lab8-ej-clustering-

March 8, 2025

0.1 Lab 8: Clustering

Environmental justice (EJ) seeks to ensure that all communities, regardless of socioeconomic status or demographic background, have equal access to clean air, water, and, in the form of energy justice, access to renewable energy resources while minimizing exposure to environmental hazards. In this lab, we will apply clustering analysis to explore how U.S. counties group together based on renewable energy potential, demographic characteristics, and environmental risk factors.

The EEIP dataset was collated by the National Renewable Energy Lab (NREL) and contains a large set of features from multiple other databases including SLOPE (renewable energy potential) and EJSCREEN (environmental risk indicators).

Link to metadata: https://ucsb.box.com/s/x3olvh3rd8w5h7xz8jnm3v8g3t4ajjsg

First you will step through a guided clustering exploration of renewable energy production potential. Then you will formulate a question of your own that brings in an environmental justice component.

0.1.1 Step 0: Load Libraries and Data

[1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from tabulate import tabulate
import seaborn as sns

# Load the EEIP dataset
eeip_data = pd.read_csv("data/eeip.data.csv")
```

[2]:
```python
eeip_data
```

[2]:
```
      county_fips      county state  county_pop  utilitypv_techpot_quint  \
0            1001     Autauga    AL       55200                        3
1            1003     Baldwin    AL      208107                        5
2            1005     Barbour    AL       25782                        4
3            1007        Bibb    AL       22527                        2
4            1009      Blount    AL       57645                        2
…               …           …   …           …                        …
3103        56037  Sweetwater    WY       44117                        5
```

```
3104        56039       Teton    WY       23059                               1
3105        56041       Uinta    WY       20609                               5
3106        56043    Washakie    WY        8129                               4
3107        56045      Weston    WY        7100                               5

      utility_pv_technical_generation_potential_mwh  \
0                                      3.585222e+07
1                                      1.257822e+08
2                                      6.614827e+07
3                                      1.926909e+07
4                                      2.261702e+07
…                                                 …
3103                                   1.028709e+09
3104                                   5.383872e+06
3105                                   1.290202e+08
3106                                   9.235474e+07
3107                                   2.133493e+08

      residentialpv_techpot_quint  \
0                             4.0
1                             5.0
2                             3.0
3                             2.0
4                             4.0
…                               …
3103                          4.0
3104                          2.0
3105                          2.0
3106                          1.0
3107                          1.0

      residential_pv_technical_generation_potential_mwh  \
0                                             122752.69
1                                             483008.57
2                                              42823.79
3                                              37917.84
4                                             122024.81
…                                                     …
3103                                           92790.23
3104                                           33922.11
3105                                           40010.06
3106                                           21053.43
3107                                           16341.73

      landbasedwind_techpot_quint  \
0                               3
1                               3
```

```
2                               3
3                               3
4                               3
...                         ...
3103                            5
3104                            1
3105                            5
3106                            5
3107                            5

      land_based_wind_technical_generation_potential_mwh  ...  rmpprox_2_prop  \
0                                           4.374954e+06  ...        0.156250
1                                           4.368632e+06  ...        0.127660
2                                           5.898865e+06  ...        0.173913
3                                           3.986770e+06  ...        0.000000
4                                           6.117475e+06  ...        0.085714
...                                                  ...  ...             ...
3103                                        2.032127e+08  ...        0.117647
3104                                        4.040734e+05  ...        0.000000
3105                                        3.105440e+07  ...        0.000000
3106                                        2.107509e+07  ...        0.750000
3107                                        3.022243e+07  ...        0.200000

      rmpprox_3_prop  rmpprox_4_prop  rmpprox_5_prop  tsdf_indicator  \
0           0.031250        0.000000        0.000000             0.0
1           0.031915        0.053191        0.000000             0.0
2           0.304348        0.130435        0.000000             0.0
3           0.000000        0.000000        0.000000             0.0
4           0.057143        0.000000        0.000000             0.0
...              ...             ...             ...             ...
3103        0.000000        0.323529        0.029412             0.0
3104        0.000000        0.000000        0.000000             0.0
3105        0.250000        0.250000        0.187500             0.0
3106        0.000000        0.000000        0.000000             0.0
3107        0.000000        0.400000        0.200000             0.0

      tsdfprox_1_prop  tsdfprox_2_prop  tsdfprox_3_prop  tsdfprox_4_prop  \
0            0.343750         0.187500         0.281250         0.156250
1            0.521277         0.297872         0.159574         0.021277
2            1.000000         0.000000         0.000000         0.000000
3            0.333333         0.600000         0.066667         0.000000
4            0.714286         0.171429         0.114286         0.000000
...               ...              ...              ...              ...
3103         0.088235         0.411765         0.411765         0.058824
3104         1.000000         0.000000         0.000000         0.000000
3105         0.812500         0.062500         0.125000         0.000000
3106         1.000000         0.000000         0.000000         0.000000
```

```
3107          0.400000           0.000000           0.600000           0.000000

      tsdfprox_5_prop
0              0.031250
1              0.000000
2              0.000000
3              0.000000
4              0.000000
…                   …
3103           0.029412
3104           0.000000
3105           0.000000
3106           0.000000
3107           0.000000

[3108 rows x 152 columns]
```

0.2 Part I:

In this part, we will step through an analysis that examines how US counties cluster in their potential production of renewable energy.

0.2.1 Step 1: Exploratory Data Analysis

First we need to check for missing data and remove incomplete rows. Since clustering is a distance-based technique, we also need to ensure that the features used for clustering are scaled appropriately to prevent dominant features from skewing results. For our first analysis, use the following variables from the SLOPE dataset related to energy production potential as your features: - `utility_pv_technical_generation_potential_mwh` - `residential_pv_technical_generation_potential_mwh` - `land_based_wind_technical_generation_potenti` - `commercial_pv_technical_generation_potential_mwh`

Information on these variables is available on line 7 of the ColumnsExplained tab of the metadata

Once you have removed incomplete rows and scaled, print the shape of your processed dataframe.

[3]:
```python
slope_features = eeip_data[["utility_pv_technical_generation_potential_mwh",
                            "residential_pv_technical_generation_potential_mwh",
                            "land_based_wind_technical_generation_potential_mwh",
                            "commercial_pv_technical_generation_potential_mwh"]]

slope_features.head()
```

[3]:
```
   utility_pv_technical_generation_potential_mwh  \
0                                   3.585222e+07
1                                   1.257822e+08
2                                   6.614827e+07
3                                   1.926909e+07
4                                   2.261702e+07
```

```
   residential_pv_technical_generation_potential_mwh  \
0                                           122752.69
1                                           483008.57
2                                            42823.79
3                                            37917.84
4                                           122024.81

   land_based_wind_technical_generation_potential_mwh  \
0                                          4374954.41
1                                          4368631.72
2                                          5898864.51
3                                          3986770.03
4                                          6117474.83

   commercial_pv_technical_generation_potential_mwh
0                                          72863.02
1                                         361886.15
2                                          88221.80
3                                          64286.62
4                                         290436.00
```

[4]:
```
# Drop incomplete rows
slope_features = slope_features.dropna()
slope_features_columns = slope_features.columns

# Scale the data
scaler = StandardScaler()
slope_features_scaled = scaler.fit_transform(slope_features)

slope_features_scaled = pd.DataFrame(slope_features_scaled,␣
 ↪columns=slope_features_columns)
slope_features_scaled.head()
```

[4]:
```
   utility_pv_technical_generation_potential_mwh  \
0                                      -0.331171
1                                       0.638649
2                                      -0.004453
3                                      -0.510007
4                                      -0.473902

   residential_pv_technical_generation_potential_mwh  \
0                                          -0.121811
1                                           0.549126
2                                          -0.270669
3                                          -0.279806
4                                          -0.123166
```

```
   land_based_wind_technical_generation_potential_mwh  \
0                                          -0.369593
1                                          -0.370021
2                                          -0.266443
3                                          -0.395868
4                                          -0.251646

   commercial_pv_technical_generation_potential_mwh
0                                         -0.214820
1                                          0.081081
2                                         -0.199095
3                                         -0.223600
4                                          0.007931
```

[5]:
```
print(f"Shape of processed dataframe: {slope_features_scaled.shape}")
```

```
Shape of processed dataframe: (3107, 4)
```

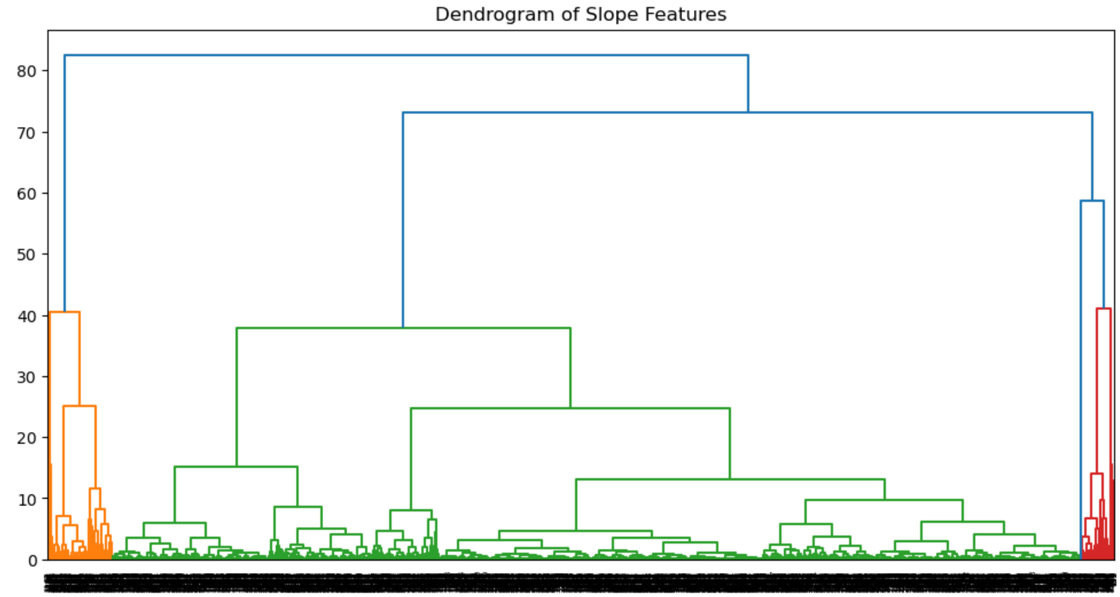
0.2.2 Step 2: Hierarchical Clustering Analysis

Now that we have preprocessed our dataset and standardized the energy potential features, we will use hierarchical clustering to explore how counties group together based on their energy potential.

A dendrogram is a tree-like visualization that shows how clusters are merged step by step. By analyzing the dendrogram, we can decide the optimal number of clusters by looking at the distance between merges.

Use `linkage()` to perform the clustering. Use 'ward' for the `method` parameter, a method which minimizes the variance within clusters, producing well-balanced groups. We will first visualize the **full dendrogram** using `dendrogram` before deciding on a truncation strategy.

[6]:
```
slope_link = linkage(slope_features_scaled, method='ward', metric='euclidean')

plt.figure(figsize=(12, 6))
dendrogram(slope_link, labels=slope_features.index, leaf_rotation=90)
plt.title('Dendrogram of Slope Features')
plt.show()
```



0.2.3 Step 3: Set Up Cluster Interpretation

After determining the optimal number of clusters from the dendrogram, we now assign each county to a cluster using the extracted cluster labels.

To better understand the clustering results, we will: - Define `num_clusters` as the ideal number of clusters based on the dendrogram created above - Extract cluster assignment attribute from the hierarchical clustering using `fcluster()` with `criterion = "maxclust"` - Create a new variable `Cluster` in your cleaned dataframe and assign cluster labels to it - Compute and print the mean values of the original energy potential features for each cluster.

This summary will help us interpret how counties differ in energy potential across clusters and inform possible next steps for analysis.

[7]:
```
num_clusters = 4

caa = fcluster(slope_link, num_clusters, criterion = "maxclust")

slope_features["Cluster"] = caa
slope_features_scaled["Cluster"] = caa

og_cluster_mean = slope_features.groupby('Cluster').mean().reset_index()
scaled_cluster_mean = slope_features_scaled.groupby('Cluster').mean().
 ↪reset_index()

print(f"Original mean energy potential feature values per cluster:")
print()
print(og_cluster_mean.to_markdown())
```

```
print("\n")

print(f"Scaled mean energy potential feature values per cluster:")
print()
print(scaled_cluster_mean.to_markdown())
```

```
Original mean energy potential feature values per cluster:

|    |   Cluster |   utility_pv_technical_generation_potential_mwh |
residential_pv_technical_generation_potential_mwh |
land_based_wind_technical_generation_potential_mwh |
commercial_pv_technical_generation_potential_mwh |
|---:|----------:|-----------------------------------------------:|------------
---------------------------------------:|-------------------------------------
---------------:|-------------------------------------------------:|
|  0 |         1 |                                   3.29082e+08 |
84339.3          |                                        5.42242e+07 |
89428.4          |
|  1 |         2 |                                   4.94304e+07 |
122893           |                                        7.09061e+06 |
180186           |
|  2 |         3 |                                   5.94802e+07 |
2.1382e+06   |                                   4.18947e+06 |
3.25869e+06 |
|  3 |         4 |                                   5.2458e+06  |
1.46256e+07 |                                   2.04652e+06 |
3.70152e+07 |


Scaled mean energy potential feature values per cluster:

|    |   Cluster |   utility_pv_technical_generation_potential_mwh |
residential_pv_technical_generation_potential_mwh |
land_based_wind_technical_generation_potential_mwh |
commercial_pv_technical_generation_potential_mwh |
|---:|----------:|-----------------------------------------------:|------------
---------------------------------------:|-------------------------------------
---------------:|-------------------------------------------------:|
|  0 |         1 |                                     2.83107   |
-0.193351 |                                     3.00458   |
-0.19786   |
|  1 |         2 |                                    -0.184742  |
-0.121549 |                                    -0.185776 |
-0.104943 |
|  2 |         3 |                                    -0.0763628 |
3.63175   |                                   -0.382148 |
3.04682   |
```

| 3 | 4 | -0.661236 |
26.8882 | -0.527199 |
37.6066 |

0.2.4 Step 4: Visualizing Energy Potential Across Clusters

Now that we have assigned cluster labels, we want to understand how energy potential differs across clusters. To do this, we will visualize these differences using a grouped bar chart.

Each bar should represent the mean value of an energy potential indicator for a specific cluster. These different patterns of potential is what caused the model to segregate the clusters in the way that it did.

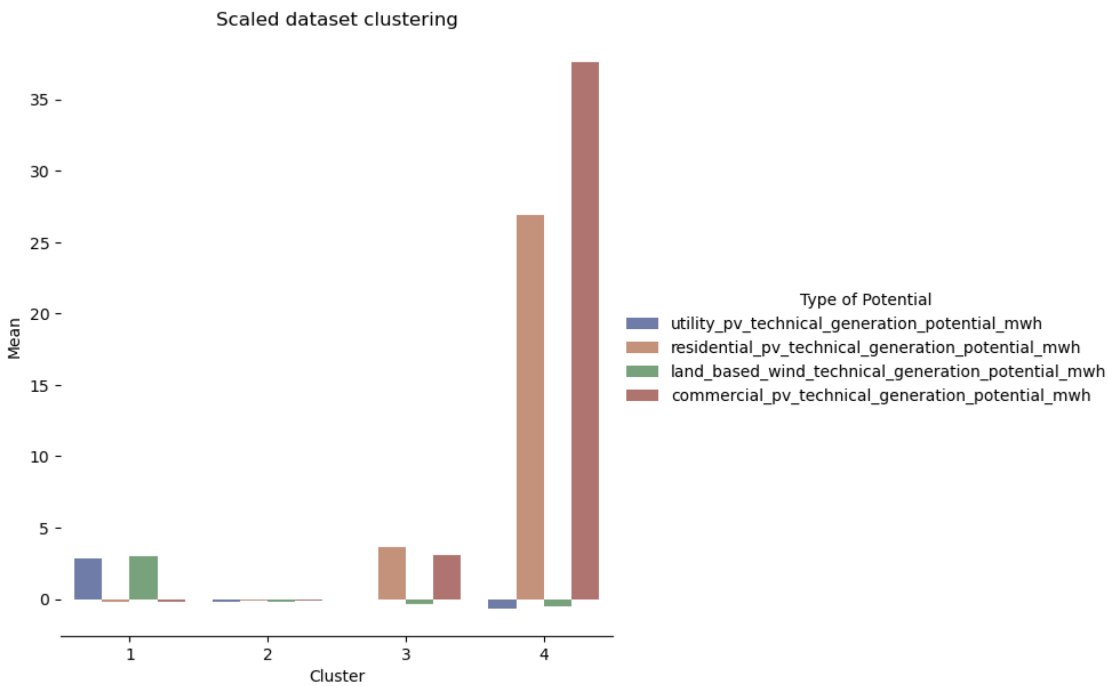
[8]:
```
scaled_cluster_melt = scaled_cluster_mean.melt(id_vars=["Cluster"],␣
 ↪var_name="Type of Potential", value_name="Mean")
scaled_cluster_melt
```

[8]:

	Cluster	Type of Potential	Mean
0	1	utility_pv_technical_generation_potential_mwh	2.831068
1	2	utility_pv_technical_generation_potential_mwh	-0.184742
2	3	utility_pv_technical_generation_potential_mwh	-0.076363
3	4	utility_pv_technical_generation_potential_mwh	-0.661236
4	1	residential_pv_technical_generation_potential_mwh	-0.193351
5	2	residential_pv_technical_generation_potential_mwh	-0.121549
6	3	residential_pv_technical_generation_potential_mwh	3.631745
7	4	residential_pv_technical_generation_potential_mwh	26.888227
8	1	land_based_wind_technical_generation_potential…	3.004580
9	2	land_based_wind_technical_generation_potential…	-0.185776
10	3	land_based_wind_technical_generation_potential…	-0.382148
11	4	land_based_wind_technical_generation_potential…	-0.527199
12	1	commercial_pv_technical_generation_potential_mwh	-0.197860
13	2	commercial_pv_technical_generation_potential_mwh	-0.104943
14	3	commercial_pv_technical_generation_potential_mwh	3.046819
15	4	commercial_pv_technical_generation_potential_mwh	37.606587

[19]:
```
# Draw a nested barplot by species and sex
g = sns.catplot(
    data=scaled_cluster_melt, kind="bar",
    x="Cluster", y="Mean", hue="Type of Potential",
    errorbar="sd", palette="dark", alpha=.6, height=6
)

g.despine(left=True)
g.set_axis_labels("Cluster", "Mean")
g.legend.set_title("Type of Potential")
g.set(title="Scaled dataset clustering")
```

[19]: <seaborn.axisgrid.FacetGrid at 0x162caacb0>



[15]:
```
og_cluster_melt = og_cluster_mean.melt(id_vars=["Cluster"], var_name="Type of␣
↪Potential", value_name="Mean")
og_cluster_melt
```

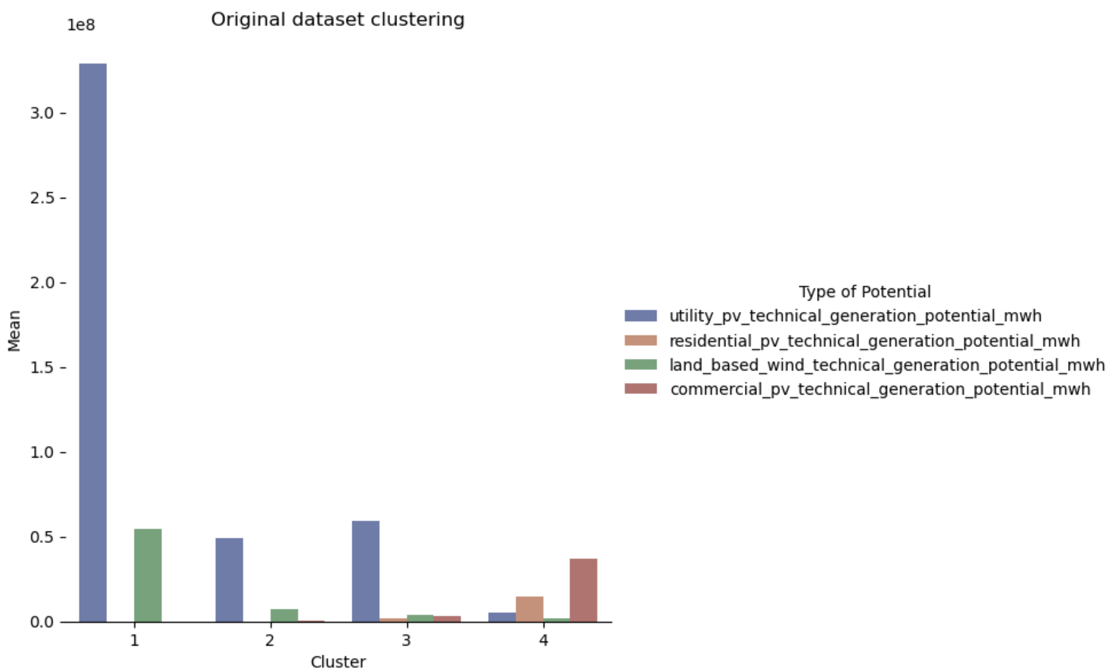
[15]:

	Cluster	Type of Potential	Mean
0	1	utility_pv_technical_generation_potential_mwh	3.290819e+08
1	2	utility_pv_technical_generation_potential_mwh	4.943040e+07
2	3	utility_pv_technical_generation_potential_mwh	5.948022e+07
3	4	utility_pv_technical_generation_potential_mwh	5.245800e+06
4	1	residential_pv_technical_generation_potential_mwh	8.433934e+04
5	2	residential_pv_technical_generation_potential_mwh	1.228932e+05
6	3	residential_pv_technical_generation_potential_mwh	2.138205e+06
7	4	residential_pv_technical_generation_potential_mwh	1.462565e+07
8	1	land_based_wind_technical_generation_potential…	5.422419e+07
9	2	land_based_wind_technical_generation_potential…	7.090613e+06
10	3	land_based_wind_technical_generation_potential…	4.189469e+06
11	4	land_based_wind_technical_generation_potential…	2.046515e+06
12	1	commercial_pv_technical_generation_potential_mwh	8.942839e+04
13	2	commercial_pv_technical_generation_potential_mwh	1.801859e+05
14	3	commercial_pv_technical_generation_potential_mwh	3.258689e+06
15	4	commercial_pv_technical_generation_potential_mwh	3.701515e+07

[20]:
```
# Draw a nested barplot by species and sex
g = sns.catplot(
    data=og_cluster_melt, kind="bar",
```

```
    x="Cluster", y="Mean", hue="Type of Potential",
    errorbar="sd", palette="dark", alpha=.6, height=6
)

g.despine(left=True)
g.set_axis_labels("Cluster", "Mean")
g.legend.set_title("Type of Potential")
g.set(title="Original dataset clustering")
```

[20]: <seaborn.axisgrid.FacetGrid at 0x162ccb580>



0.2.5 Step 5: Interpret Clustering Results

Interpret your plot of the resulting clusters. How would you characterize and compare the four different clusters in terms of their profile of energy generation?

In the clustering of the original data set means, we see utility have the greatest variance, especially in county cluster 1. However, in the scaled data, we find that there is a high devience in cluster 4, especially in the residential and commerical potential. Multiple inferences can be derrived from this.

This means that overall, utility is has a higher absolute mean mwh, however, when commerical generation is utilized, it is used at a much higher scale, which is visible in cluster 4. This means certian counties specilize in different industries.

0.3 Part II: Environmental Justice Metrics

Now it's your turn.

So far, we have clustered counties based on **energy potential**, but energy potential alone does not tell the full story of **energy equity and access**. To deepen the analysis, we need to consider environmental justice (EJ) factors that affect communities' ability to benefit from renewable energy and the environmental burdens they already experience.

1. Explore EJSCREEN variables:
    - The EJSCREEN subset of our dataset contains metrics on pollution burden, demographics (population size), and health risks (*lines 31-41 of the ColumnsExplained tab in the metadata sheet linked above*).
    - Identify 1-3 variables that could be important for energy equity analysis. You could introduce them either as clustering features, as post-clustering variables to help interpret the clusters, or both.
2. Modify the clustering approach:
    - Add your selected EJSCREEN variables to our feature set.

    - Re-run the hierarchical clustering analysis with the expanded dataset (if you added any as clustering features).
3. Interpret the Results: Your interpretation could include considerations such as:
    - How do clusters change when EJSCREEN variables are included?

    - Are counties with high renewable energy potential also burdened by environmental risks?

    - What policy recommendations might emerge from these findings?

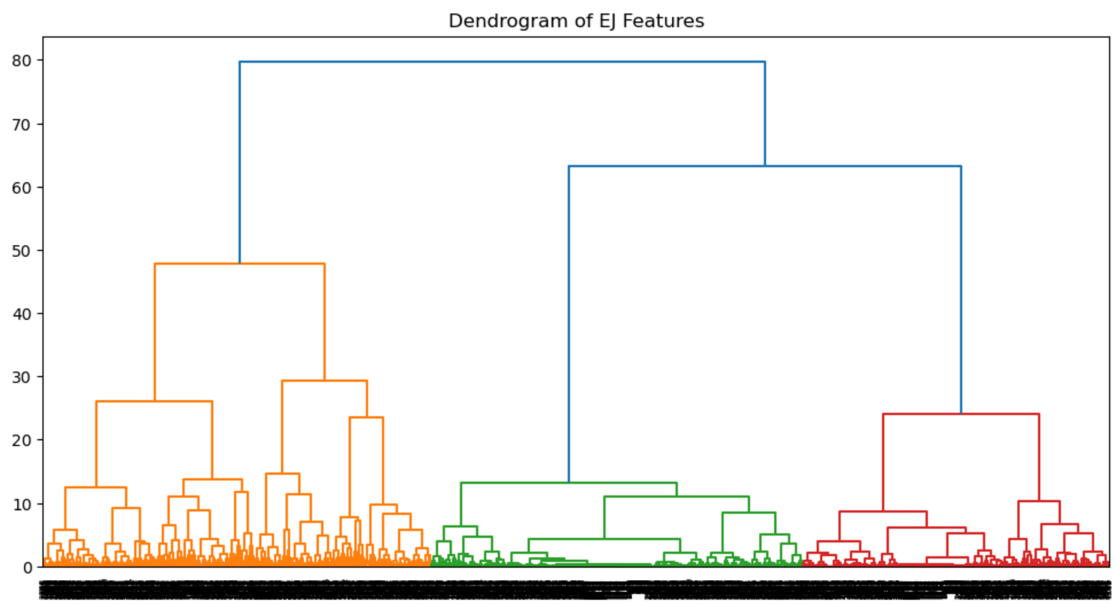
[31]:
```
eeip_data.columns[-20:]
```

[31]:
```
Index(['wastewaterdischarge_4_prop', 'wastewaterdischarge_5_prop',
       'npl_indicator', 'nplprox_1_prop', 'nplprox_2_prop', 'nplprox_3_prop',
       'nplprox_4_prop', 'nplprox_5_prop', 'rmp_indicator', 'rmpprox_1_prop',
       'rmpprox_2_prop', 'rmpprox_3_prop', 'rmpprox_4_prop', 'rmpprox_5_prop',
       'tsdf_indicator', 'tsdfprox_1_prop', 'tsdfprox_2_prop',
       'tsdfprox_3_prop', 'tsdfprox_4_prop', 'tsdfprox_5_prop'],
      dtype='object')
```

[32]:
```
# explore
EJ_features = eeip_data[[
    # Prop low income
    "lowincome_indicator",
    # Proximity to national priority list proportion
    "nplprox_1_prop",
    # Prop less than hs education
    "lessthanhs_indicator"
    ]]
```

Apply same transformations

[34]:
```
# Drop incomplete rows
EJ_features = EJ_features.dropna()
EJ_features_columns = EJ_features.columns

# Scale the data
scaler = StandardScaler()
EJ_features_scaled = scaler.fit_transform(EJ_features)

EJ_features_scaled = pd.DataFrame(EJ_features_scaled,␣
↪columns=EJ_features_columns)

EJ_link = linkage(EJ_features_scaled, method='ward', metric='euclidean')

plt.figure(figsize=(12, 6))
dendrogram(EJ_link, labels=EJ_features.index, leaf_rotation=90)
plt.title('Dendrogram of EJ Features')
plt.show()
```



[37]:
```
EJ_num_clusters = 3

EJ_caa = fcluster(EJ_link, EJ_num_clusters, criterion = "maxclust")

EJ_features["Cluster"] = EJ_caa
EJ_features_scaled["Cluster"] = EJ_caa

EJ_og_cluster_mean = EJ_features.groupby('Cluster').mean().reset_index()
```

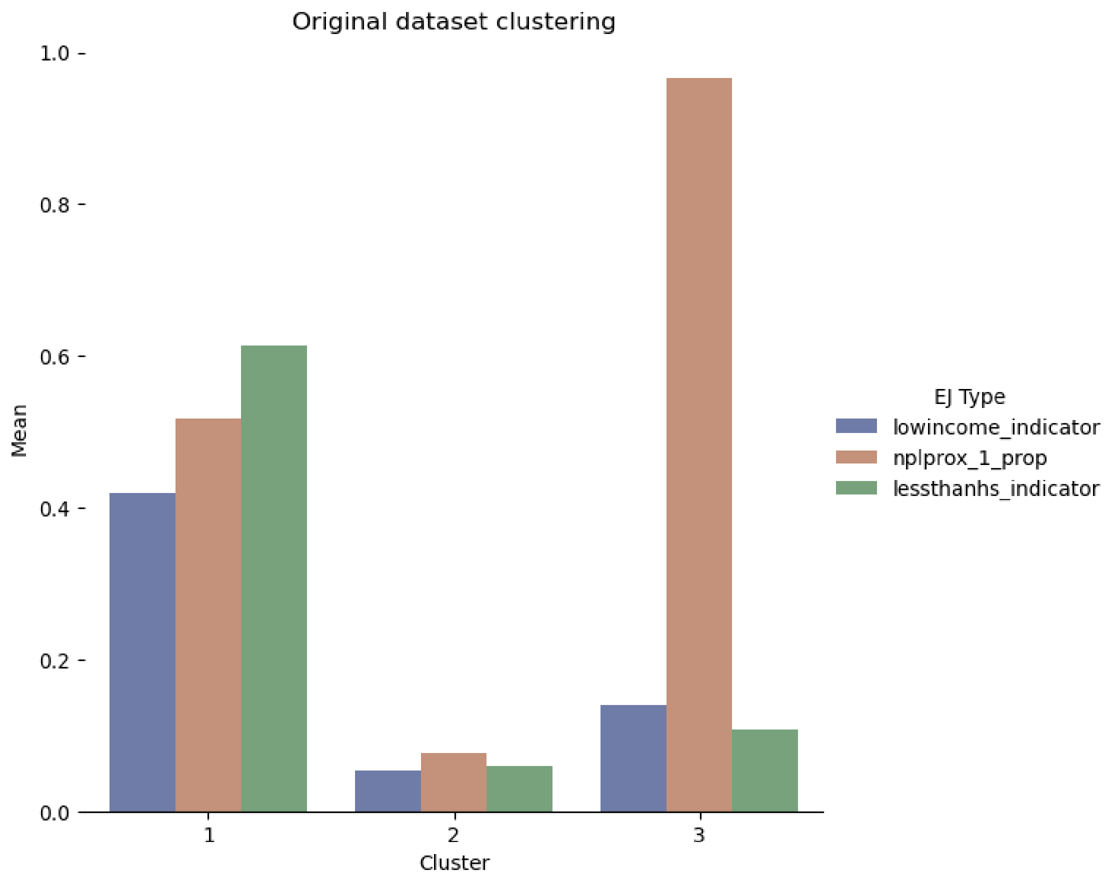
```
EJ_scaled_cluster_mean = EJ_features_scaled.groupby('Cluster').mean().
 ↪reset_index()

print(f"Original EJ feature values per cluster:")
print()
print(EJ_og_cluster_mean.to_markdown())
print("\n")

print(f"Scaled EJ feature values per cluster:")
print()
print(EJ_scaled_cluster_mean.to_markdown())
```

```
Original EJ feature values per cluster:

|    |   Cluster |   lowincome_indicator |   nplprox_1_prop |
lessthanhs_indicator |
|---:|----------:|----------------------:|-----------------:|--------------------
----:|
|  0 |         1 |             0.419565  |        0.518449  |
0.614855  |
|  1 |         2 |             0.0527917 |        0.0766473 |
0.0583698 |
|  2 |         3 |             0.14048   |        0.966223  |
0.106371  |


Scaled EJ feature values per cluster:

|    |   Cluster |   lowincome_indicator |   nplprox_1_prop |
lessthanhs_indicator |
|---:|----------:|----------------------:|-----------------:|--------------------
----:|
|  0 |         1 |              0.909694 |        0.0552866 |
0.978611 |
|  1 |         2 |             -0.693988 |       -0.927157  |
-0.622359 |
|  2 |         3 |             -0.31058  |        1.05101   |
-0.484264 |
```

[40]:
```
EJ_og_cluster_melt = EJ_og_cluster_mean.melt(id_vars=["Cluster"], var_name="EJ␣
 ↪Type", value_name="Mean").reset_index()
# Draw a nested barplot by species and sex
g = sns.catplot(
    data=EJ_og_cluster_melt, kind="bar",
    x="Cluster", y="Mean", hue="EJ Type",
    errorbar="sd", palette="dark", alpha=.6, height=6
)
```

```
g.despine(left=True)
g.set_axis_labels("Cluster", "Mean")
g.legend.set_title("EJ Type")
g.set(title="Original dataset clustering")
```

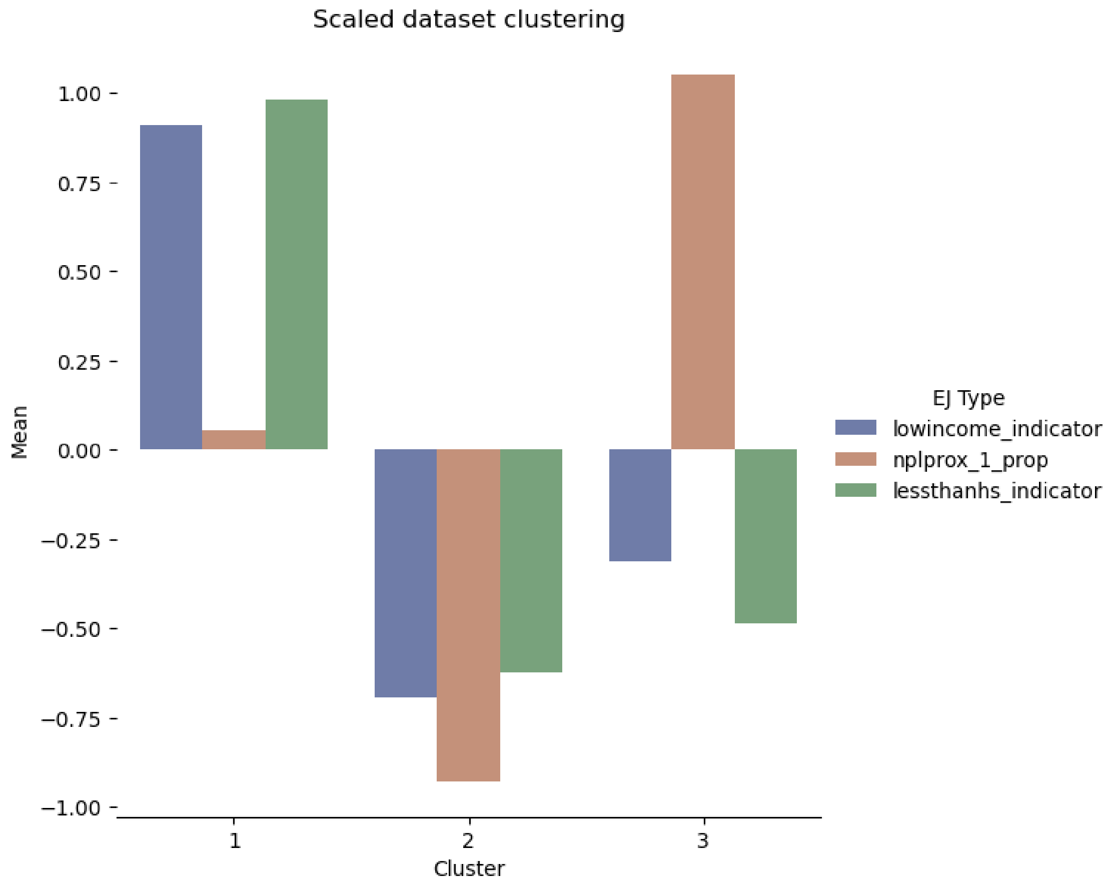
[40]: <seaborn.axisgrid.FacetGrid at 0x1671c4040>



[43]:
```
EJ_scaled_cluster_melt = EJ_scaled_cluster_mean.melt(id_vars=["Cluster"],␣
 ↪var_name="EJ Type", value_name="Mean").reset_index()
# Draw a nested barplot by species and sex
g = sns.catplot(
    data=EJ_scaled_cluster_melt, kind="bar",
    x="Cluster", y="Mean", hue="EJ Type",
    errorbar="sd", palette="dark", alpha=.6, height=6
)

g.despine(left=True)
g.set_axis_labels("Cluster", "Mean")
```

```
g.legend.set_title("EJ Type")
g.set(title="Scaled dataset clustering")
```

[43]: <seaborn.axisgrid.FacetGrid at 0x1750968c0>



Which EJSCREEN variable(s) did you add to the analysis? Why did you choose these? What is the question you are interested in? What did you learn from the analysis

Your answer here