HNG_STAGE8_A

March 27, 2025

[122]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[ ]:
```
#run this code to install the libraries, if you don't have them
!pip install networkx pandas plotly numpy matplotlib seaborn
```

[ ]:
```
#importing the necessary requirements
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[ ]:
```
#reading the cross-checked dataset into a pandas dataframe
df = pd.read_csv("/content/drive/MyDrive/ANAMBRA_crosschecked.csv")
```

0.0.1 BASIC DATA EXPLORATION

[ ]:
```
df.shape
```

[ ]:
```
(3679, 19)
```

[ ]:
```
df.describe()
```

[ ]:
```
       Accredited_Voters  Registered_Voters  Transcription_Count          APC  \
count        3679.000000        3679.000000               3679.0  3679.000000
mean          115.236477         450.597445                 -1.0     1.260669
std            79.122946         333.768034                  0.0     7.910370
min             0.000000           1.000000                 -1.0     0.000000
25%            56.000000         203.000000                 -1.0     0.000000
50%           103.000000         397.000000                 -1.0     0.000000
75%           159.000000         640.500000                 -1.0     1.000000
max           582.000000        3770.000000                 -1.0   350.000000

                LP          PDP          NNPP
```

count	3679.000000	3679.000000	3679.000000
mean	103.258766	2.413971	0.556401
std	77.716562	11.511426	5.703382
min	0.000000	0.000000	0.000000
25%	45.000000	0.000000	0.000000
50%	91.000000	1.000000	0.000000
75%	144.500000	2.000000	0.000000
max	574.000000	465.000000	251.000000

<google.colab._quickchart_helpers.SectionTitle at 0x7c35450e06d0>

```
from matplotlib import pyplot as plt
_df_0['Accredited_Voters'].plot(kind='hist', bins=20, title='Accredited_Voters')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_1['Registered_Voters'].plot(kind='hist', bins=20, title='Registered_Voters')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_2['Transcription_Count'].plot(kind='hist', bins=20, title='Transcription_Count')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_3['APC'].plot(kind='hist', bins=20, title='APC')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

<google.colab._quickchart_helpers.SectionTitle at 0x7c3544eec910>

```
from matplotlib import pyplot as plt
_df_4.plot(kind='scatter', x='Accredited_Voters', y='Registered_Voters', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_5.plot(kind='scatter', x='Registered_Voters', y='Transcription_Count', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_6.plot(kind='scatter', x='Transcription_Count', y='APC', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_7.plot(kind='scatter', x='APC', y='LP', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

<google.colab._quickchart_helpers.SectionTitle at 0x7c3544eed990>

```
from matplotlib import pyplot as plt
_df_8['Accredited_Voters'].plot(kind='line', figsize=(8, 4), title='Accredited_Voters')
```

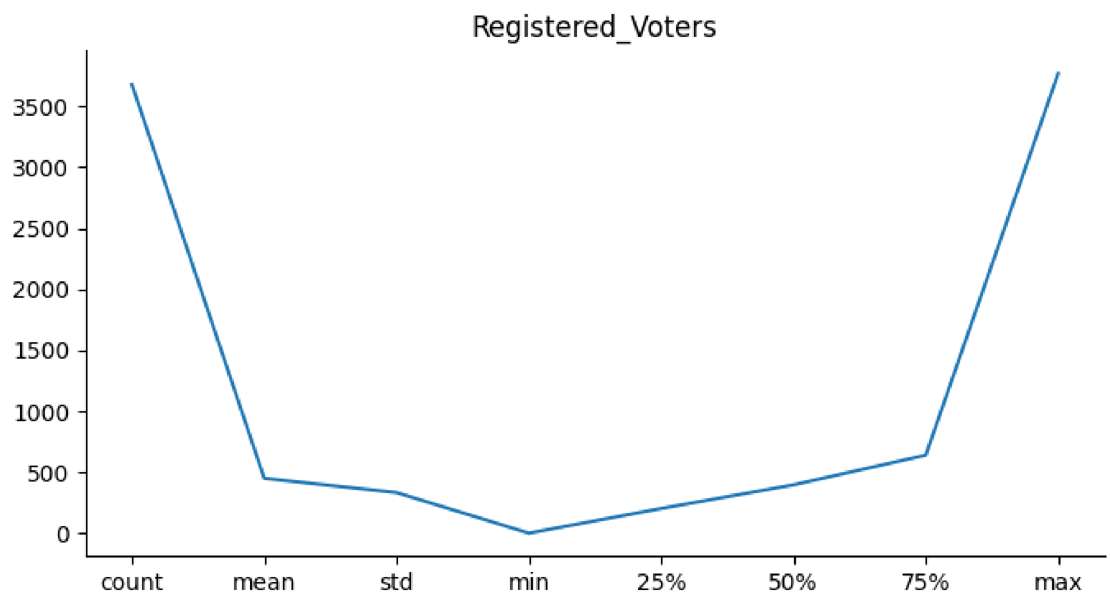
```
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_9['Registered_Voters'].plot(kind='line', figsize=(8, 4),␣
↪title='Registered_Voters')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_10['Transcription_Count'].plot(kind='line', figsize=(8, 4),␣
↪title='Transcription_Count')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_11['APC'].plot(kind='line', figsize=(8, 4), title='APC')
plt.gca().spines[['top', 'right']].set_visible(False)
```

[ ]:
```
from matplotlib import pyplot as plt
_df_9['Registered_Voters'].plot(kind='line', figsize=(8, 4),␣
↪title='Registered_Voters')
plt.gca().spines[['top', 'right']].set_visible(False)
```



[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   State                   3679 non-null   object
```

```
 1   LGA                      3679 non-null   object
 2   Ward                     3679 non-null   object
 3   PU-Code                  3679 non-null   object
 4   PU-Name                  3679 non-null   object
 5   Accredited_Voters        3679 non-null   int64
 6   Registered_Voters        3679 non-null   int64
 7   Results_Found            3679 non-null   bool
 8   Transcription_Count      3679 non-null   int64
 9   Result_Sheet_Stamped     3679 non-null   bool
 10  Result_Sheet_Corrected   3679 non-null   bool
 11  Result_Sheet_Invalid     3679 non-null   bool
 12  Result_Sheet_Unclear     3679 non-null   bool
 13  Result_Sheet_Unsigned    3679 non-null   object
 14  APC                      3679 non-null   int64
 15  LP                       3679 non-null   int64
 16  PDP                      3679 non-null   int64
 17  NNPP                     3679 non-null   int64
 18  Results_File             3679 non-null   object
dtypes: bool(5), int64(7), object(7)
memory usage: 420.5+ KB
```

[ ]:
```
df.head()
```

[ ]:
```
     State     LGA       Ward       PU-Code             PU-Name  \
0  ANAMBRA  AGUATA  ACHINA   I  04-01-01-001  ST. CHARLE'S SCHOOL
1  ANAMBRA  AGUATA  ACHINA   I  04-01-01-005       AMANKWU SQUARE
2  ANAMBRA  AGUATA  ACHINA   I  04-01-01-006     COOPERATIVE HALL
3  ANAMBRA  AGUATA  ACHINA   I  04-01-01-008      OCHIEOBU SQUARE
4  ANAMBRA  AGUATA  ACHINA   I  04-01-01-010   OYE MOTOR PARK  II

   Accredited_Voters  Registered_Voters  Results_Found  Transcription_Count  \
0                171                630           True                   -1
1                153                500           True                   -1
2                121                386           True                   -1
3                134                426           True                   -1
4                 63                166           True                   -1

   Result_Sheet_Stamped  Result_Sheet_Corrected  Result_Sheet_Invalid  \
0                 False                   False                 False
1                 False                   False                 False
2                 False                   False                 False
3                 False                   False                 False
4                 False                   False                 False

   Result_Sheet_Unclear Result_Sheet_Unsigned  APC   LP  PDP  NNPP  \
0                 False               UNKNOWN    0    0    0     0
1                 False               UNKNOWN    3  142    0     1
```

```
2                  False                UNKNOWN    0    0    0     0
3                  False                UNKNOWN    0  124    4     1
4                  False                UNKNOWN    4   57    0     0

                                              Results_File
0  https://docs.inecelectionresults.net/elections…
1  https://docs.inecelectionresults.net/elections…
2  https://docs.inecelectionresults.net/elections…
3  https://docs.inecelectionresults.net/elections…
4  https://docs.inecelectionresults.net/elections…
```

[ ]:
```python
duplicates = df[df.duplicated()]
print(duplicates)
```

```
Empty DataFrame
Columns: [State, LGA, Ward, PU-Code, PU-Name, Accredited_Voters,
Registered_Voters, Results_Found, Transcription_Count, Result_Sheet_Stamped,
Result_Sheet_Corrected, Result_Sheet_Invalid, Result_Sheet_Unclear,
Result_Sheet_Unsigned, APC, LP, PDP, NNPP, Results_File]
Index: []
```

[ ]:
```python
# Get unique count for all object columns
unique_counts = df.select_dtypes(include='object').nunique()

print(unique_counts)
```

```
State                       1
LGA                        21
Ward                      296
PU-Code                  3679
PU-Name                  3371
Result_Sheet_Unsigned       1
Results_File             3525
dtype: int64
```

0.0.2 LATITUDE AND LONGITUDE values using Geocoding

[ ]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3679 non-null   object
 1   lga                     3679 non-null   object
 2   ward                    3679 non-null   object
 3   pu-code                 3679 non-null   object
```

```
 4   pu-name                 3679 non-null   object
 5   accredited_voters       3679 non-null   int64
 6   registered_voters       3679 non-null   int64
 7   results_found           3679 non-null   bool
 8   transcription_count     3679 non-null   int64
 9   result_sheet_stamped    3679 non-null   bool
 10  result_sheet_corrected  3679 non-null   bool
 11  result_sheet_invalid    3679 non-null   bool
 12  result_sheet_unclear    3679 non-null   bool
 13  result_sheet_unsigned   3679 non-null   object
 14  apc                     3679 non-null   int64
 15  lp                      3679 non-null   int64
 16  pdp                     3679 non-null   int64
 17  nnpp                    3679 non-null   int64
 18  results_file            3679 non-null   object
dtypes: bool(5), int64(7), object(7)
memory usage: 420.5+ KB
```

[ ]:
```
# standardizing the dataframe
df.columns = df.columns.str.lower()
```

[ ]:
```
import pandas as pd
import requests
import time

# Google Geocoding API Key
API_KEY = "AIzaSyB4jKVUQN14ExqzVE3IjbKGIDHF-EBLkBo"

# Dictionary to cache API responses and avoid duplicate requests
cache = {}

# Function to get latitude and longitude with caching
def get_lat_lon(state, lga, ward, pu_name, pu_code):
    """
    Queries Google Geocoding API to get latitude & longitude for a polling unit.
    Uses caching to avoid redundant requests.
    """
    pu_identifier = pu_name if pd.notna(pu_name) else pu_code
    query = f"{pu_identifier}, {ward}, {lga}, {state}, Nigeria"

    # Check cache to avoid duplicate API requests
    if query in cache:
        return cache[query]

    url = f"https://maps.googleapis.com/maps/api/geocode/json?
 ↪address={query}&key={API_KEY}"
```

```python
    try:
        response = requests.get(url)
        response.raise_for_status()
        data = response.json()

        if data['status'] == 'OK':
            lat = data['results'][0]['geometry']['location']['lat']
            lon = data['results'][0]['geometry']['location']['lng']
            cache[query] = (lat, lon)  # Store result in cache
            return lat, lon
        else:
            print(f"Geocoding failed for {query}: {data['status']}")
            return None, None
    except Exception as e:
        print(f"Error fetching coordinates for {query}: {e}")
        return None, None


# Add Latitude and Longitude columns with progress tracking
batch_size = 100  # Save every 100 rows
output_file = "polling_units_with_coordinates.csv"

#.loc[] did not update correctly in a loop. using .at[] to fix it
for i in range(0, len(df), batch_size):
    print(f"Processing rows {i} to {i + batch_size}...")

    for j in range(i, min(i + batch_size, len(df))):
        lat, lon = get_lat_lon(df.at[j, 'state'], df.at[j, 'lga'], df.at[j,␣
 ↪'ward'], df.at[j, 'pu-name'], df.at[j, 'pu-code'])
        df.at[j, 'latitude'] = lat
        df.at[j, 'longitude'] = lon

    # Save progress after each batch
    df.to_csv(output_file, index=False)
    time.sleep(1)  # Prevent API rate limiting

print("  Done! Data saved to:", output_file)
```

```
Processing rows 0 to 100…
Processing rows 100 to 200…
Processing rows 200 to 300…
Processing rows 300 to 400…
Processing rows 400 to 500…
Processing rows 500 to 600…
Processing rows 600 to 700…
Processing rows 700 to 800…
Processing rows 800 to 900…
```

```
Processing rows 900 to 1000…
Processing rows 1000 to 1100…
Processing rows 1100 to 1200…
Processing rows 1200 to 1300…
Processing rows 1300 to 1400…
Processing rows 1400 to 1500…
Processing rows 1500 to 1600…
Processing rows 1600 to 1700…
Processing rows 1700 to 1800…
Processing rows 1800 to 1900…
Processing rows 1900 to 2000…
Processing rows 2000 to 2100…
Processing rows 2100 to 2200…
Processing rows 2200 to 2300…
Processing rows 2300 to 2400…
Processing rows 2400 to 2500…
Processing rows 2500 to 2600…
Processing rows 2600 to 2700…
Processing rows 2700 to 2800…
Processing rows 2800 to 2900…
Processing rows 2900 to 3000…
Processing rows 3000 to 3100…
Geocoding failed for NWAKPADULU ESTATE SQUARE (OPEN SPACE OPPOSITE U & I
PHARMACY), AGU OKA, AWKA SOUTH, ANAMBRA, Nigeria: ZERO_RESULTS
Processing rows 3100 to 3200…
Processing rows 3200 to 3300…
Processing rows 3300 to 3400…
Processing rows 3400 to 3500…
Processing rows 3500 to 3600…
Processing rows 3600 to 3700…
Geocoding failed for NGENE SQUARE (OPEN SPACE AT SS JOHN & PAUL CATHOLIC
CHURCH), AWKA  V, AWKA SOUTH, ANAMBRA, Nigeria: ZERO_RESULTS
 Done! Data saved to: polling_units_with_coordinates.csv
```

[ ]:
```python
# Convert cache to DataFrame
geocoded_df = pd.DataFrame(cache.items(), columns=["query", "coordinates"])
```

[ ]:
```python
# Saving the geocoded CSV file
geocoded_df.to_csv("geocoded_polling_units.csv", index=False)
```

[ ]:
```python
df2 = pd.read_csv("/content/polling_units_with_coordinates.csv")
```

[ ]:
```python
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 21 columns):
 #   Column                   Non-Null Count  Dtype
```

```
---  ------                   --------------  -----
 0   state                    3679 non-null   object
 1   lga                      3679 non-null   object
 2   ward                     3679 non-null   object
 3   pu-code                  3679 non-null   object
 4   pu-name                  3679 non-null   object
 5   accredited_voters        3679 non-null   int64
 6   registered_voters        3679 non-null   int64
 7   results_found            3679 non-null   bool
 8   transcription_count      3679 non-null   int64
 9   result_sheet_stamped     3679 non-null   bool
 10  result_sheet_corrected   3679 non-null   bool
 11  result_sheet_invalid     3679 non-null   bool
 12  result_sheet_unclear     3679 non-null   bool
 13  result_sheet_unsigned    3679 non-null   object
 14  apc                      3679 non-null   int64
 15  lp                       3679 non-null   int64
 16  pdp                      3679 non-null   int64
 17  nnpp                     3679 non-null   int64
 18  results_file             3679 non-null   object
 19  latitude                 3677 non-null   float64
 20  longitude                3677 non-null   float64
dtypes: bool(5), float64(2), int64(7), object(7)
memory usage: 478.0+ KB
```

0.0.3 Manual Google Map search for the two failed poiling units

6.196884147507653, 7.070810480791814 - NGENE SQUARE (OPEN SPACE AT SS JOHN & PAUL CATHOLIC CHURCH), AWKA V, AWKA SOUTH, ANAMBRA, Nigeria

6.237219899633787, 7.106972291586833 - NWAKPADULU ESTATE SQUARE (OPEN SPACE OPPOSITE U & I PHARMACY), AGU OKA, AWKA SOUTH, ANAMBRA, Nigeria

[ ]:
```python
# Dictionary of manually collected coordinates
manual_coords = {
    "NWAKPADULU ESTATE SQUARE (OPEN SPACE OPPOSITE U & I PHARMACY)": (6.
 ↪237219899633787, 7.106972291586833),
    "NGENE SQUARE (OPEN SPACE AT SS JOHN & PAUL CATHOLIC CHURCH)": (6.
 ↪196884147507653, 7.070810480791814),
}

# Apply the manual coordinates where `pu-name` matches
for pu_name, (lat, lon) in manual_coords.items():
    cond = df2["pu-name"].str.contains(pu_name, na=False, case=False,␣
 ↪regex=False)
    df2.loc[cond, "latitude"] = lat
    df2.loc[cond, "longitude"] = lon
```

[ ]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3679 non-null   object
 1   lga                     3679 non-null   object
 2   ward                    3679 non-null   object
 3   pu-code                 3679 non-null   object
 4   pu-name                 3679 non-null   object
 5   accredited_voters       3679 non-null   int64
 6   registered_voters       3679 non-null   int64
 7   results_found           3679 non-null   bool
 8   transcription_count     3679 non-null   int64
 9   result_sheet_stamped    3679 non-null   bool
 10  result_sheet_corrected  3679 non-null   bool
 11  result_sheet_invalid    3679 non-null   bool
 12  result_sheet_unclear    3679 non-null   bool
 13  result_sheet_unsigned   3679 non-null   object
 14  apc                     3679 non-null   int64
 15  lp                      3679 non-null   int64
 16  pdp                     3679 non-null   int64
 17  nnpp                    3679 non-null   int64
 18  results_file            3679 non-null   object
 19  latitude                3679 non-null   float64
 20  longitude               3679 non-null   float64
dtypes: bool(5), float64(2), int64(7), object(7)
memory usage: 478.0+ KB
```

[ ]:
```
#taking a close inspection at the inputed values
df2[df2["pu-name"].isin(manual_coords.keys())][["pu-name", "latitude",␣
↪"longitude"]]
```

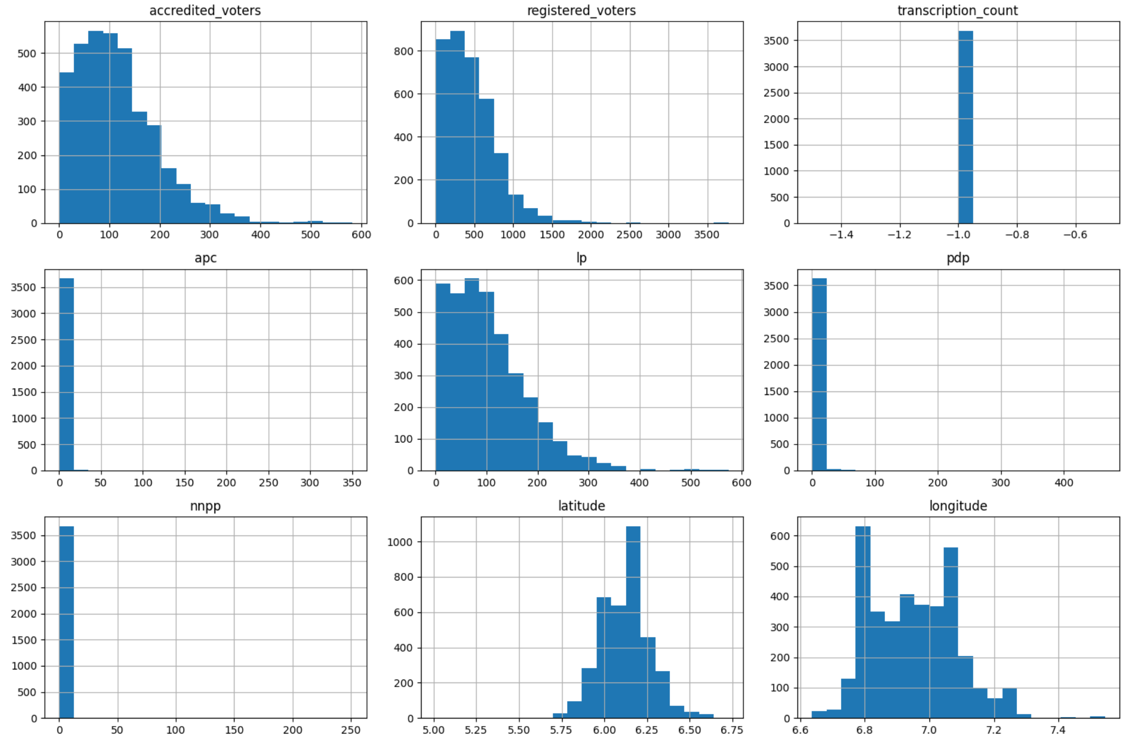
[ ]:

	pu-name	latitude	longitude
3063	NWAKPADULU ESTATE SQUARE (OPEN SPACE OPPOSITE …	6.237220	7.106972
3631	NGENE SQUARE (OPEN SPACE AT SS JOHN & PAUL CAT…	6.196884	7.070810

[ ]:
```
#saving the sorted excel file
df2.to_csv("Geo_Anambra_data.csv", index=False)
```

[ ]:
```
numeric_columns = ['accredited_voters', 'registered_voters',␣
↪'transcription_count', 'apc', 'lp', 'pdp', 'nnpp', 'latitude', 'longitude']

df[numeric_columns].hist(figsize=(15, 10), bins=20)
plt.tight_layout()
```

```
plt.show()
```



0.1 Advanced Neighbor Identification

[ ]:
```
!pip install haversine
```

```
Collecting haversine
  Downloading haversine-2.9.0-py2.py3-none-any.whl.metadata (5.8 kB)
Downloading haversine-2.9.0-py2.py3-none-any.whl (7.7 kB)
Installing collected packages: haversine
Successfully installed haversine-2.9.0
```

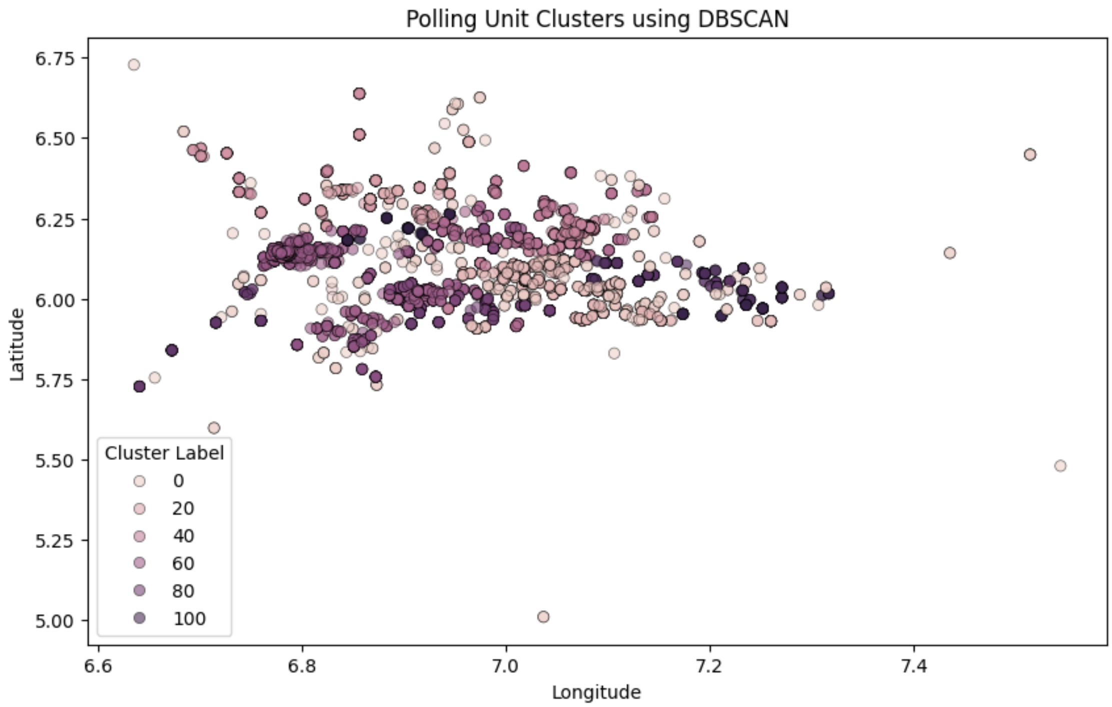
[ ]:
```
#importing the necessary requirement
import geopandas as gpd
from sklearn.cluster import DBSCAN
from haversine import haversine
from geopy.distance import great_circle
from sklearn.preprocessing import StandardScaler
```

[ ]:
```
#loading the geo dataframe
df = pd.read_csv("/content/Geo_Anambra_data.csv")

# Extract latitude and longitude
```

```
coords = df[['latitude', 'longitude']].values

# Standardize features
scaler = StandardScaler()
coords_scaled = scaler.fit_transform(coords)

# Apply DBSCAN for clustering
epsilon = 0.1  # Adjust based on desired proximity (in normalized scale)
min_samples = 5  # Minimum points to form a cluster
clustering = DBSCAN(eps=epsilon, min_samples=min_samples, metric='euclidean').
 ↪fit(coords_scaled)

df['cluster'] = clustering.labels_

# Debugging: Check number of clusters
num_clusters = len(set(clustering.labels_)) - (1 if -1 in clustering.labels_␣
 ↪else 0)
print(f"Identified clusters: {num_clusters}")
print(df['cluster'].value_counts())
```

```
Identified clusters: 116
cluster
 65     671
 38     314
 63     217
 11     213
-1      196
       …
 72       5
 80       5
 74       5
 96       5
 107      5
Name: count, Length: 117, dtype: int64
```

[ ]:
```
# Visualization
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='longitude', y='latitude', hue='cluster',␣
 ↪edgecolor='k', alpha=0.6)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("Polling Unit Clusters using DBSCAN")
plt.legend(title='Cluster Label')
plt.show()
```



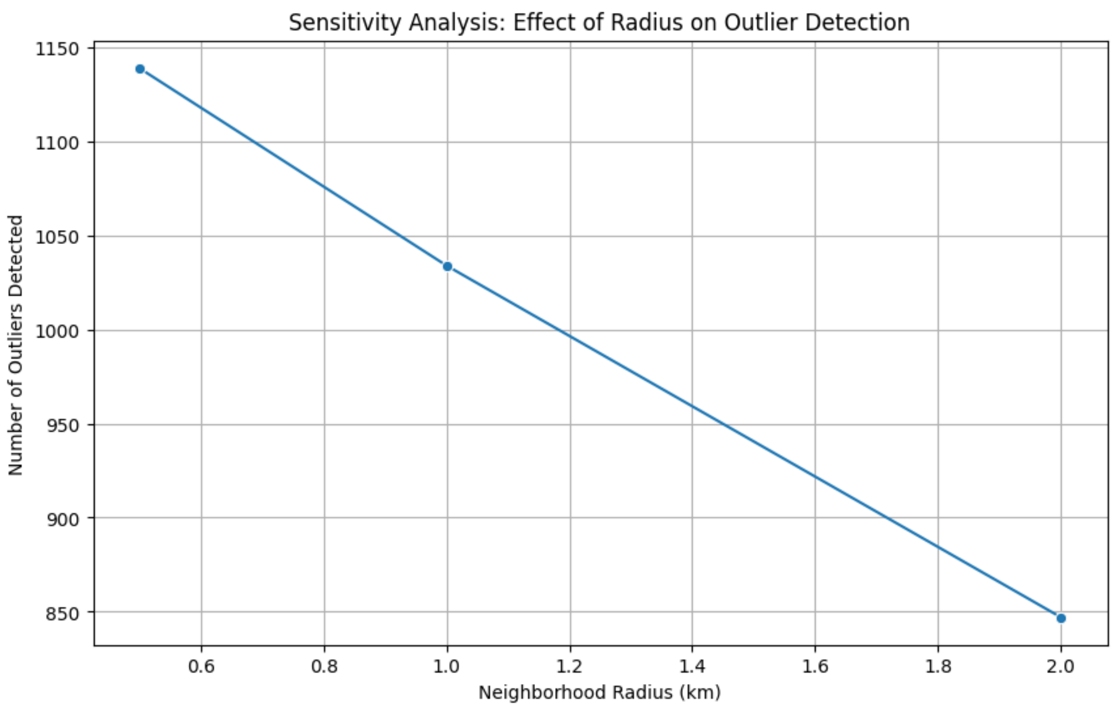
[ ]:
```
df.cluster.nunique()
```

[ ]: 117

[ ]:
```
#sensitivity analysis

# Extract latitude and longitude
coords = df[['latitude', 'longitude']].values

# Standardize features
scaler = StandardScaler()
coords_scaled = scaler.fit_transform(coords)

# Sensitivity Analysis: Varying Neighborhood Radii
radii = [0.005, 0.01, 0.02]  # Approximate scaling for 500m, 1km, 2km
results = {}

for epsilon in radii:
    clustering = DBSCAN(eps=epsilon, min_samples=5, metric='euclidean').
 ↪fit(coords_scaled)
    df[f'cluster_eps_{int(epsilon*100000)}'] = clustering.labels_
    outliers = np.sum(clustering.labels_ == -1)
    results[epsilon] = outliers
    print(f"Epsilon: {epsilon}, Outliers detected: {outliers}")
```

```python
# Plot results using Seaborn for better aesthetics
plt.figure(figsize=(10, 6))
sns.lineplot(x=[epsilon * 100 for epsilon in radii], y=list(results.values()),␣
↪marker='o', linestyle='-')
plt.xlabel("Neighborhood Radius (km)")
plt.ylabel("Number of Outliers Detected")
plt.title("Sensitivity Analysis: Effect of Radius on Outlier Detection")
plt.grid(True)
plt.show()
```

```
Epsilon: 0.005, Outliers detected: 1139
Epsilon: 0.01, Outliers detected: 1034
Epsilon: 0.02, Outliers detected: 847
```



[ ]:
```python
import folium
import numpy as np
import matplotlib.cm as cm
import matplotlib.colors as colors

# Function to generate a Folium Map
def plot_clusters_folium(df, eps_m):
    """
    Plots clustering results using Folium for a given distance threshold.
```

```
    """
    cluster_col = f"cluster_eps_{eps_m}"  # Match actual column names in the dataset

    if cluster_col not in df.columns:
        raise KeyError(f"Column '{cluster_col}' not found in dataframe. Available columns: {df.columns.tolist()}")

    df_clustered = df.copy()

    # Create a base map centered around the dataset
    center_lat, center_lon = df["latitude"].mean(), df["longitude"].mean()
    map_clusters = folium.Map(location=[center_lat, center_lon], zoom_start=10)

    # Define colors for clusters
    unique_clusters = df_clustered[cluster_col].unique()
    colormap = cm.get_cmap('viridis', len(unique_clusters))  # Get color map
    cluster_colors = {c: colors.to_hex(colormap(i / max(len(unique_clusters) - 1, 1))) for i, c in enumerate(unique_clusters)}

    # Plot each polling unit
    for _, row in df_clustered.iterrows():
        cluster_id = row[cluster_col]
        color = cluster_colors.get(cluster_id, "#000000")  # Black for noise

        folium.CircleMarker(
            location=[row["latitude"], row["longitude"]],
            radius=3,
            color=color,
            fill=True,
            fill_color=color,
            fill_opacity=0.6,
            popup=f"PU: {row['pu-name']} | Cluster: {cluster_id}",
        ).add_to(map_clusters)

    return map_clusters

# Generate maps for different radii
map_500m = plot_clusters_folium(df, 500)
map_1000m = plot_clusters_folium(df, 1000)
map_2000m = plot_clusters_folium(df, 2000)

# Save maps
map_500m.save("polling_units_500m.html")
map_1000m.save("polling_units_1000m.html")
map_2000m.save("polling_units_2000m.html")
```

```
print("Maps saved successfully!")
```

```
<ipython-input-99-8cb67c952234>:24: MatplotlibDeprecationWarning: The get_cmap
function was deprecated in Matplotlib 3.7 and will be removed in 3.11. Use
``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()`` or
``pyplot.get_cmap()`` instead.
  colormap = cm.get_cmap('viridis', len(unique_clusters))  # Get color map
```

```
Maps saved successfully!
```

[190]:
```
map_500m
```

[190]: <folium.folium.Map at 0x7c352af58b90>

[191]:
```
map_1000m
```

[191]: <folium.folium.Map at 0x7c352a08db10>

[192]:
```
map_2000m
```

[192]: <folium.folium.Map at 0x7c3529375b10>

0.1.1 Sophisticated Outlier Score Calculation

[ ]:
```
pip install geopandas libpysal esda
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.11/dist-
packages (1.0.1)
Collecting libpysal
  Downloading libpysal-4.13.0-py3-none-any.whl.metadata (4.8 kB)
Collecting esda
  Downloading esda-2.7.0-py3-none-any.whl.metadata (2.0 kB)
Requirement already satisfied: numpy>=1.22 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (2.0.2)
Requirement already satisfied: pyogrio>=0.7.2 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (0.10.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-
packages (from geopandas) (24.2)
Requirement already satisfied: pandas>=1.4.0 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (2.2.2)
Requirement already satisfied: pyproj>=3.3.0 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (3.7.1)
Requirement already satisfied: shapely>=2.0.0 in /usr/local/lib/python3.11/dist-
packages (from geopandas) (2.0.7)
Requirement already satisfied: beautifulsoup4>=4.10 in
/usr/local/lib/python3.11/dist-packages (from libpysal) (4.13.3)
Requirement already satisfied: platformdirs>=2.0.2 in
/usr/local/lib/python3.11/dist-packages (from libpysal) (4.3.7)
```

```
Requirement already satisfied: requests>=2.27 in /usr/local/lib/python3.11/dist-
packages (from libpysal) (2.32.3)
Requirement already satisfied: scipy>=1.8 in /usr/local/lib/python3.11/dist-
packages (from libpysal) (1.14.1)
Requirement already satisfied: scikit-learn>=1.1 in
/usr/local/lib/python3.11/dist-packages (from libpysal) (1.6.1)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.11/dist-
packages (from beautifulsoup4>=4.10->libpysal) (2.6)
Requirement already satisfied: typing-extensions>=4.0.0 in
/usr/local/lib/python3.11/dist-packages (from beautifulsoup4>=4.10->libpysal)
(4.12.2)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas>=1.4.0->geopandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas>=1.4.0->geopandas) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas>=1.4.0->geopandas) (2025.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-
packages (from pyogrio>=0.7.2->geopandas) (2025.1.31)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.27->libpysal) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests>=2.27->libpysal) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests>=2.27->libpysal) (2.3.0)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn>=1.1->libpysal) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.1->libpysal)
(3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas>=1.4.0->geopandas) (1.17.0)
Downloading libpysal-4.13.0-py3-none-any.whl (2.8 MB)
                         2.8/2.8 MB
19.4 MB/s eta 0:00:00
Downloading esda-2.7.0-py3-none-any.whl (142 kB)
                         142.8/142.8 kB
13.0 MB/s eta 0:00:00
Installing collected packages: libpysal, esda
Successfully installed esda-2.7.0 libpysal-4.13.0
```

[ ]:
```python
import pandas as pd
import geopandas as gpd
import numpy as np
import libpysal as ps
import esda
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from shapely.geometry import Point
```

Local Moran's I to identify localized spatial autocorrelation.

[159]:
```
# Convert DataFrame to GeoDataFrame
geometry = [Point(xy) for xy in zip(df.longitude, df.latitude)]
gdf = gpd.GeoDataFrame(df, geometry=geometry, crs="EPSG:4326")

# Use a key numerical column for analysis (e.g., total votes or a party's votes)
variable = "accredited_voters"  # Change to apc, pdp, lp, etc., as needed

# Standardize the variable (Z-score normalization)
gdf["z_score"] = (gdf[variable] - gdf[variable].mean()) / gdf[variable].std()

# Generate spatial weights matrix (K-nearest neighbors)
w = ps.weights.KNN.from_dataframe(gdf, k=6)
w.transform = "r"

# Compute Local Moran's I
mi = esda.moran.Moran_Local(gdf["z_score"], w)

# Store results
gdf["local_moran"] = mi.Is  # Moran's I score
gdf["p_value"] = mi.p_sim    # Significance of spatial autocorrelation

# Classify spatial clusters and outliers
gdf["cluster"] = np.select(
    [
        (mi.q == 1) & (mi.p_sim < 0.05),  # High-High (hotspot)
        (mi.q == 3) & (mi.p_sim < 0.05),  # Low-Low (coldspot)
        (mi.q == 2) & (mi.p_sim < 0.05),  # High-Low (outlier)
        (mi.q == 4) & (mi.p_sim < 0.05),  # Low-High (outlier)
    ],
    ["High-High", "Low-Low", "High-Low", "Low-High"],
    default="Not Significant",
)

# Save results
gdf.to_csv("outlier_scores.csv", index=False)
```

```
/usr/local/lib/python3.11/dist-packages/libpysal/weights/distance.py:153:
UserWarning: The weights matrix is not fully connected:
 There are 129 disconnected components.
  W.__init__(self, neighbors, id_order=ids, **kwargs)
```

[150]:
```
gdf.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 29 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3679 non-null   object
 1   lga                     3679 non-null   object
 2   ward                    3679 non-null   object
 3   pu-code                 3679 non-null   object
 4   pu-name                 3679 non-null   object
 5   accredited_voters       3679 non-null   int64
 6   registered_voters       3679 non-null   int64
 7   results_found           3679 non-null   bool
 8   transcription_count     3679 non-null   int64
 9   result_sheet_stamped    3679 non-null   bool
 10  result_sheet_corrected  3679 non-null   bool
 11  result_sheet_invalid    3679 non-null   bool
 12  result_sheet_unclear    3679 non-null   bool
 13  result_sheet_unsigned   3679 non-null   object
 14  apc                     3679 non-null   int64
 15  lp                      3679 non-null   int64
 16  pdp                     3679 non-null   int64
 17  nnpp                    3679 non-null   int64
 18  results_file            3679 non-null   object
 19  latitude                3679 non-null   float64
 20  longitude               3679 non-null   float64
 21  cluster                 3679 non-null   object
 22  cluster_eps_500         3679 non-null   int64
 23  cluster_eps_1000        3679 non-null   int64
 24  cluster_eps_2000        3679 non-null   int64
 25  geometry                3679 non-null   geometry
 26  z_score                 3679 non-null   float64
 27  local_moran             3679 non-null   float64
 28  p_value                 3679 non-null   float64
dtypes: bool(5), float64(5), geometry(1), int64(10), object(8)
memory usage: 707.9+ KB
```

[151]:
```
gdf.describe()
```

[151]:
```
       accredited_voters  registered_voters  transcription_count          apc  \
count        3679.000000        3679.000000               3679.0  3679.000000
mean          115.236477         450.597445                 -1.0     1.260669
std            79.122946         333.768034                  0.0     7.910370
min             0.000000           1.000000                 -1.0     0.000000
25%            56.000000         203.000000                 -1.0     0.000000
50%           103.000000         397.000000                 -1.0     0.000000
75%           159.000000         640.500000                 -1.0     1.000000
```

```
max           582.000000        3770.000000                  -1.0   350.000000

                 lp          pdp         nnpp     latitude    longitude  \
count   3679.000000  3679.000000  3679.000000  3679.000000  3679.000000
mean     103.258766     2.413971     0.556401     6.122716     6.951253
std       77.716562    11.511426     5.703382     0.144069     0.135774
min        0.000000     0.000000     0.000000     5.010543     6.635640
25%       45.000000     0.000000     0.000000     6.024489     6.835383
50%       91.000000     1.000000     0.000000     6.133923     6.947753
75%      144.500000     2.000000     0.000000     6.206922     7.057911
max      574.000000   465.000000   251.000000     6.727458     7.543680

       cluster_eps_500  cluster_eps_1000  cluster_eps_2000       z_score  \
count      3679.000000       3679.000000       3679.000000  3.679000e+03
mean         73.675455         78.837456         84.632509 -3.090156e-17
std          72.952513         74.741619         71.492200  1.000000e+00
min          -1.000000         -1.000000         -1.000000 -1.456423e+00
25%          -1.000000         -1.000000          5.000000 -7.486637e-01
50%          55.000000         63.000000         74.000000 -1.546514e-01
75%         135.000000        140.500000        156.000000  5.531079e-01
max         225.000000        230.000000        215.000000  5.899218e+00

       local_moran      p_value
count  3679.000000  3679.000000
mean      0.161142     0.193824
std       0.710832     0.151367
min      -2.640027     0.001000
25%      -0.068766     0.059000
50%       0.056167     0.164000
75%       0.342204     0.321000
max      12.477042     0.500000
```

[152]:
```
gdf.head()
```

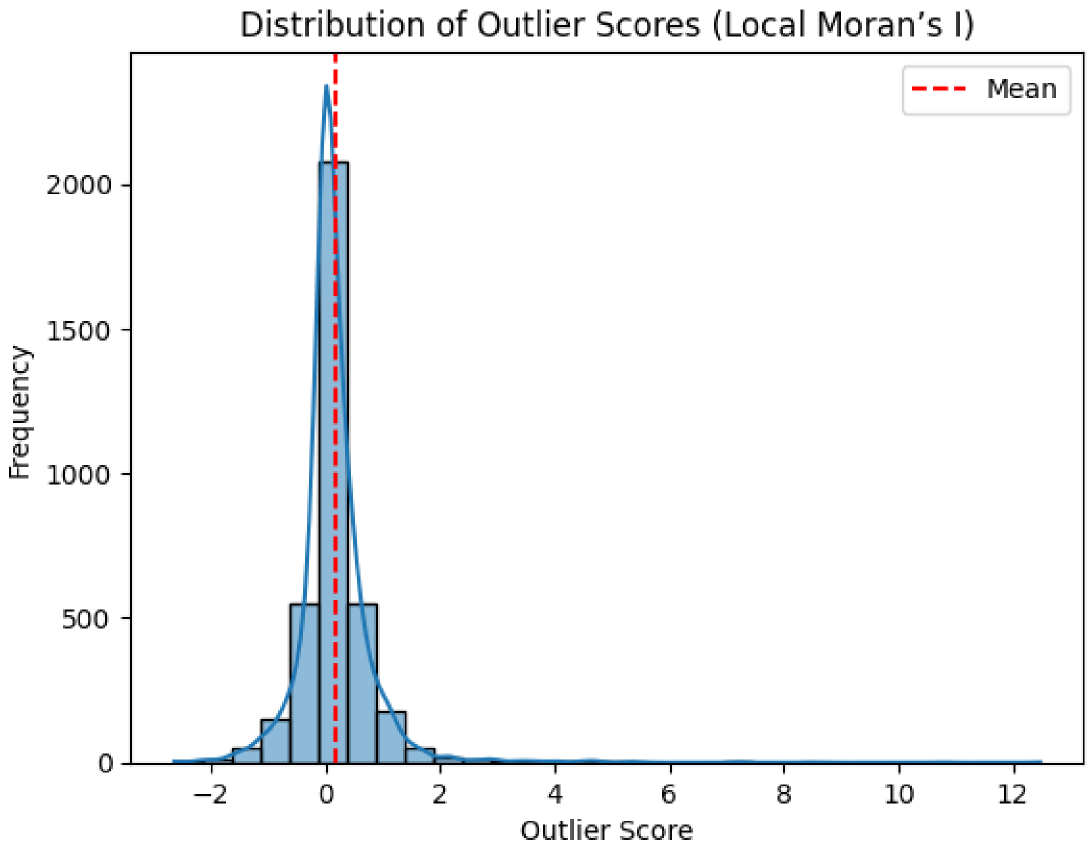
[152]:
```
     state     lga       ward       pu-code              pu-name  \
0  ANAMBRA  AGUATA  ACHINA   I  04-01-01-001  ST. CHARLE'S SCHOOL
1  ANAMBRA  AGUATA  ACHINA   I  04-01-01-005       AMANKWU SQUARE
2  ANAMBRA  AGUATA  ACHINA   I  04-01-01-006     COOPERATIVE HALL
3  ANAMBRA  AGUATA  ACHINA   I  04-01-01-008      OCHIEOBU SQUARE
4  ANAMBRA  AGUATA  ACHINA   I  04-01-01-010   OYE MOTOR PARK  II

   accredited_voters  registered_voters  results_found  transcription_count  \
0                171                630           True                   -1
1                153                500           True                   -1
2                121                386           True                   -1
3                134                426           True                   -1
4                 63                166           True                   -1
```

```
   result_sheet_stamped  …  latitude  longitude          cluster  \
0                 False  …  5.975354   7.131900  Not Significant
1                 False  …  5.975354   7.131900  Not Significant
2                 False  …  5.960727   7.118174  Not Significant
3                 False  …  5.975354   7.131900  Not Significant
4                 False  …  5.965620   7.119106  Not Significant

  cluster_eps_500  cluster_eps_1000  cluster_eps_2000  \
0              -1                -1                -1
1              -1                -1                -1
2              -1                -1                -1
3              -1                -1                -1
4              -1                -1                -1

                 geometry   z_score local_moran  p_value
0   POINT (7.1319 5.97535)  0.704771   -0.100086    0.351
1   POINT (7.1319 5.97535)  0.477277   -0.049683    0.400
2  POINT (7.11817 5.96073)  0.072843   -0.024154    0.190
3   POINT (7.1319 5.97535)  0.237144   -0.015195    0.453
4  POINT (7.11911 5.96562) -0.660194    0.138257    0.288

[5 rows x 29 columns]
```

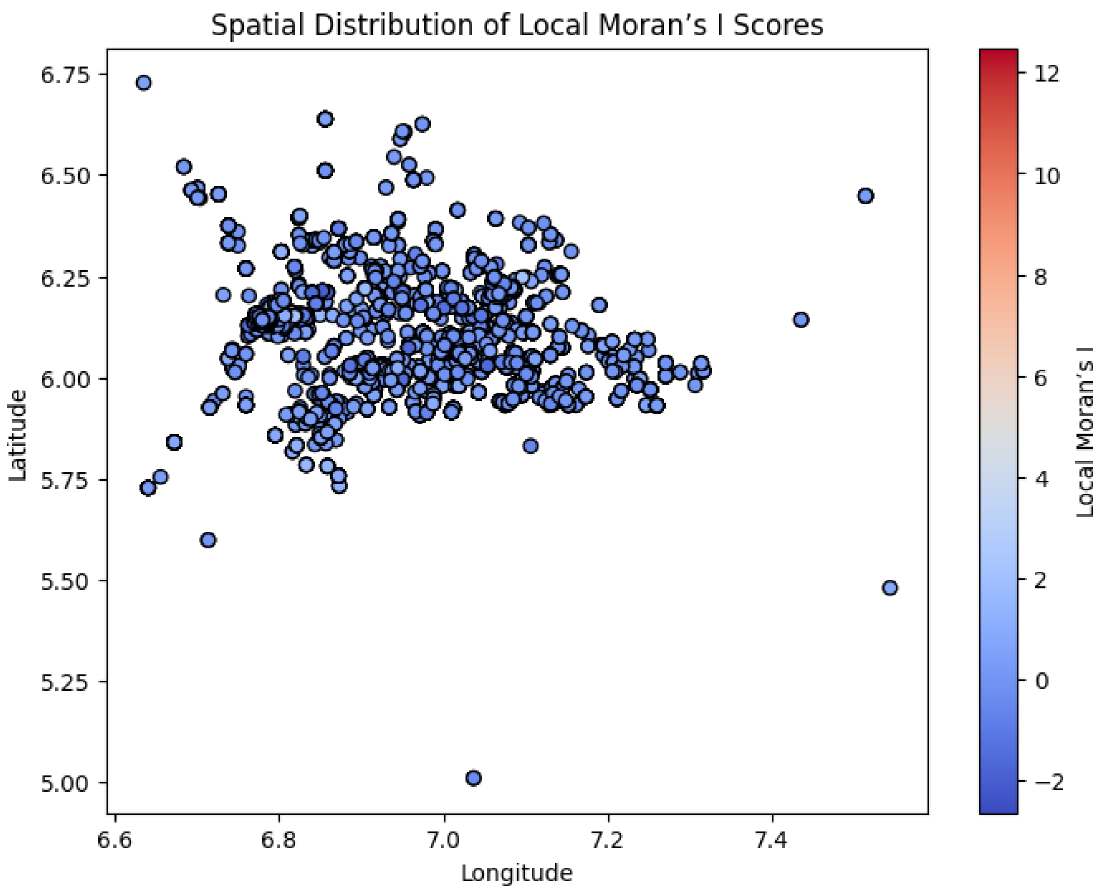
[153]:
```
gdf.shape
```

[153]:
```
(3679, 29)
```

[154]:
```
#Histogram of Outlier Scores
sns.histplot(gdf["local_moran"], bins=30, kde=True)
plt.axvline(x=gdf["local_moran"].mean(), color="red", linestyle="--",␣
 ↪label="Mean")
plt.title("Distribution of Outlier Scores (Local Moran's I)")
plt.xlabel("Outlier Score")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```




```
[ ]: #Scatter Plot of Spatial Autocorrelation
plt.figure(figsize=(8,6))
plt.scatter(gdf["longitude"], gdf["latitude"], c=gdf["local_moran"],␣
  ↪cmap="coolwarm", edgecolors="black")
plt.colorbar(label="Local Moran's I")
plt.title("Spatial Distribution of Local Moran's I Scores")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.show()
```



[188]:
```python
#Interactive Map with Outlier Classification
import folium
from folium.plugins import MarkerCluster

# Define cluster colors
cluster_colors = {
    "High-High": "red",
    "Low-Low": "blue",
    "High-Low": "purple",
    "Low-High": "orange",
    "Not Significant": "gray"
}

m = folium.Map(location=[gdf.latitude.mean(), gdf.longitude.mean()],␣
 ↪zoom_start=10)
marker_cluster = MarkerCluster().add_to(m)

for _, row in gdf.iterrows():
    folium.CircleMarker(
```

```
        location=[row["latitude"], row["longitude"]],
        radius=6,
        color=cluster_colors[row["cluster"]],
        fill=True,
        fill_color=cluster_colors[row["cluster"]],
        fill_opacity=0.7,
        popup=f"Polling Unit: {row['pu-name']}<br>Cluster:␣
↪{row['cluster']}<br>Outlier Score: {row['local_moran']:.3f}",
    ).add_to(marker_cluster)

m.save("outlier_map.html")
```

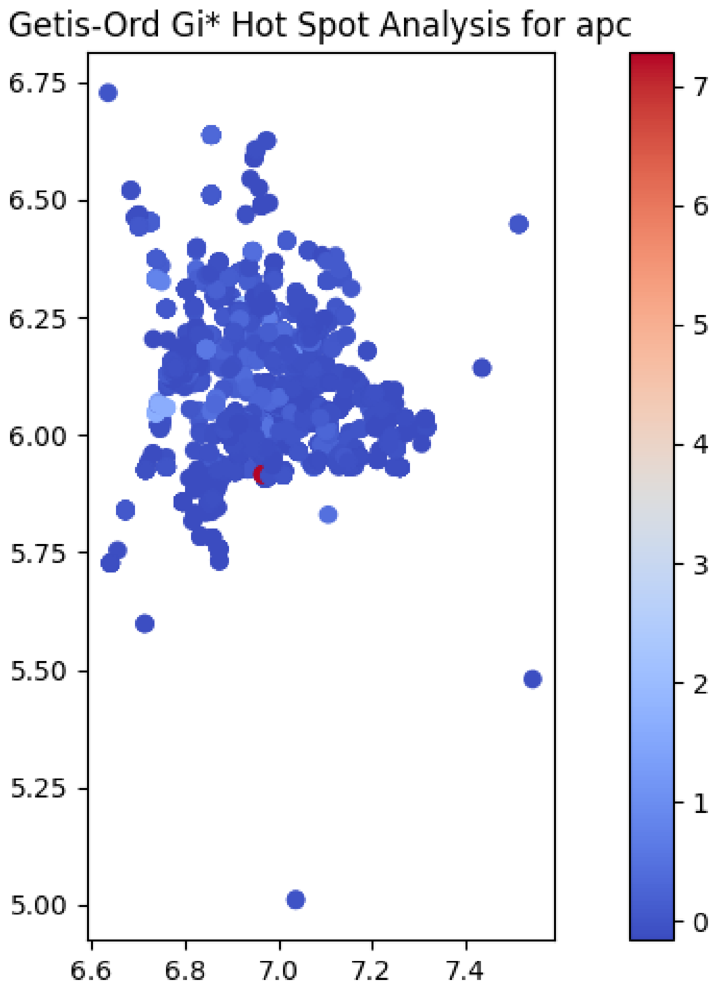
[189]:
```
m
```

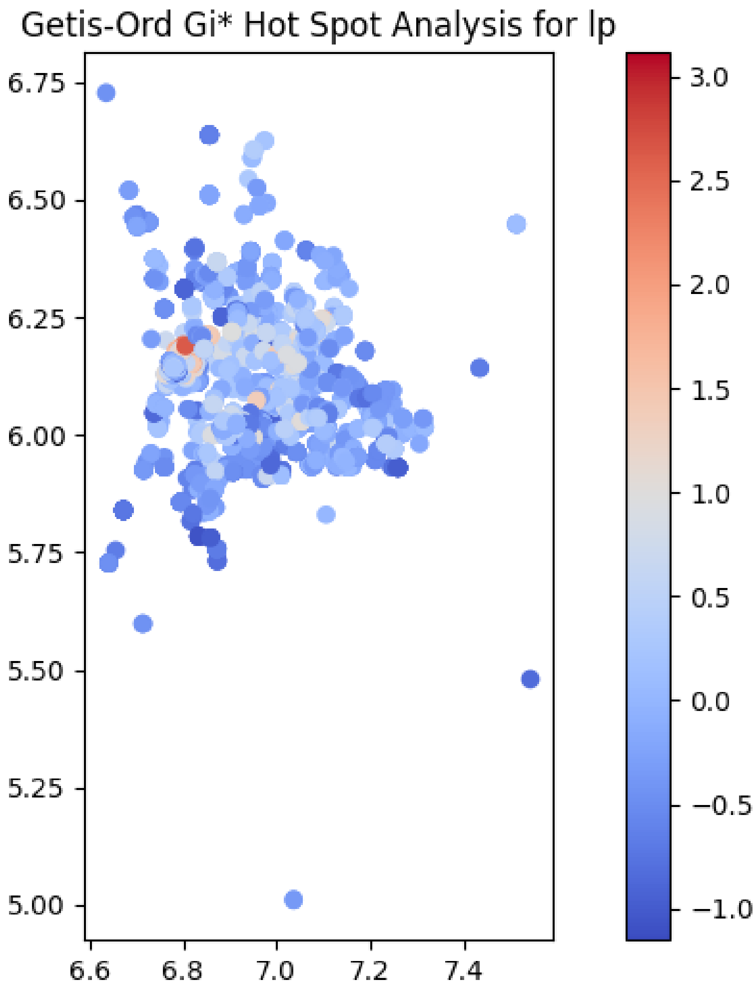
[189]: <folium.folium.Map at 0x7c34fad1f450>

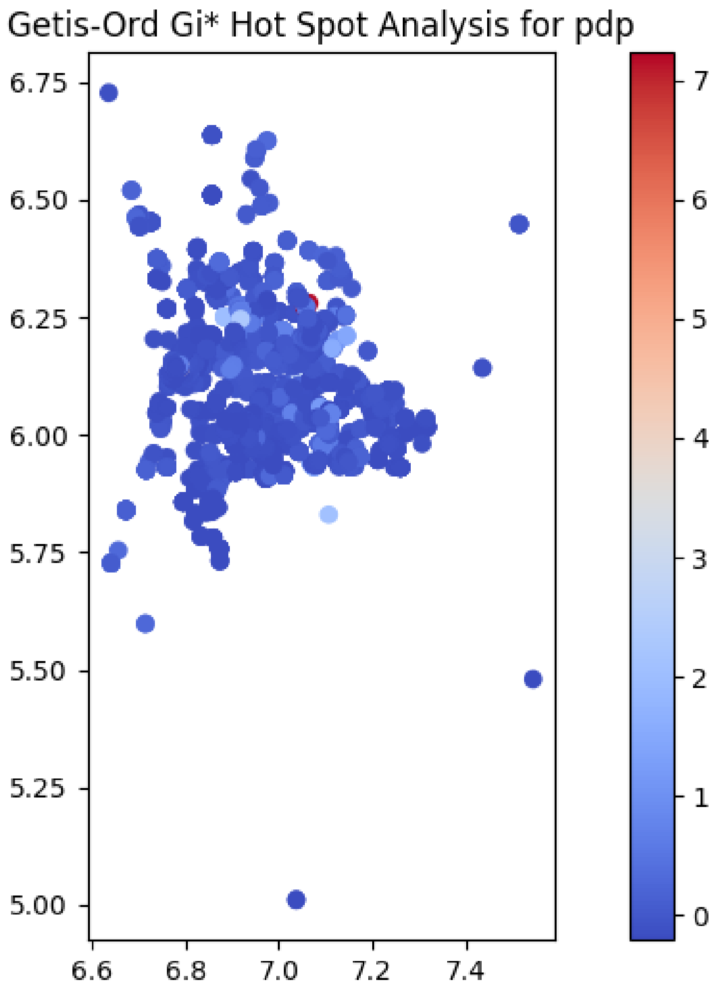
Getis-Ord Gi(Hot Spot Analysis) to detect significant vote concentration.

[123]:
```
import geopandas as gpd
import numpy as np
import libpysal as lps
from esda.getisord import G_Local
import matplotlib.pyplot as plt
```

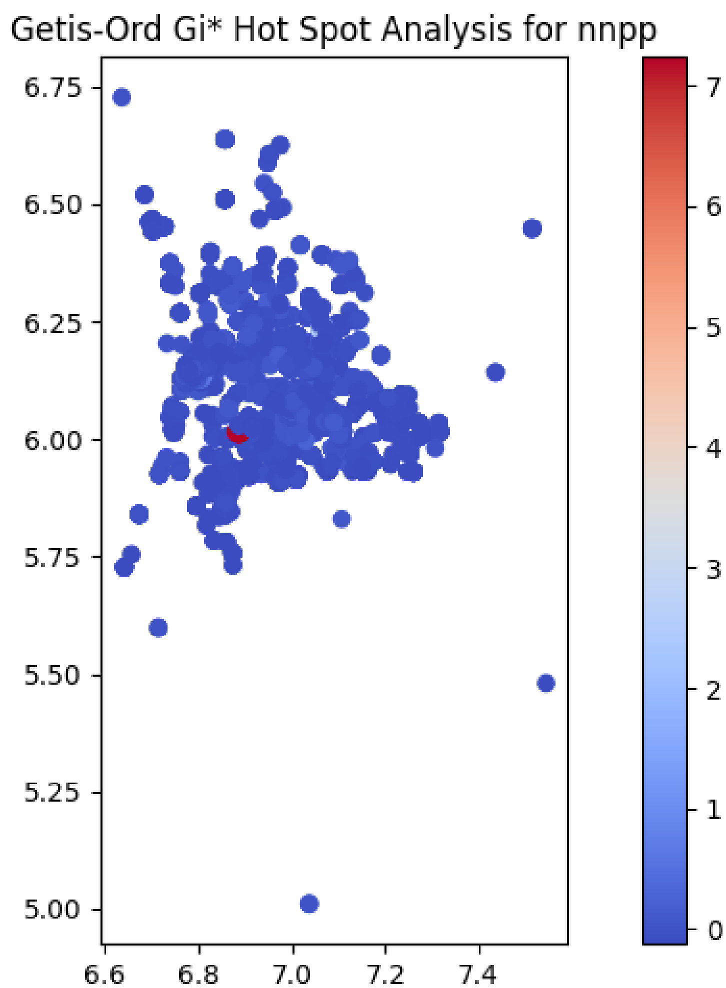
[160]:
```
vote_columns = ["apc", "lp", "pdp", "nnpp"]

for vote_column in vote_columns:
    gdf[vote_column] = gdf[vote_column].astype(float)  # Ensure numeric dtype
    gi = G_Local(gdf[vote_column], w)

    gdf[f"Gi_star_{vote_column}"] = gi.Zs
    gdf[f"Gi_p_value_{vote_column}"] = gi.p_sim
    gdf[f"hotspot_{vote_column}"] = gdf[f"Gi_p_value_{vote_column}"] < 0.05

    # Plot the result
    fig, ax = plt.subplots(figsize=(10, 6))
    gdf.plot(column=f"Gi_star_{vote_column}", cmap="coolwarm", legend=True,␣
↪ax=ax)
    plt.title(f"Getis-Ord Gi* Hot Spot Analysis for {vote_column}")
    plt.show()
```


Getis-Ord Gi* Hot Spot Analysis for apc
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
6.6
6.8
7.0
7.2
7.4
7
6
5
4
3
2
1
0


Getis-Ord Gi* Hot Spot Analysis for lp
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
6.6
6.8
7.0
7.2
7.4
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0



Getis-Ord Gi* Hot Spot Analysis for pdp
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
6.6
6.8
7.0
7.2
7.4
7
6
5
4
3
2
1
0



[161]:
```
# Compute Gi* values and aggregate across parties
gdf["Gi_outlier_score"] = (
    gdf["Gi_star_apc"] +
    gdf["Gi_star_lp"] +
    gdf["Gi_star_pdp"] +
    gdf["Gi_star_nnpp"]
)
```

[162]:
```
gdf.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 42 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   state                    3679 non-null   object
```

```
 1   lga                      3679 non-null   object
 2   ward                     3679 non-null   object
 3   pu-code                  3679 non-null   object
 4   pu-name                  3679 non-null   object
 5   accredited_voters        3679 non-null   int64
 6   registered_voters        3679 non-null   int64
 7   results_found            3679 non-null   bool
 8   transcription_count      3679 non-null   int64
 9   result_sheet_stamped     3679 non-null   bool
 10  result_sheet_corrected   3679 non-null   bool
 11  result_sheet_invalid     3679 non-null   bool
 12  result_sheet_unclear     3679 non-null   bool
 13  result_sheet_unsigned    3679 non-null   object
 14  apc                      3679 non-null   float64
 15  lp                       3679 non-null   float64
 16  pdp                      3679 non-null   float64
 17  nnpp                     3679 non-null   float64
 18  results_file             3679 non-null   object
 19  latitude                 3679 non-null   float64
 20  longitude                3679 non-null   float64
 21  cluster                  3679 non-null   object
 22  cluster_eps_500          3679 non-null   int64
 23  cluster_eps_1000         3679 non-null   int64
 24  cluster_eps_2000         3679 non-null   int64
 25  geometry                 3679 non-null   geometry
 26  z_score                  3679 non-null   float64
 27  local_moran              3679 non-null   float64
 28  p_value                  3679 non-null   float64
 29  Gi_star_apc              3679 non-null   float64
 30  Gi_p_value_apc           3679 non-null   float64
 31  hotspot_apc              3679 non-null   bool
 32  Gi_star_lp               3679 non-null   float64
 33  Gi_p_value_lp            3679 non-null   float64
 34  hotspot_lp               3679 non-null   bool
 35  Gi_star_pdp              3679 non-null   float64
 36  Gi_p_value_pdp           3679 non-null   float64
 37  hotspot_pdp              3679 non-null   bool
 38  Gi_star_nnpp             3679 non-null   float64
 39  Gi_p_value_nnpp          3679 non-null   float64
 40  hotspot_nnpp             3679 non-null   bool
 41  Gi_outlier_score         3679 non-null   float64
dtypes: bool(9), float64(18), geometry(1), int64(6), object(8)
memory usage: 981.0+ KB
```

[163]:

```
gdf.describe()
```

[163]:

	accredited_voters	registered_voters	transcription_count	apc	\
count	3679.000000	3679.000000	3679.0	3679.000000	
mean	115.236477	450.597445	-1.0	1.260669	
std	79.122946	333.768034	0.0	7.910370	
min	0.000000	1.000000	-1.0	0.000000	
25%	56.000000	203.000000	-1.0	0.000000	
50%	103.000000	397.000000	-1.0	0.000000	
75%	159.000000	640.500000	-1.0	1.000000	
max	582.000000	3770.000000	-1.0	350.000000	

	lp	pdp	nnpp	latitude	longitude	\
count	3679.000000	3679.000000	3679.000000	3679.000000	3679.000000	
mean	103.258766	2.413971	0.556401	6.122716	6.951253	
std	77.716562	11.511426	5.703382	0.144069	0.135774	
min	0.000000	0.000000	0.000000	5.010543	6.635640	
25%	45.000000	0.000000	0.000000	6.024489	6.835383	
50%	91.000000	1.000000	0.000000	6.133923	6.947753	
75%	144.500000	2.000000	0.000000	6.206922	7.057911	
max	574.000000	465.000000	251.000000	6.727458	7.543680	

	cluster_eps_500	…	p_value	Gi_star_apc	Gi_p_value_apc	\
count	3679.000000	…	3679.000000	3679.000000	3679.000000	
mean	73.675455	…	0.198497	-0.017896	0.187905	
std	72.952513	…	0.152308	0.259182	0.152642	
min	-1.000000	…	0.001000	-0.159413	0.001000	
25%	-1.000000	…	0.059000	-0.117274	0.050000	
50%	55.000000	…	0.170000	-0.075135	0.143000	
75%	135.000000	…	0.325000	0.009143	0.284000	
max	225.000000	…	0.500000	7.278107	0.500000	

	Gi_star_lp	Gi_p_value_lp	Gi_star_pdp	Gi_p_value_pdp	Gi_star_nnpp	\
count	3679.000000	3679.000000	3679.000000	3679.000000	3679.000000	
mean	-0.008287	0.208504	0.003776	0.189745	0.013586	
std	0.529294	0.152278	0.480842	0.154573	0.513088	
min	-1.152793	0.001000	-0.209760	0.001000	-0.124148	
25%	-0.372325	0.063000	-0.166301	0.045500	-0.097488	
50%	-0.056719	0.191000	-0.093933	0.147000	-0.068266	
75%	0.307682	0.337000	0.007416	0.329000	-0.009916	
max	3.113074	0.500000	7.232171	0.500000	7.237301	

	Gi_p_value_nnpp	Gi_outlier_score
count	3679.000000	3679.000000
mean	0.192821	-0.008820
std	0.164227	0.964679
min	0.001000	-1.619501
25%	0.001000	-0.583804
50%	0.216000	-0.132854

```
75%          0.336000          0.344025
max          0.481000          7.426682

[8 rows x 24 columns]
```

[164]:
```
gdf.head()
```

[164]:
```
     state     lga        ward       pu-code              pu-name  \
0  ANAMBRA  AGUATA  ACHINA   I  04-01-01-001  ST. CHARLE'S SCHOOL
1  ANAMBRA  AGUATA  ACHINA   I  04-01-01-005       AMANKWU SQUARE
2  ANAMBRA  AGUATA  ACHINA   I  04-01-01-006     COOPERATIVE HALL
3  ANAMBRA  AGUATA  ACHINA   I  04-01-01-008      OCHIEOBU SQUARE
4  ANAMBRA  AGUATA  ACHINA   I  04-01-01-010   OYE MOTOR PARK   II

   accredited_voters  registered_voters  results_found  transcription_count  \
0                171                630           True                   -1
1                153                500           True                   -1
2                121                386           True                   -1
3                134                426           True                   -1
4                 63                166           True                   -1

   result_sheet_stamped  …  Gi_star_lp  Gi_p_value_lp  hotspot_lp  \
0                 False  …   -0.104510          0.442       False
1                 False  …   -0.408527          0.161       False
2                 False  …   -0.286840          0.253       False
3                 False  …   -0.369978          0.186       False
4                 False  …   -0.408830          0.160       False

  Gi_star_pdp  Gi_p_value_pdp  hotspot_pdp  Gi_star_nnpp  Gi_p_value_nnpp  \
0   -0.064976           0.453        False      0.077752            0.064
1   -0.064976           0.453        False      0.048577            0.119
2   -0.108411           0.409        False     -0.009916            0.313
3   -0.122795           0.291        False      0.048577            0.119
4   -0.108411           0.408        False     -0.009916            0.313

  hotspot_nnpp  Gi_outlier_score
0        False         -0.124730
1        False         -0.521028
2        False         -0.459232
3        False         -0.477192
4        False         -0.665365

[5 rows x 42 columns]
```

Integrate additional anomaly detection methods

[165]:
```
gdf.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 42 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   state                   3679 non-null   object
 1   lga                     3679 non-null   object
 2   ward                    3679 non-null   object
 3   pu-code                 3679 non-null   object
 4   pu-name                 3679 non-null   object
 5   accredited_voters       3679 non-null   int64
 6   registered_voters       3679 non-null   int64
 7   results_found           3679 non-null   bool
 8   transcription_count     3679 non-null   int64
 9   result_sheet_stamped    3679 non-null   bool
 10  result_sheet_corrected  3679 non-null   bool
 11  result_sheet_invalid    3679 non-null   bool
 12  result_sheet_unclear    3679 non-null   bool
 13  result_sheet_unsigned   3679 non-null   object
 14  apc                     3679 non-null   float64
 15  lp                      3679 non-null   float64
 16  pdp                     3679 non-null   float64
 17  nnpp                    3679 non-null   float64
 18  results_file            3679 non-null   object
 19  latitude                3679 non-null   float64
 20  longitude               3679 non-null   float64
 21  cluster                 3679 non-null   object
 22  cluster_eps_500         3679 non-null   int64
 23  cluster_eps_1000        3679 non-null   int64
 24  cluster_eps_2000        3679 non-null   int64
 25  geometry                3679 non-null   geometry
 26  z_score                 3679 non-null   float64
 27  local_moran             3679 non-null   float64
 28  p_value                 3679 non-null   float64
 29  Gi_star_apc             3679 non-null   float64
 30  Gi_p_value_apc          3679 non-null   float64
 31  hotspot_apc             3679 non-null   bool
 32  Gi_star_lp              3679 non-null   float64
 33  Gi_p_value_lp           3679 non-null   float64
 34  hotspot_lp              3679 non-null   bool
 35  Gi_star_pdp             3679 non-null   float64
 36  Gi_p_value_pdp          3679 non-null   float64
 37  hotspot_pdp             3679 non-null   bool
 38  Gi_star_nnpp            3679 non-null   float64
 39  Gi_p_value_nnpp         3679 non-null   float64
 40  hotspot_nnpp            3679 non-null   bool
 41  Gi_outlier_score        3679 non-null   float64
dtypes: bool(9), float64(18), geometry(1), int64(6), object(8)
```

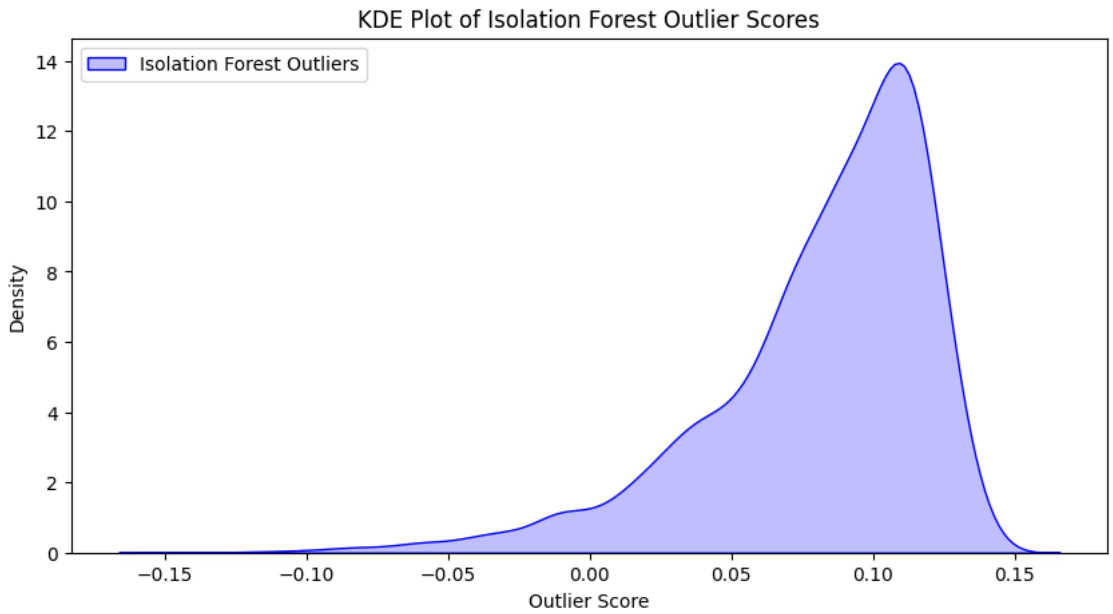
memory usage: 981.0+ KB

[166]:
```python
from sklearn.ensemble import IsolationForest
import numpy as np

# Select relevant numerical features
features = ["latitude", "longitude", "apc", "lp", "pdp", "nnpp",
            "cluster_eps_500", "cluster_eps_1000", "cluster_eps_2000",
            "local_moran", "Gi_outlier_score"]

# Convert to NumPy array and handle missing values
X = gdf[features].fillna(0).values

# Initialize and fit Isolation Forest
iso_forest = IsolationForest(n_estimators=100, contamination=0.05,␣
 ↪random_state=42)
iso_forest.fit(X)  # Fit the model

# Compute anomaly scores (higher = normal, lower = outlier)
gdf["iso_forest_score"] = iso_forest.decision_function(X)

# Display results
gdf[["pu-code", "iso_forest_score"]].head()
```

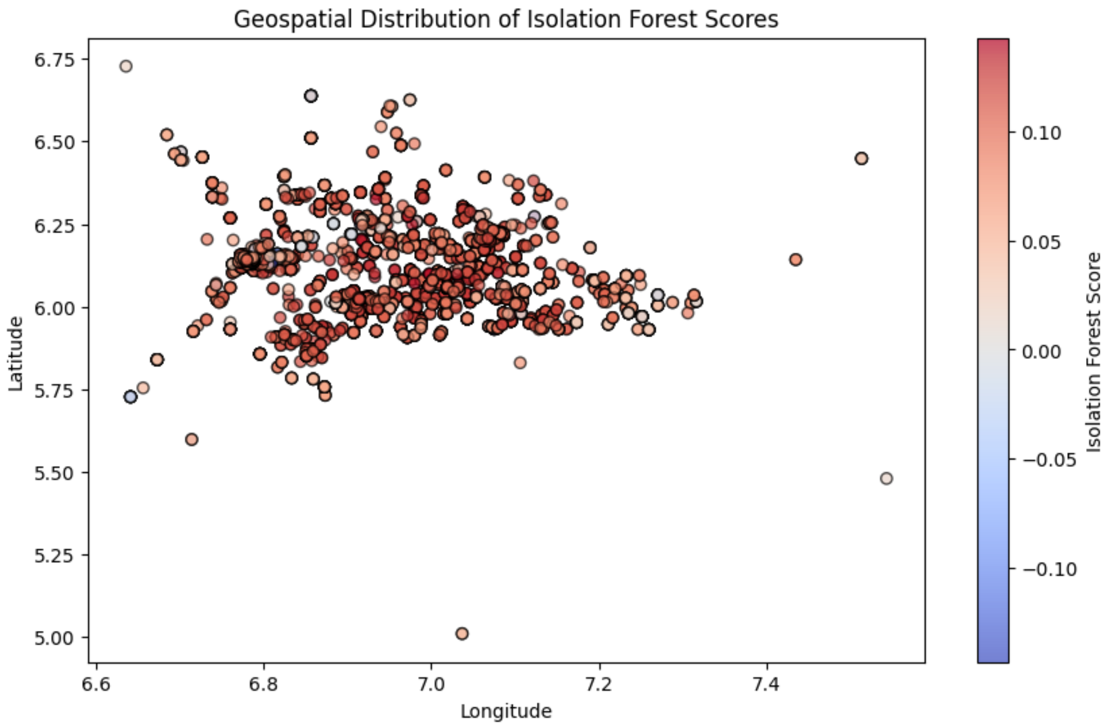
[166]:

	pu-code	iso_forest_score
0	04-01-01-001	0.112147
1	04-01-01-005	0.097351
2	04-01-01-006	0.118558
3	04-01-01-008	0.105581
4	04-01-01-010	0.107372

[167]:
```python
plt.figure(figsize=(10, 5))

# KDE Plot for Absolute Outlier Scores
sns.kdeplot(gdf["iso_forest_score"], fill=True, color="blue", label="Isolation␣
 ↪Forest Outliers")

plt.title("KDE Plot of Isolation Forest Outlier Scores")
plt.xlabel("Outlier Score")
plt.ylabel("Density")
plt.legend()
plt.show()
```



[169]:
```
# Scatter Plot (Latitude vs. Longitude) with Outlier Scores
plt.figure(figsize=(10, 6))
sc = plt.scatter(gdf["longitude"], gdf["latitude"], c=gdf["iso_forest_score"],␣
 ↪cmap="coolwarm", edgecolors="k", alpha=0.7)
plt.colorbar(sc, label="Isolation Forest Score")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("Geospatial Distribution of Isolation Forest Scores")
plt.show()
```

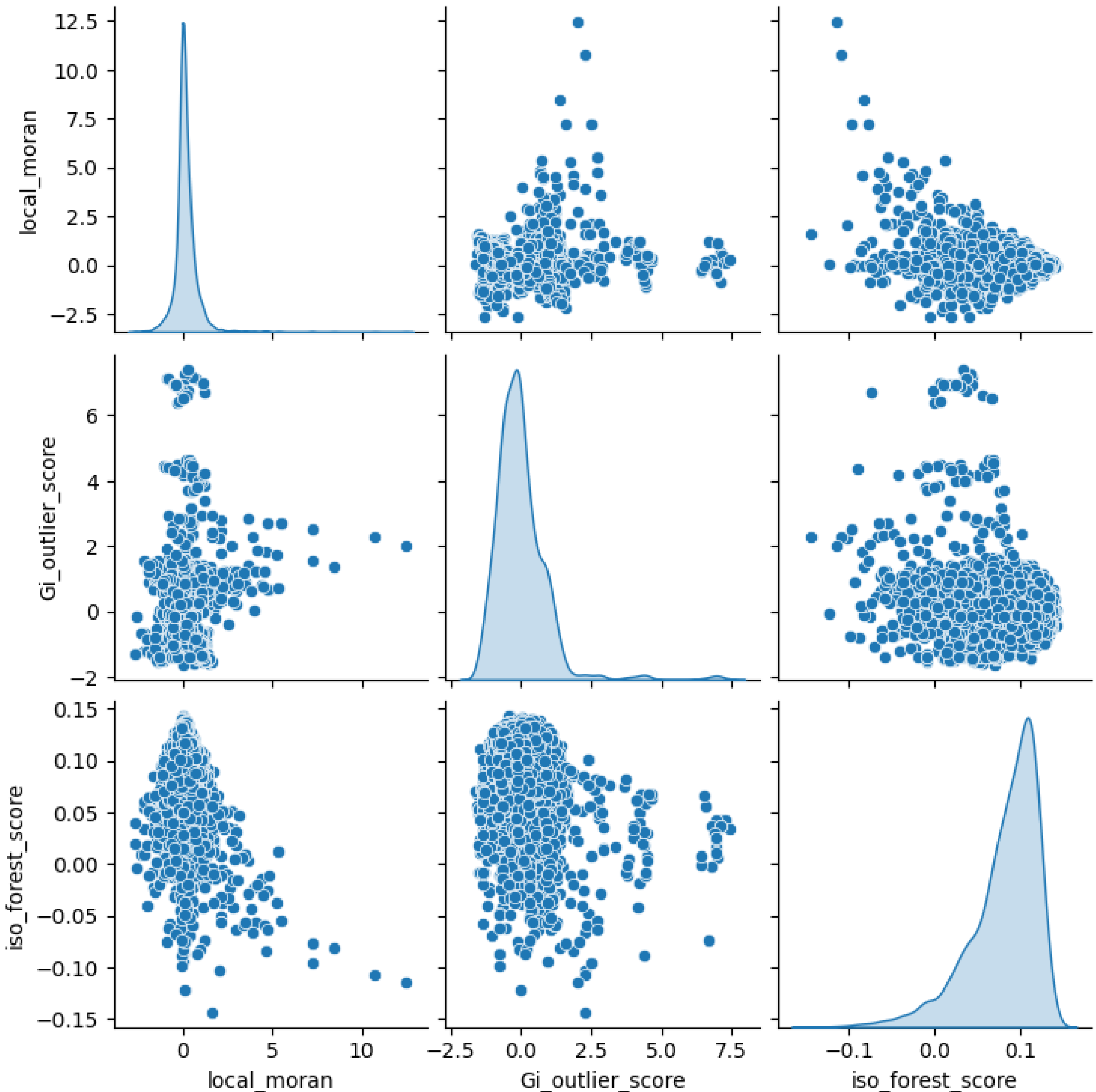


[170]:
```
import folium
from folium.plugins import HeatMap

# Initialize the map centered around the dataset
o = folium.Map(location=[gdf["latitude"].mean(), gdf["longitude"].mean()],␣
↪zoom_start=10)

# Add data points to the map
for _, row in gdf.iterrows():
    folium.CircleMarker(
        location=[row["latitude"], row["longitude"]],
        radius=5,
        color="red" if row["iso_forest_score"] < 0 else "blue",
        fill=True,
        fill_color="red" if row["iso_forest_score"] < 0 else "blue",
        fill_opacity=0.7,
        popup=f"PU Code: {row['pu-code']}<br>Score: {row['iso_forest_score']:.
↪4f}"
    ).add_to(o)

o.save("outlier_map_2.html")
```

[186]:
```
o
```

[186]: <folium.folium.Map at 0x7c3500280cd0>

[172]:
```
#compare outlier score accross method
# Select relevant columns
outlier_cols = ["local_moran", "Gi_outlier_score", "iso_forest_score"]

# Pairwise comparison with KDE plots
sns.pairplot(gdf[outlier_cols], diag_kind="kde")
plt.show()
```



[176]:
```
gdf.to_csv('second_to_last.csv')
```

0.1.2 conclusion

[178]:
```python
#Create a Combined Outlier Indicator
# Normalize scores (if necessary)
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
gdf[["local_moran_norm", "Gi_outlier_norm", "iso_forest_norm"]] = scaler.
 ↪fit_transform(
    gdf[["local_moran", "Gi_outlier_score", "iso_forest_score"]]
)

# Aggregate scores into a combined outlier score
gdf["combined_outlier_score"] = (
    gdf["local_moran_norm"] + gdf["Gi_outlier_norm"] + gdf["iso_forest_norm"]
) / 3

# Flag as an outlier if above a certain threshold
threshold = gdf["combined_outlier_score"].quantile(0.95)
gdf["final_outlier_flag"] = gdf["combined_outlier_score"] > threshold

# Show results
gdf[["pu-code", "local_moran", "Gi_outlier_score", "iso_forest_score",␣
 ↪"combined_outlier_score", "final_outlier_flag"]].head()
```

[178]:
```
       pu-code  local_moran  Gi_outlier_score  iso_forest_score  \
0  04-01-01-001    -0.100086         -0.124730          0.112147
1  04-01-01-005    -0.049683         -0.521028          0.097351
2  04-01-01-006    -0.024154         -0.459232          0.118558
3  04-01-01-008    -0.015195         -0.477192          0.105581
4  04-01-01-010     0.138257         -0.665365          0.107372

   combined_outlier_score  final_outlier_flag
0                0.408699               False
1                0.377957               False
2                0.405524               False
3                0.389929               False
4                0.388467               False
```

[181]:
```python
import folium

# Create a base map centered at the mean latitude and longitude
p = folium.Map(location=[gdf.latitude.mean(), gdf.longitude.mean()],␣
 ↪zoom_start=7)

# Define colors manually
def get_marker_color(outlier_flag):
```

```python
    return "red" if outlier_flag else "blue"

# Add polling units to the map WITHOUT clustering
for _, row in gdf.iterrows():
    folium.CircleMarker(
        location=[row["latitude"], row["longitude"]],
        radius=4,
        color=get_marker_color(row["final_outlier_flag"]),
        fill=True,
        fill_color=get_marker_color(row["final_outlier_flag"]),
        fill_opacity=0.7,
    ).add_to(p)
```

[184]:
```python
p.save("outlier_map_3.html")
```

[193]:
```python
p
```

[193]: <folium.folium.Map at 0x7c34fd627450>

[185]:
```python
gdf.to_csv('last.csv')
```

[194]:
```python
gdf.info()
```

```
<class 'geopandas.geodataframe.GeoDataFrame'>
RangeIndex: 3679 entries, 0 to 3678
Data columns (total 48 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   state                  3679 non-null   object
 1   lga                    3679 non-null   object
 2   ward                   3679 non-null   object
 3   pu-code                3679 non-null   object
 4   pu-name                3679 non-null   object
 5   accredited_voters      3679 non-null   int64
 6   registered_voters      3679 non-null   int64
 7   results_found          3679 non-null   bool
 8   transcription_count    3679 non-null   int64
 9   result_sheet_stamped   3679 non-null   bool
 10  result_sheet_corrected 3679 non-null   bool
 11  result_sheet_invalid   3679 non-null   bool
 12  result_sheet_unclear   3679 non-null   bool
 13  result_sheet_unsigned  3679 non-null   object
 14  apc                    3679 non-null   float64
 15  lp                     3679 non-null   float64
 16  pdp                    3679 non-null   float64
 17  nnpp                   3679 non-null   float64
 18  results_file           3679 non-null   object
```

```
 19  latitude                3679 non-null   float64
 20  longitude               3679 non-null   float64
 21  cluster                 3679 non-null   object
 22  cluster_eps_500         3679 non-null   int64
 23  cluster_eps_1000        3679 non-null   int64
 24  cluster_eps_2000        3679 non-null   int64
 25  geometry                3679 non-null   geometry
 26  z_score                 3679 non-null   float64
 27  local_moran             3679 non-null   float64
 28  p_value                 3679 non-null   float64
 29  Gi_star_apc             3679 non-null   float64
 30  Gi_p_value_apc          3679 non-null   float64
 31  hotspot_apc             3679 non-null   bool
 32  Gi_star_lp              3679 non-null   float64
 33  Gi_p_value_lp           3679 non-null   float64
 34  hotspot_lp              3679 non-null   bool
 35  Gi_star_pdp             3679 non-null   float64
 36  Gi_p_value_pdp          3679 non-null   float64
 37  hotspot_pdp             3679 non-null   bool
 38  Gi_star_nnpp            3679 non-null   float64
 39  Gi_p_value_nnpp         3679 non-null   float64
 40  hotspot_nnpp            3679 non-null   bool
 41  Gi_outlier_score        3679 non-null   float64
 42  iso_forest_score        3679 non-null   float64
 43  local_moran_norm        3679 non-null   float64
 44  Gi_outlier_norm         3679 non-null   float64
 45  iso_forest_norm         3679 non-null   float64
 46  combined_outlier_score  3679 non-null   float64
 47  final_outlier_flag      3679 non-null   bool
dtypes: bool(10), float64(23), geometry(1), int64(6), object(8)
memory usage: 1.1+ MB
```

[195]:
```python
# Sort the DataFrame by 'combined_outlier_score' from highest to lowest
gdf_sorted = gdf.sort_values(by="combined_outlier_score", ascending=False)

# Save the sorted DataFrame as a CSV file
gdf_sorted.to_csv("sorted_outliers.csv", index=False)
```

[197]:
```python
#top_five outlier polling units
top_5_outliers = gdf_sorted.head(5)
```

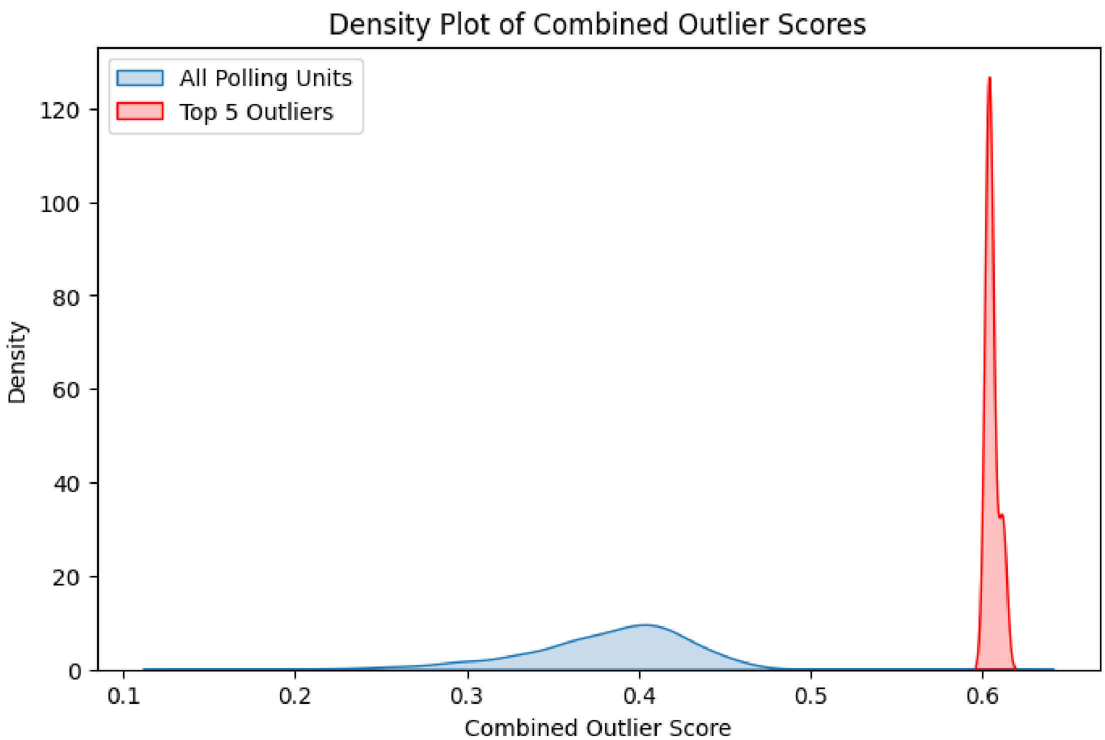
[198]:
```python
import folium

# Base map centered around the top 5 outliers
s = folium.Map(location=[top_5_outliers.latitude.mean(), top_5_outliers.
 ↪longitude.mean()], zoom_start=10)
```

```
# Add markers for the top 5 outlier polling units
for _, row in top_5_outliers.iterrows():
    folium.Marker(
        location=[row["latitude"], row["longitude"]],
        popup=f"PU: {row['pu-code']}<br>Outlier Score: 
 ↪{row['combined_outlier_score']}",
        icon=folium.Icon(color="red", icon="info-sign"),
    ).add_to(s)

# Save the map
s.save("top_5_outliers_map.html")
```

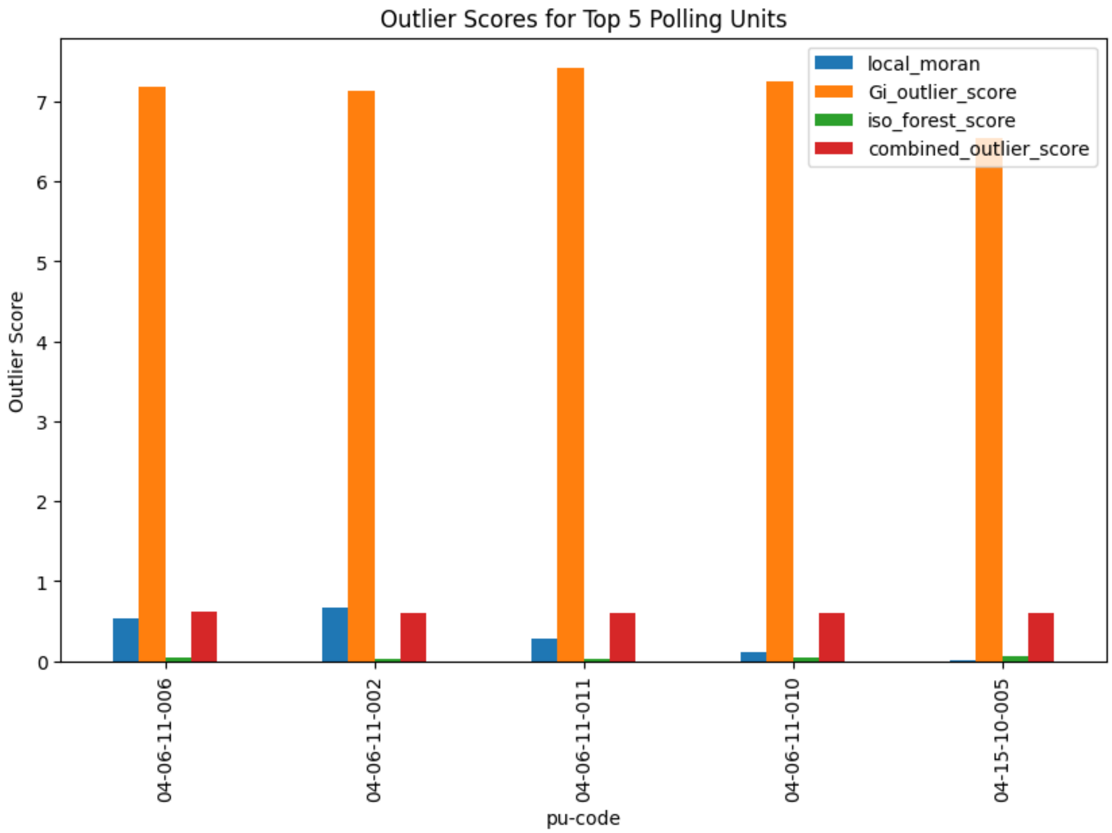
[199]:
```
s
```

[199]: <folium.folium.Map at 0x7c34f38f3350>

[201]:
```
#KDE Plot for Outlier Scores
plt.figure(figsize=(8, 5))
sns.kdeplot(gdf_sorted["combined_outlier_score"], label="All Polling Units", 
 ↪fill=True)
sns.kdeplot(top_5_outliers["combined_outlier_score"], label="Top 5 Outliers", 
 ↪fill=True, color="red")
plt.legend()
plt.title("Density Plot of Combined Outlier Scores")
plt.xlabel("Combined Outlier Score")
plt.ylabel("Density")
plt.show()
```



[203]:
```
#Bar Chart for Top 5 Outliers
top_5_outliers.plot(
    x="pu-code",
    y=["local_moran", "Gi_outlier_score", "iso_forest_score",␣
 ↪"combined_outlier_score"],
    kind="bar",
    figsize=(10, 6),
    title="Outlier Scores for Top 5 Polling Units"
)
plt.ylabel("Outlier Score")
plt.show()
```



[200]:
```
top_5_outliers
```

[200]:
```
        state          lga                ward        pu-code  \
911   ANAMBRA   AWKA NORTH  MGBAKWU  I (ANEZIKE)  04-06-11-006
908   ANAMBRA   AWKA NORTH  MGBAKWU  I (ANEZIKE)  04-06-11-002
916   ANAMBRA   AWKA NORTH  MGBAKWU  I (ANEZIKE)  04-06-11-011
915   ANAMBRA   AWKA NORTH  MGBAKWU  I (ANEZIKE)  04-06-11-010
1977  ANAMBRA  NNEWI SOUTH        EZINIFITE  III  04-15-10-005

                    pu-name  accredited_voters  registered_voters  \
911         AMAEZE V. HALL II                165                731
908            C/S MGBAKWU II                181               1021
916         AMAUDALA V SQUARE                140                521
915   AMAMKPU VILLAGE SQUARE                 125                768
1977              ERIMA HALL I                114                444

      results_found  transcription_count  result_sheet_stamped  …  \
911            True                   -1                 False  …
908            True                   -1                 False  …
916            True                   -1                 False  …
915            True                   -1                 False  …
```

```
1977           True                   -1                False  …

      Gi_star_nnpp  Gi_p_value_nnpp  hotspot_nnpp Gi_outlier_score  \
911      -0.068361            0.226         False         7.186319
908      -0.068361            0.226         False         7.132298
916      -0.068361            0.227         False         7.426682
915      -0.097535            0.001          True         7.253486
1977     -0.068361            0.224         False         6.538304

      iso_forest_score  local_moran_norm  Gi_outlier_norm  iso_forest_norm  \
911           0.043277          0.210007         0.973429         0.651945
908           0.035905          0.219526         0.967458         0.626156
916           0.033936          0.193329         1.000000         0.619271
915           0.042443          0.182266         0.980854         0.649028
1977          0.066770          0.175119         0.901795         0.734121

     combined_outlier_score  final_outlier_flag
911                0.611794                True
908                0.604380                True
916                0.604200                True
915                0.604049                True
1977               0.603678                True

[5 rows x 48 columns]
```