs

    for party in ['APC', 'LP', 'PDP', 'NNPP']:  # replace with actual party columns
        # Get votes for the current party from all neighboring units
        neighbour_votes = df[df['id'].isin(neighbours_list)][party]

        if not neighbour_votes.empty:
            avg_neighbor_votes = np.mean(neighbour_votes)
            std = np.std(neighbour_votes)

            # Skip calculations if the standard deviation is zero
            if std == 0:
                continue

            # Calculate z-scores for the current unit
            unit_votes = unit[party]
            unit_z_score = (unit_votes - avg_neighbor_votes) / std

            outlier_info = {
                'id': unit_id,
                'Neighbour_ID': neighbours_list,  # Use the list of neighbors
                'Party': party,
                'Outlier_Score': unit_z_score
            }

            # Identify outlier votes and calculate z-scores for neighbors
            for votes in neighbour_votes:
                z_score = (votes - avg_neighbor_votes) / std
                if np.abs(z_score) > threshold:
                    outlier_info['Outlier_Vote'] = votes
                    outlier_info['Outlier_Z_Score'] = z_score
```

```
                    break  # Assuming we only take the first significant outlier

                outlier_data.append(outlier_info)

    return pd.DataFrame(outlier_data)

neighbours_data.csv')

# Calculate outlier scores and outliers
outlier_scores_df = calculate_outlier_score(akwa, transformed_ndf, threshold = 3)


# Display the resulting DataFrame
outlier_scores_df
```

The Outlier score indicates how different the current unit's votes are compared to the average votes of its neighbours. A high absolute value of the `Outlier_Score` (typically beyond a certain threshold, like 3 or -3) indicates that the current unit is voting significantly differently from its neighbours. Although for this data set the standard deviation for the votes of different parties varies significantly (e.g., APC: ~30.86, LP: ~19.69, PDP: ~25.77, NNPP: ~2.78).

```
akwa.describe()
```

	Latitude	Longitude	Accredited_Voters	Registered_Voters	Transcription_Count	APC	LP	PDP	NNPP
count	3981.000000	3981.000000	3981.00000	3981.00000	3981.0	3981.000000	3981.000000	3981.000000	3981.000000
mean	5.078895	7.799677	113.80206	381.85255	-1.0	13.038935	7.648832	12.179352	0.516704
std	0.138628	0.075324	38.74682	181.00941	0.0	30.860331	19.692722	25.771210	2.776071
min	4.484238	7.478372	0.00000	122.00000	-1.0	0.000000	0.000000	0.000000	0.000000
25%	5.126348	7.789896	99.00000	298.00000	-1.0	0.000000	0.000000	0.000000	0.000000
50%	5.126348	7.789896	99.00000	298.00000	-1.0	0.000000	0.000000	0.000000	0.000000
75%	5.126348	7.789896	99.00000	298.00000	-1.0	0.000000	0.000000	0.000000	0.000000
max	5.541972	8.279080	396.00000	1708.00000	-1.0	324.000000	221.000000	181.000000	66.000000

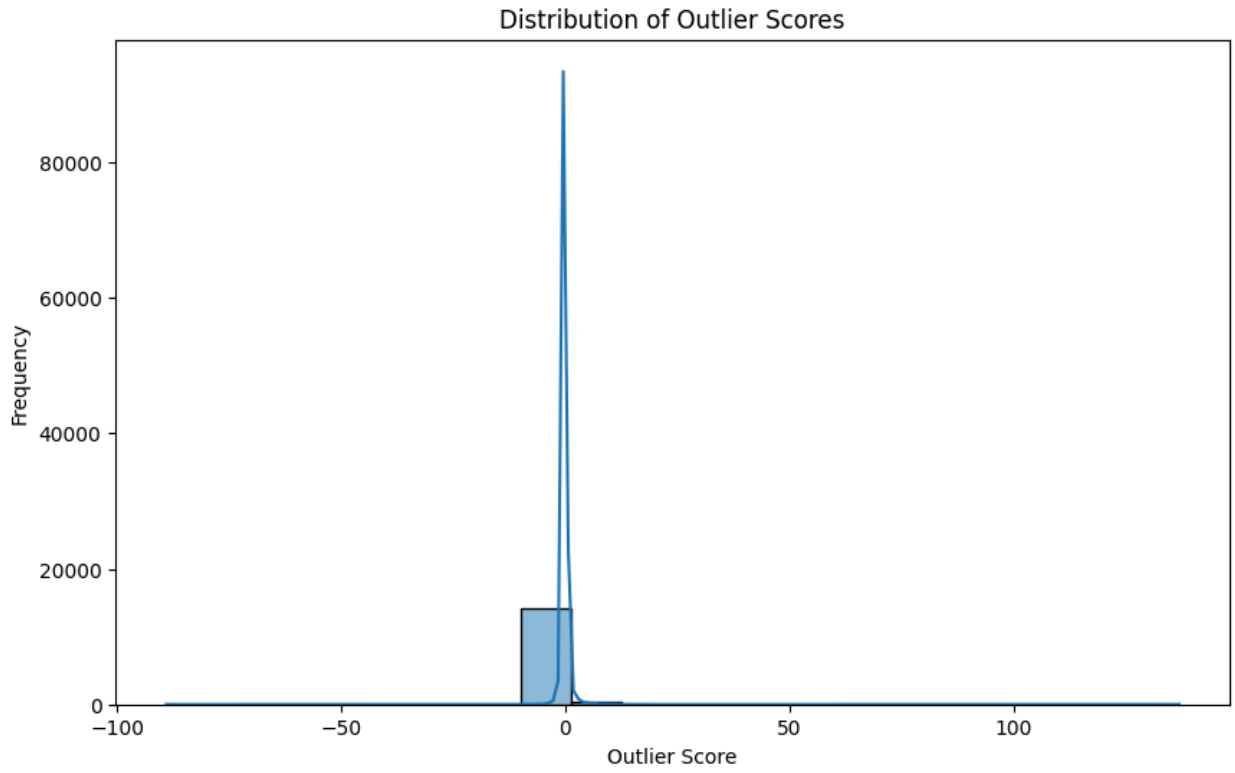
Considering the differences in the standard deviations for the different parties in the data, a threshold of 3 should work well for identifying significant outliers across different parties. A threshold of 3 is widely accepted yet it might miss some smaller yet significant deviations, especially for LP and NNPP with very low standard deviations. But lowering it would identify too many false positives. Given these considerations, using a threshold of 3 was reasonable.

```
merged_akwa = pd.merge(akwa, outlier_scores_df, on='id', how='left')
merged_akwa.head(20)
sorted_outliers = merged_akwa.sort_values(by='Outlier_Score', ascending=False)
```

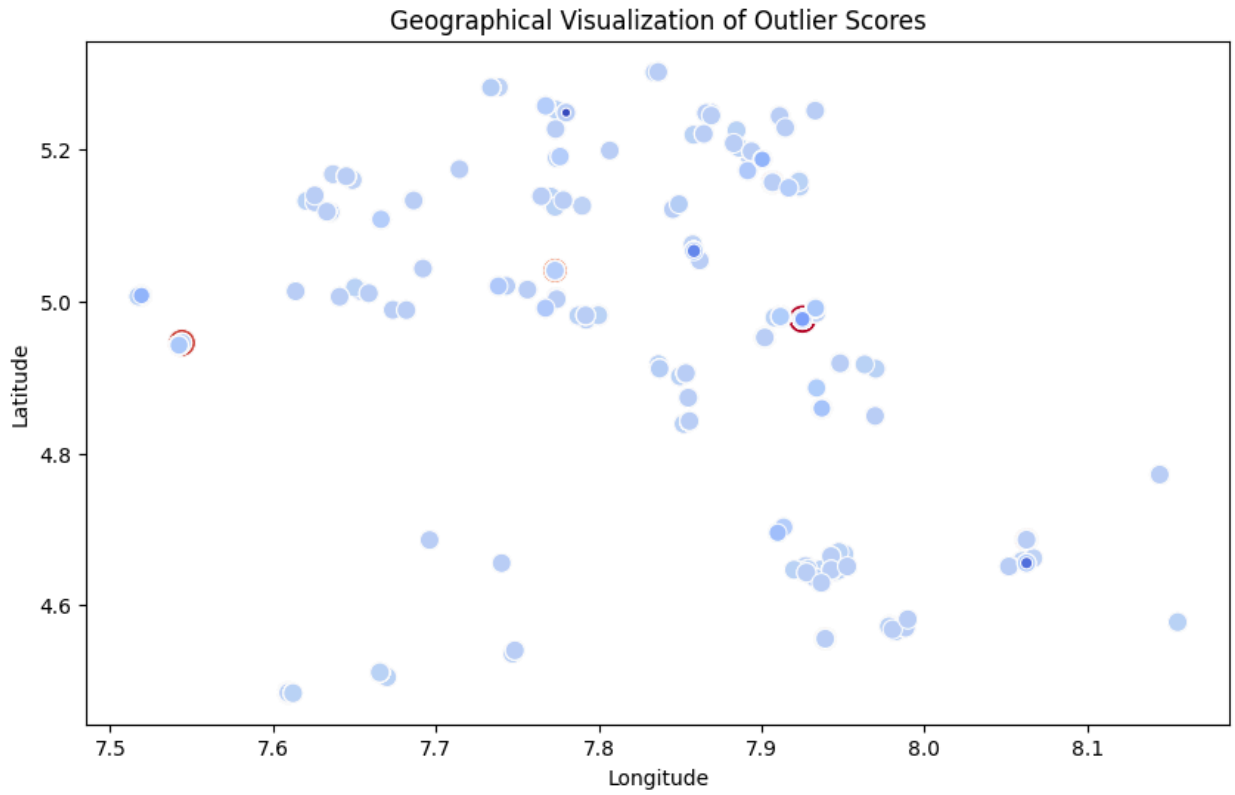
Link to the cleaned and sorted dataset by the Outlier Scores:
https://drive.google.com/file/d/134AdizTbnsuuKa9wXsYWlheUYnAUw7cL/view?usp=sharing

Visuals:

The distribution appears somewhat skewed towards the right, suggesting there might be more outliers with higher scores than those with lower scores, which is true.
The bars throughout the X-axis seem to have a moderate height, indicating that there's a spread of outlier scores across the range, and no single score is overwhelmingly frequent.



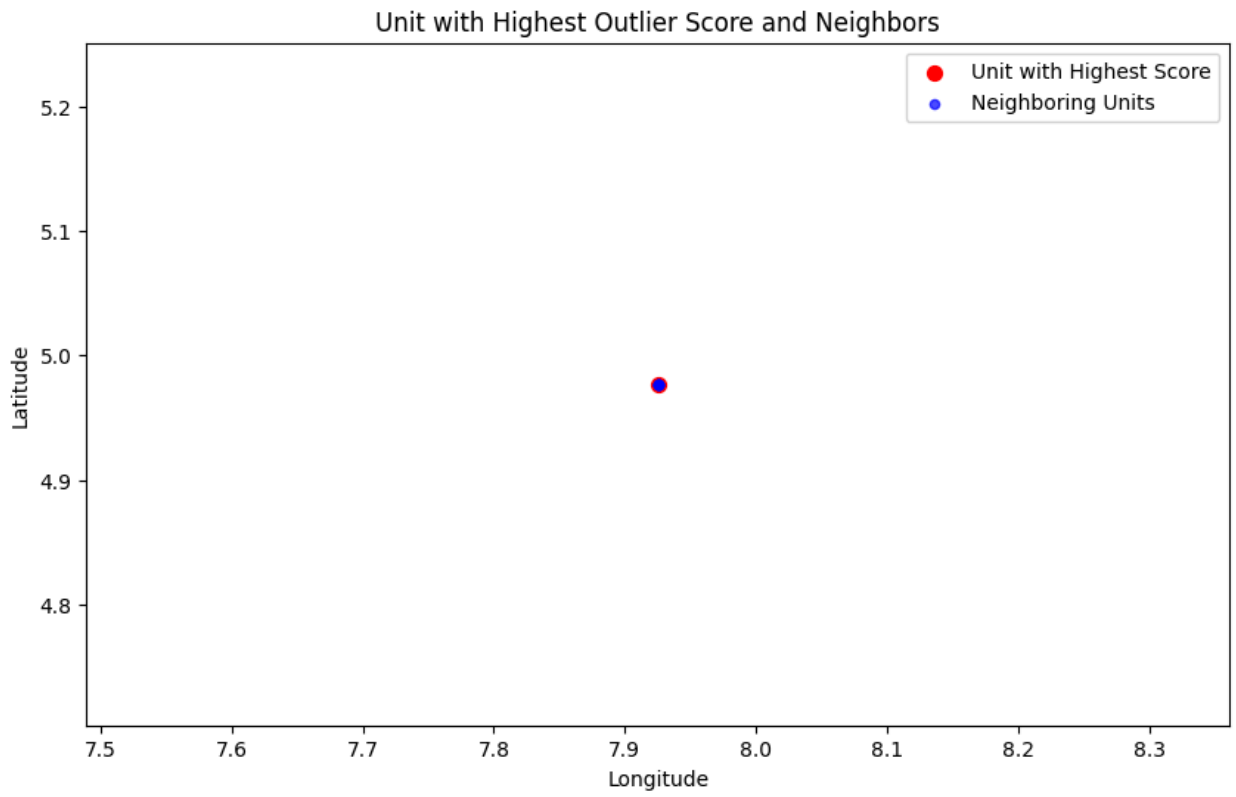
Here we only see the respective polling units in which their outlier scores have been calculated, we can not tell which unit is the outlier within its cluster yet. But we can see some units that have no neighbour units and as you can guess those units will probably behave differently from the rest.

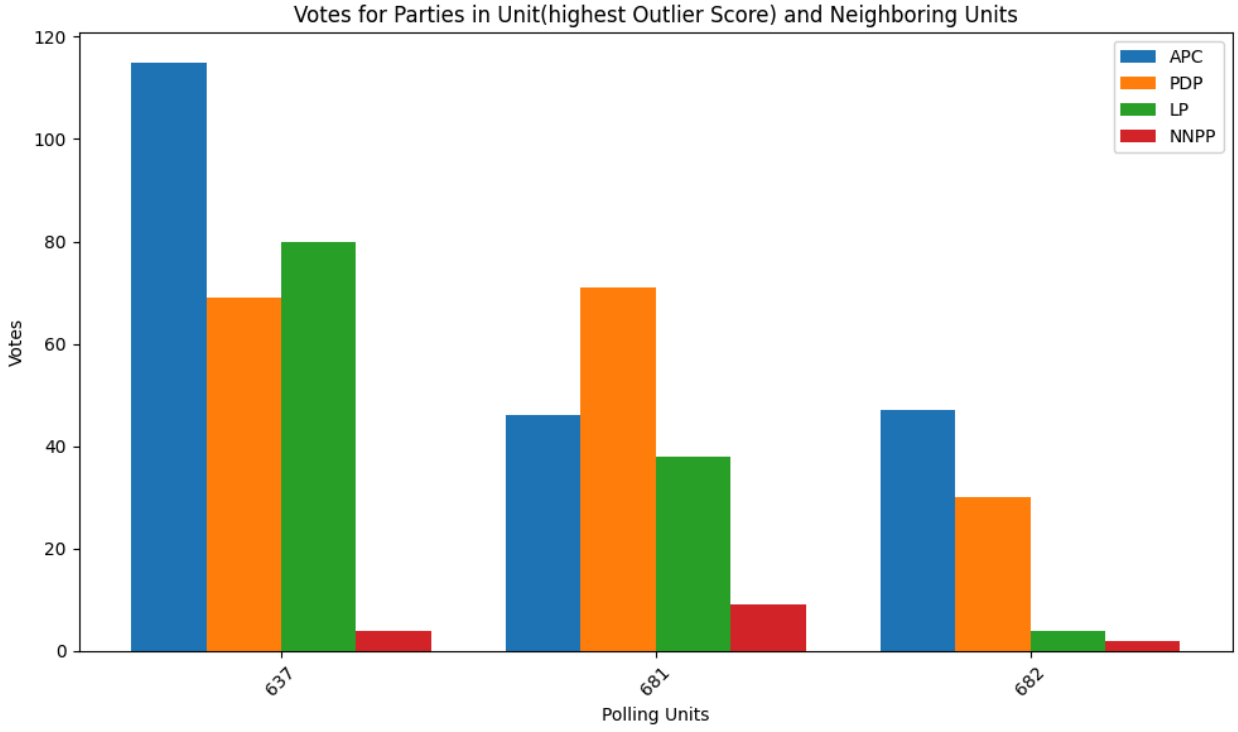


Observations:

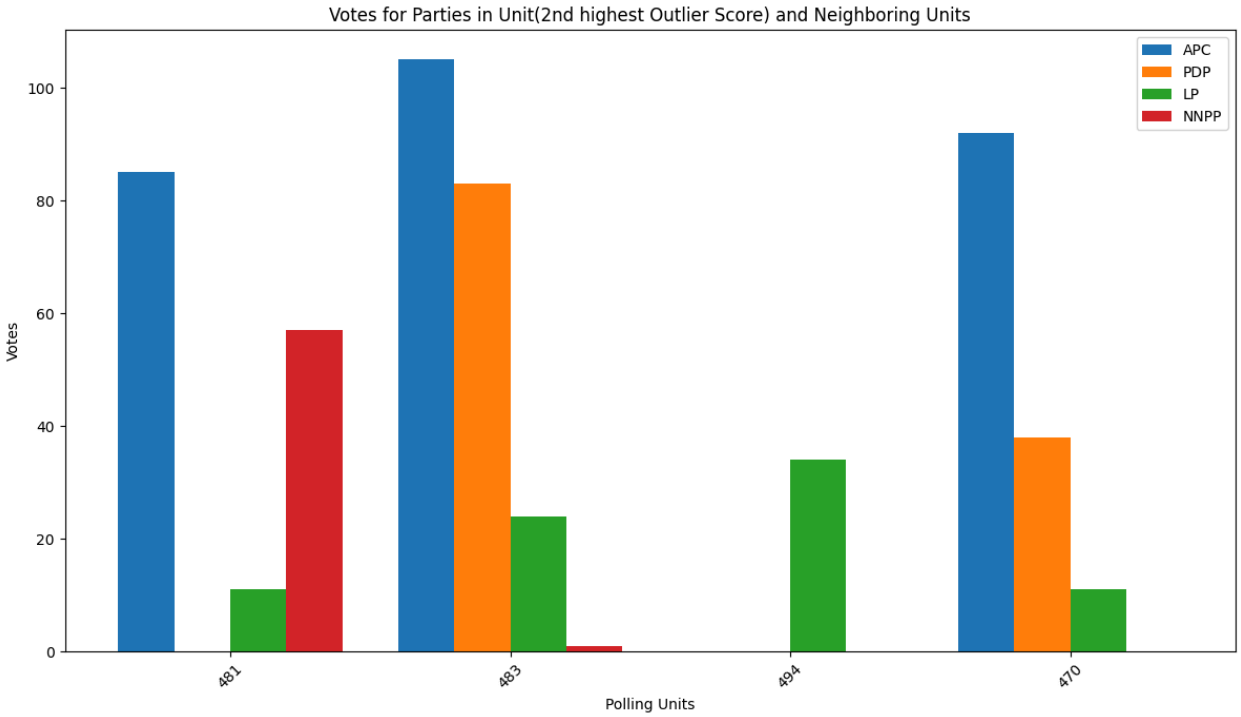
- 3,979 polling units were recorded and in NKWOT Ward alone, specifically 'AKWA IBOM IKONO VILLAGE SQUARE, IKOT AKPAN INYANG ' there are 2,981 polling units. Though in all 2,981 polling units in that ward there were 0 votes casted for all parties. This is strange as there are 99 accredited voters out of the 298 that registered.

- The highest Outlier score falls in `AKWA IBOM EKET TOWN HALL ATA IDUNG AFAHA EKET ` under the `Full Address` column. As shown in the scatter plot below. This unit's votes show abnormal behaviour, with APC having higher votes than usual than its surrounding units. In this unit APC had 115 votes and in the neighbouring had 46, 47 respectively. In this cluster APC is leading but for the particular unit, the spike in number of votes is suspicious.

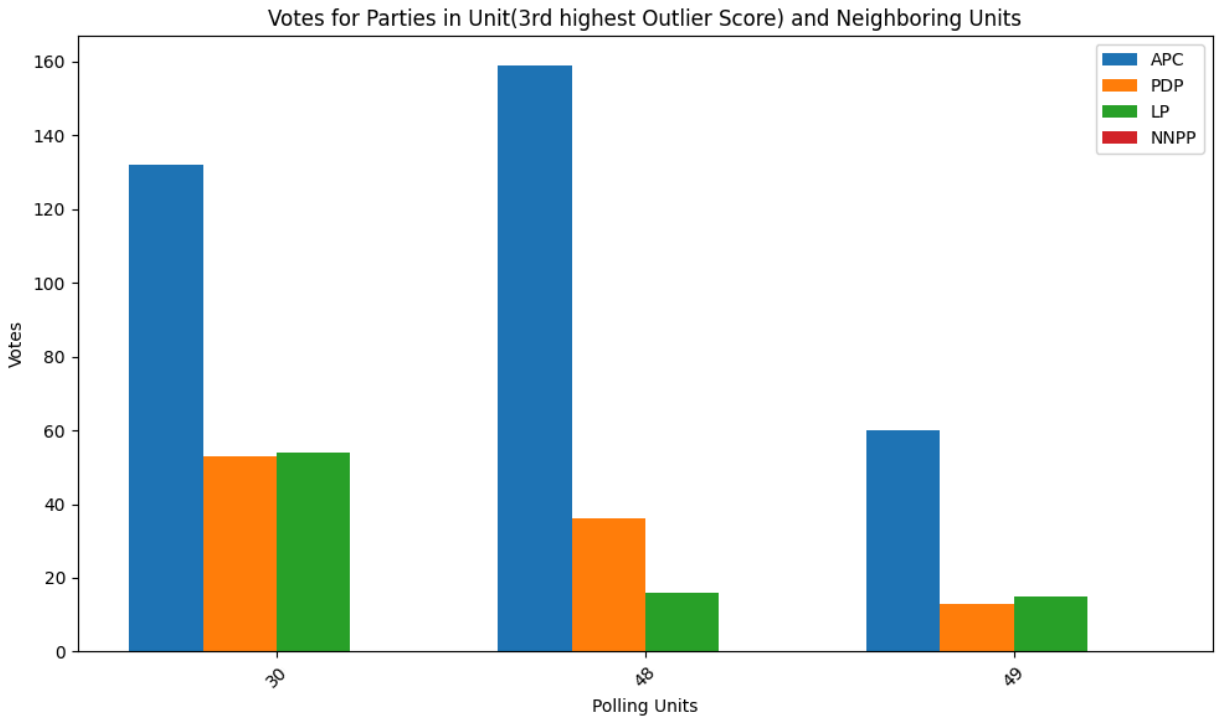

Unit with Highest Outlier Score and Neighbors
Unit with Highest Score
Neighboring Units
Latitude
Longitude
5.2
5.1
5.0
4.9
4.8
7.5
7.6
7.7
7.8
7.9
8.0
8.1
8.2
8.3


Votes for Parties in Unit(highest Outlier Score) and Neighboring Units
APC
PDP
LP
NNPP
Votes
Polling Units
120
100
80
60
40
20
0
637
681
682

- This is the unit with the 2nd highest outlier score for NNPP. In the neighbouring units the votes cast for NNPP are 0 and 1. But in `PRY SCH, OBON EBOT`, 57 votes were cast for NNPP a great deviation from the general trend of its neighbours.



- Here the outlier score is for the LP as is easily seen in the dataset there are less votes for LP and close to none for all units. But in this cluster we see LP has quite a few supporters, with votes cast below 20. But we can see the difference in votes in unit 30.



Findings:

- **Top 3 Outliers**: The analysis highlighted the top 3 polling units with the highest outlier scores. These units showed voting patterns that were significantly different from their neighbouring units, indicating potential irregularities or influences.
- **Geographical Patterns**: Certain geographical areas showed more significant deviations, which could be indicative of localised influences or anomalies
- **Vote Manipulation Indicators**: The outlier scores provide a quantitative measure to identify potential vote manipulation or irregularities. Units with high outlier scores warrant further investigation.
- **Importance of Geospatial Analysis**: This analysis underscores the importance of incorporating geospatial techniques in election data analysis to ensure the integrity and transparency of election results.

Conclusion

The analysis has successfully identified polling units with significant deviations in voting patterns, indicating potential irregularities. This approach not only helps in ensuring election integrity but also provides a framework for future analyses to maintain transparent and accurate

election results. The findings emphasise the need for continuous monitoring and investigation of outlier units to uphold the credibility of the electoral process.