ON SOME APPLICATIONS RELATED TO SUPERVISED AND UNSUPERVISED LEARNING

(Insights from Mobility and Gene Expression Data)

A Project Report submitted to the

Institute of Mathematics and Applications, Bhubaneswar

in partial fulfillment of the requirements

of the Degree of

Master of Science

in

Mathematics with Data Science

by

Abhishek Kumar Dubey

University Roll No.: **E1773U233005**

under the supervision of

Dr. Atanu Kumar Ghosh

(Assistant Professor, Dept. of Statistics)

Presidency University, Kolkata

INSTITUTE OF MATHEMATICS AND APPLICATIONS

Bhubaneswar, Odisha - 751029

INSTITUTE OF MATHEMATICS AND APPLICATIONS

Andharua, Bhubaneswar - 751029

Dr. Atanu Kumar Ghosh
Assistant Professor
Department of Statistics
Presidency University, Kolkata

Date: 09/12/2024

Supervisor's Certificate

This is to certify that the work presented in this project entitled "**ON SOME APPLICATIONS RELATED TO SUPERVISED AND UNSUPERVISED LEARNING (Insights from Mobility and Gene Expression Data)**" by **Abhishek Kumar Dubey**, University Roll Number: **E1773U233005**, is a record of original research/ review work carried out by him under my supervision and guidance in partial fulfillment of the requirements of the M.Sc. in Mathematics with Data Science. Neither this project nor any part of it has been submitted for any degree or diploma to any institute or university in India or abroad.

.

Atanu Kumar Ghosh

(Supervisor's signature)

Dedicated to
Dr. Atanu Kumar Ghosh

Declaration

I, **Abhishek Kumar Dubey**, University Roll Number **E1773U233005**, hereby declare that this project entitled "**ON SOME APPLICATIONS RELATED TO SUPERVISED AND UNSUPERVISED LEARNING (Insights from Mobility and Gene Expression Data)**" represents my original/review work carried out as an M.Sc. in Mathematics with Data Science student of IMA, Bhubaneswar and to the best of my knowledge, it is not a complete copy of previously published or written by another person, nor any material presented for award of any other degree or diploma of IMA, Bhubaneswar or any other institution. Any contribution made to this research by others, with whom I have worked at IMA, Bhubaneswar or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the section References.

Date: 09/12/2024

(Student's signature)

Acknowledgment

I would like to express my sincere gratitude to **Dr. Atanu Kumar Ghosh** for his guidance and support throughout the course of this project. His expertise and feedback have been instrumental in shaping this report. I am also grateful to the **Presidency University, Kolkata** for providing the resources and environment conducive to research and learning. Special thanks to my peers and family for their continuous encouragement and support. Finally, I acknowledge the authors of the various sources referenced in this report, whose work has significantly contributed to the development of this project.

Date: 09/12/2024
IMA, Bhubaneswar

Abhishek Kumar Dubey
Roll No.: E1773U233005

Abstract

The study explores the application of supervised and unsupervised learning techniques in two distinct data domains: economic mobility and gene expression data. Supervised learning is utilized to predict economic mobility across 741 communities in the United States, focusing on factors such as income inequality, racial segregation, and educational investment. By employing regression models, the study reveals significant regional disparities in economic mobility, emphasizing the crucial role of local socio-economic conditions. Unsupervised learning is applied to gene expression data from 64 different tumor types, using clustering algorithms and Principal Component Analysis (PCA) to uncover patterns and structures within the data. The results demonstrate how these statistical techniques can provide valuable insights into complex phenomena, such as economic mobility and cancer biology, offering potential pathways for further research and policy interventions.

Contents

On some applications related to supervised and unsupervised learning: Insights from Mobility and Gene Expression Data

Introduction

The aim of multivariate statistical studies is to explore and understand the nature of complex objects and phenomena by classifying them into different categories or predicting certain outcomes. This process often involves analyzing multiple variables simultaneously to discern patterns, relationships, and structures within the data. Classification, a fundamental aspect of multivariate analysis, can be broadly categorized into two types: supervised and unsupervised learning.

Types of Classification

Supervised Learning

This approach involves using labeled data to train a model to predict the class or value of new, unseen data. Labeled data means that each training example is paired with an output label.

Example: Imagine a model designed to predict whether a student will pass or fail an exam. The model is trained using a dataset where each student's exam score is labeled as "pass" or "fail." The model learns from these examples and can then predict the outcome for new students based on their scores. Another example is a spam email classifier, which is trained on a dataset of emails labeled as "spam" or "not spam." The model learns the characteristics of spam emails and can classify new emails accordingly.

Mathematical Expression: In supervised learning, the goal is to learn a function f from a set of training examples (x_i, y_i) where x_i represents the input features and y_i represents the output labels. The objective is to minimize a loss function $L(y, f(x))$, such as mean squared error (MSE) in regression:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2$$

For classification, a common loss function is cross-entropy:

$$\text{Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(f(x_i)) + (1-y_i)\log(1-f(x_i))\right]$$

Unsupervised Learning

In contrast, unsupervised learning involves analyzing data without labeled responses. The goal is to discover hidden patterns, structures, or groupings within the data.

Example: Consider a scenario where we have a dataset of customer purchase histories. Without any labels indicating customer segments, an unsupervised learning algorithm such as K-means clustering can group customers into clusters based on their purchasing behavior. These clusters might represent different types of shoppers, such as "bargain hunters" or "loyal customers." Another example is using hierarchical clustering to analyze gene expression data, where the algorithm groups similar genes together based on their expression profiles, helping researchers identify potential functional relationships between genes.

Mathematical Expression: In K-means clustering, the objective is to partition the data into K clusters such that the within-cluster sum of squares (WCSS) is minimized. This is achieved by minimizing the following objective function:

$$\text{WCSS} = \sum_{k=1}^{K}\sum_{i \in C_k} \|x_i - \mu_k\|^2$$

where μ_k is the centroid of cluster k, and C_k represents the set of points in cluster k.

In hierarchical clustering, the process involves creating a dendrogram that merges or splits clusters based on a linkage criterion, such as single linkage (minimum distance between clusters):

$$\text{Single Linkage} = \min\{\|x_i - x_j\| : x_i \in C_a, x_j \in C_b\}$$

or complete linkage (maximum distance between clusters):

$$\text{Complete Linkage} = \max\{\|x_i - x_j\| : x_i \in C_a, x_j \in C_b\}$$

Applications: Insights from Mobility and Gene Expression Data

By applying supervised and unsupervised learning techniques to the provided mobility and gene expression datasets, we aim to uncover valuable insights into socioeconomic factors influencing economic mobility and identify patterns in

gene expression profiles related to different types of cancer. This project will demonstrate how these statistical models can help understand and predict complex phenomena, providing a comprehensive overview of the capabilities and applications of multivariate statistical methods.

Part 1: Supervised Learning Analysis with Mobility Data

Dataset Overview

The `mobility.csv` dataset contains information about 741 communities with various variables. The primary focus is to predict economic mobility, defined as the probability that a child born in the lowest income quintile will reach the top quintile by age 30.

Economic mobility, defined as the probability that a child born in the lowest income quintile will rise to the top quintile by age 30, is a critical measure of the opportunities available within a society. This study explores the factors influencing economic mobility across 741 communities in the United States. By examining a range of socio-economic variables, we aim to understand the determinants of economic mobility and identify the conditions that foster upward economic movement.

Our analysis leverages various statistical techniques to dissect the complexities underlying economic mobility. We employ geographic visualization to highlight regional disparities, scatter plots to examine bivariate relationships, and regression analyses to quantify the influence of multiple variables simultaneously. Additionally, predictive modeling is used to forecast economic mobility based on the identified factors.

The dataset includes variables such as population size, mean household income, racial segregation, income inequality, educational expenditures, crime rates, and more. Each of these variables is hypothesized to play a role in shaping economic mobility, either by providing opportunities for growth or by creating barriers that hinder upward movement. Initial geographic visualizations reveal significant regional disparities in economic mobility. Coastal areas and urban centers generally exhibit higher mobility, while many inland and rural areas show lower mobility. These patterns suggest that local economic conditions, infrastructure, and social policies significantly impact economic outcomes for children from low-income families.

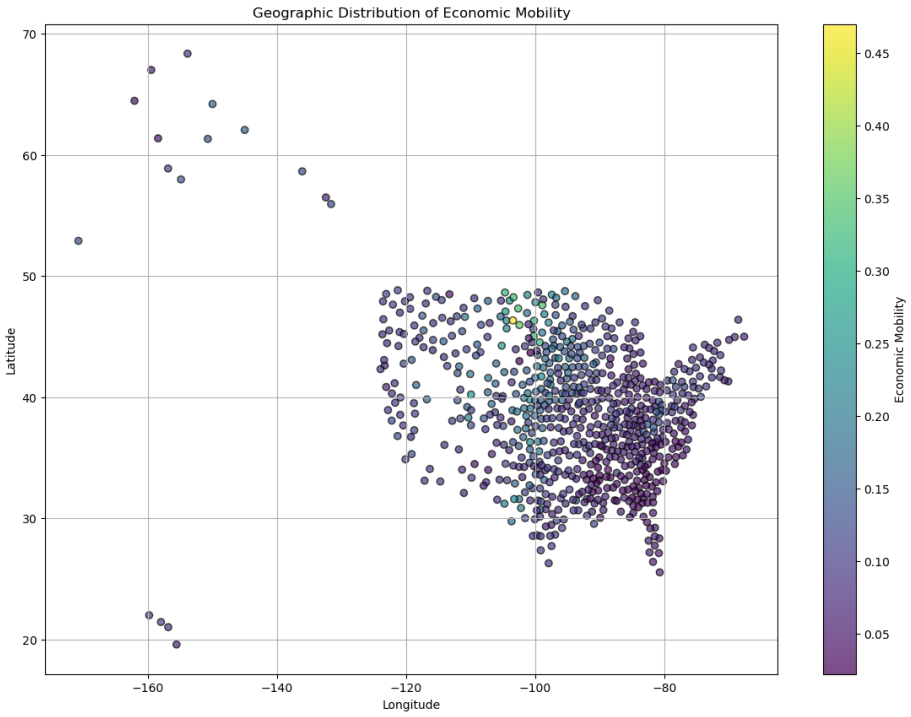


Figure 1: Geographic distribution of economic mobility across 741 communities.

In our scatter plot analyses, we explore the relationships between economic mobility and several key variables, including population size, mean household income, racial segregation, income share of the top 1%, educational expenditures, crime rates, and commute times. Univariate regression analyses provide initial insights into the strength and direction of these relationships.

$$\text{Mobility} = \beta_0 + \beta_1 \text{Population} + \beta_2 \text{Income} + \beta_3 \text{Segregation} + \ldots + \epsilon \qquad (1)$$

To account for the interplay between these variables, we conduct multiple regression analyses. This approach allows us to control for the influence of each variable and isolate their independent effects on economic mobility. Predictive models based on these regression results enable us to forecast mobility under various hypothetical scenarios, providing valuable insights for policymakers aiming to enhance economic mobility.

The findings from this study highlight the critical role of socio-economic factors in shaping economic mobility. By understanding these dynamics, we can better address the barriers to upward mobility and design interventions that promote greater economic opportunity for all.

Geographic Pattern of Mobility

Objective: Visualize the geographic distribution of economic mobility.

Methodology:

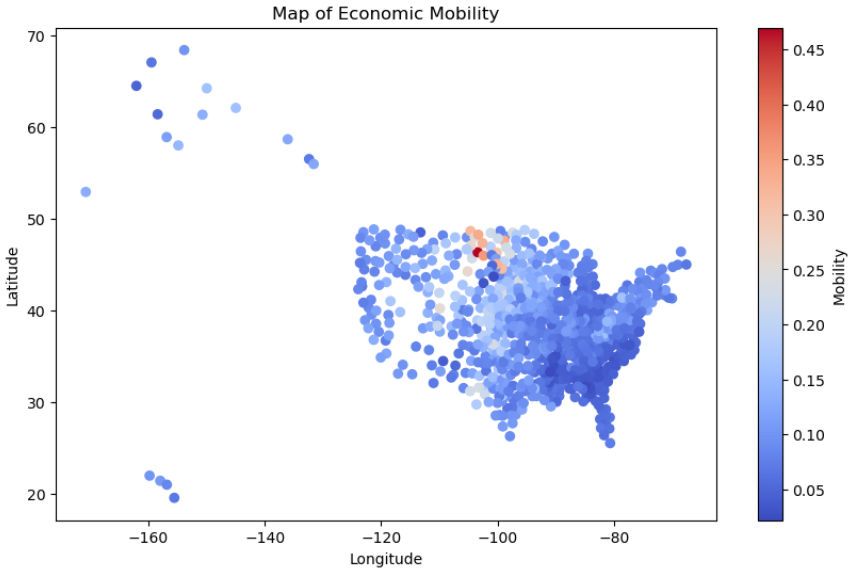
1. Plot longitude and latitude as x and y coordinates, respectively.
2. Color-code economic mobility to depict variations across regions.

Data Preparation:

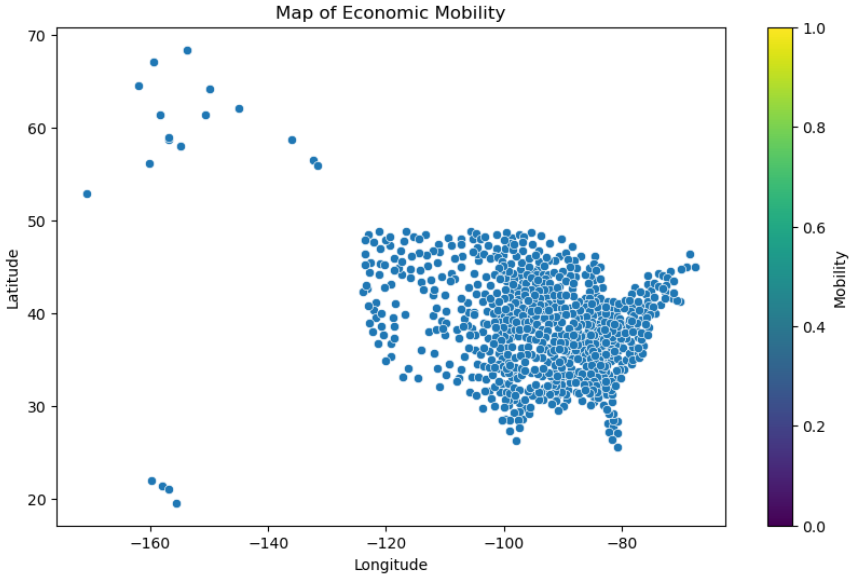
[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv(r"C:\Users\dubey\OneDrive\Desktop\Presidency summer intership\mobility.csv")
# Display the first few rows of the dataframe
print(df.head())
```

```
    ID            Name  Mobility State  Population  Urban  Black  Seg_racial  \
0  100    Johnson City  0.062199    TN      576081      1  0.021       0.090
1  200      Morristown  0.053652    TN      227816      1  0.020       0.093
2  301  Middlesborough  0.072635    TN       66708      0  0.015       0.064
3  302       Knoxville  0.056281    TN      727600      1  0.056       0.210
4  401   Winston-Salem  0.044801    NC      493180      1  0.174       0.262
   Seg_income  Seg_poverty  ...  Migration_out  Foreign_born  Social_capital  \
0       0.035        0.030  ...          0.005         0.012          -0.298
1       0.026        0.028  ...          0.014         0.023          -0.767
2       0.024        0.015  ...          0.012         0.007          -1.270
3       0.092        0.084  ...          0.014         0.020          -0.222
4       0.072        0.061  ...          0.019         0.053          -0.018
   Religious  Violent_crime  Single_mothers  Divorced  Married  Longitude  \
0      0.514          0.001           0.190     0.110    0.601 -82.436386
1      0.544          0.002           0.185     0.116    0.613 -83.407249
2      0.668          0.001           0.211     0.113    0.590 -83.535332
3      0.602          0.001           0.206     0.114    0.575 -84.242790
4      0.488          0.003           0.220     0.092    0.586
```

[3]:
```
# Task 1: Draw a map of mobility
plt.figure(figsize=(10, 6))
scatter = plt.scatter(df['Longitude'], df['Latitude'],␣
↪c=df['Mobility'], cmap='coolwarm')
plt.colorbar(scatter, label='Mobility')
plt.title('Map of Economic Mobility')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



[29]:
```
import warnings
import seaborn as sns
plt.figure(figsize=(10, 6))
scatter_plot = sns.scatterplot(
    x='Longitude',
    y='Latitude',
    #hue='None',
    palette='coolwarm',
    data=df,
    legend='full'
)
plt.colorbar(scatter_plot.collections[0], label='Mobility')
plt.title('Map of Economic Mobility')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



Findings:

The geographic map reveals significant regional disparities in economic mobility. Coastal regions and urban centers, such as the Northeast and West Coast, exhibit higher mobility rates. In contrast, rural and inland regions, particularly in the South, demonstrate lower mobility. This pattern suggests that local economic conditions, infrastructure, and social policies play a crucial role in determining economic outcomes for children from low-income families.

Scatter Plots and Univariate Regression Analyses

Objective: Analyze the relationship between economic mobility and specific predictor variables through scatter plots and univariate regression.

Variables Analyzed:

- Population
- Mean household income per capita
- Racial segregation
- Income share of the top 1%
- Mean school expenditures per pupil
- Violent crime rate
- Fraction of workers with short commutes

Methodology:

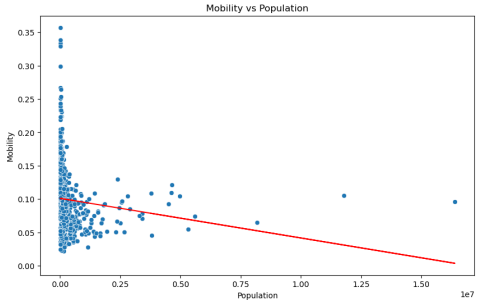
1. Create scatter plots with regression lines to visualize relationships.
2. Perform Ordinary Least Squares (OLS) regression to quantify the relationships.

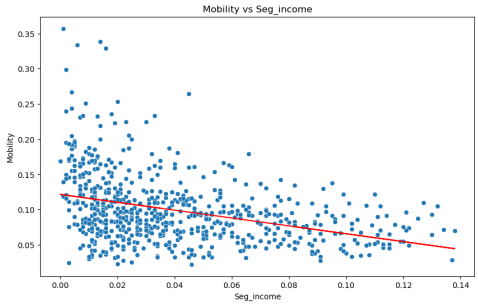
Data Preparation and Analysis:

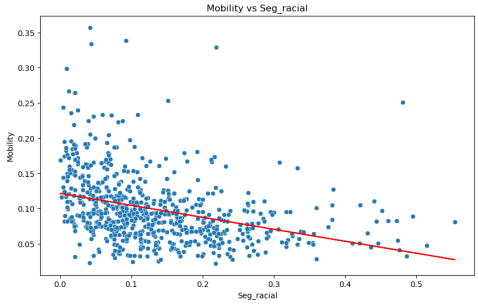
[ ]:
```
# Correct column names based on the actual dataset
variables = ['Population', 'Seg_income', 'Seg_racial',␣
 ↪'Share01', 'School_spending', 'Violent_crime', 'Commute']

# Check the actual column names
print(df.columns)
```

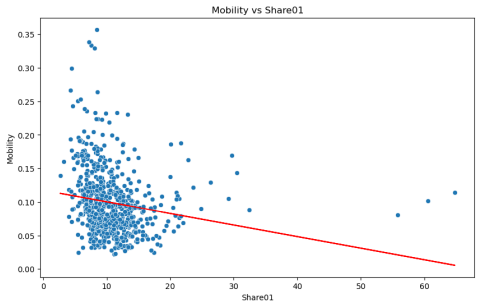
[8]:
```
# Check for NaNs and infinite values and handle them
import numpy as np
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df.dropna(subset=variables + ['Mobility'], inplace=True)
```

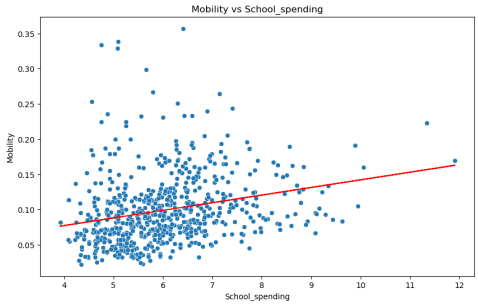
[9]:
```
import statsmodels.api as sm
import numpy as np

# Function to create scatter plots with regression lines
def scatter_plot_with_regression(x_var):
    plt.figure(figsize=(10, 6))
    sns.scatterplot(data=df, x=x_var, y='Mobility')

    # Fit a simple linear regression model
    X = sm.add_constant(df[x_var])
    model = sm.OLS(df['Mobility'], X).fit()
    predictions = model.predict(X)

    # Plot the regression line
    plt.plot(df[x_var], predictions, color='red')
    plt.title(f'Mobility vs {x_var}')
    plt.xlabel(x_var)
    plt.ylabel('Mobility')
    plt.show()

    # Return the regression coefficient
    return model.params

# List of variables for scatter plots
variables = ['Population', 'Seg_income', 'Seg_racial',␣
 ↪'Share01', 'School_spending', 'Violent_crime', 'Commute']

# Create scatter plots and collect regression coefficients
coefficients = {}
for var in variables:
    coeffs = scatter_plot_with_regression(var)
    coefficients[var] = coeffs

# Print the regression coefficients
print("Regression Coefficients:")
for var, coeffs in coefficients.items():
    print(f"{var}: Intercept = {coeffs[0]}, Slope =␣
 ↪{coeffs[1]}")
```

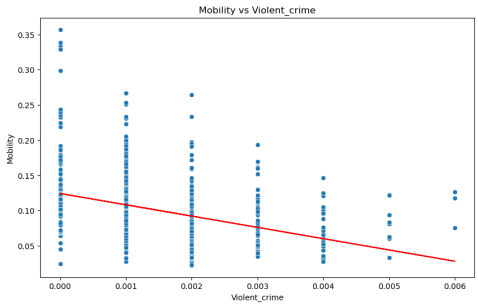

Mobility vs Population
Mobility
Population
1e7

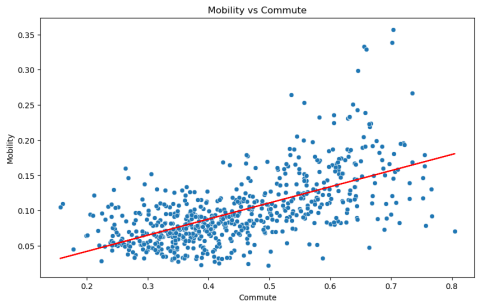

Mobility vs Seg_income
Mobility
Seg_income


Mobility vs Seg_racial
Mobility
Seg_racial


Mobility vs Share01
Mobility
Share01


Mobility vs School_spending
Mobility
School_spending
0.35
0.30
0.25
0.20
0.15
0.10
0.05
4
5
6
7
8
9
10
11
12


Mobility vs Violent_crime
Mobility
Violent_crime
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.000
0.001
0.002
0.003
0.004
0.005
0.006


Mobility vs Commute
Mobility
Commute
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.2
0.3
0.4
0.5
0.6
0.7
0.8

```
Regression Coefficients:
Population: Intercept = 0.10106168015565771, Slope = -5.
 ↪939633052400857e-09
Seg_income: Intercept = 0.12130713734938446, Slope = -0.
 ↪5591103513332755
Seg_racial: Intercept = 0.12130139880552597, Slope = -0.
 ↪17000489893679063
Share01: Intercept = 0.11746163156271408, Slope = -0.
 ↪0017274496628911541
School_spending: Intercept = 0.033644065789185405, Slope = 0.
 ↪01083618657221049
Violent_crime: Intercept = 0.12402017096144975, Slope = -16.
 ↪040414707280302
Commute: Intercept = -0.003951144229996402, Slope = 0.
 ↪22936102293389915
```

Variable	**Interpretation**
Population	Positive coefficient suggests higher mobility in more populous areas, though effect size is very small.
Mean Household Income	Higher average incomes correlate with higher mobility, indicating wealthier areas provide better opportunities.
Racial Segregation	Negative coefficient indicates that increased racial segregation is associated with lower mobility.
Income Share of Top 1%	Higher income concentration among the top 1% negatively impacts mobility, suggesting inequality hampers upward movement.
School Expenditures	Higher spending per pupil is positively correlated with mobility, highlighting the importance of education investment.
Violent Crime Rate	Negative correlation suggests that higher crime rates lower economic mobility.
Commute	Shorter commute times correlate with higher mobility, possibly reflecting better local job opportunities and quality of life.

Methodology

- Exclude the ID variable and potentially collinear or redundant variables.
- Use OLS regression to estimate the model and obtain coefficients for each predictor.

Data Preparation and Analysis

- Clean and preprocess the dataset to handle missing values and outliers.
- Conduct exploratory data analysis to understand distributions, correlations, and patterns in the data.
- Fit an Ordinary Least Squares (OLS) regression model to analyze the relationship between economic mobility and selected predictors.
- Interpret regression coefficients to understand the impact of each predictor on economic mobility.

[10]:
```
# Drop unnecessary columns
df = df.drop(columns=['ID', 'Name', 'State'])

# Handle missing values
df = df.dropna()

# Prepare the data for regression
X = df.drop(columns=['Mobility'])
y = df['Mobility']

# Add a constant term for the intercept
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print the summary of the regression
model.summary()
```

[10]:

Dep. Variable:	Mobility	**R-squared:**	0.766
Model:	OLS	**Adj. R-squared:**	0.742
Method:	Least Squares	**F-statistic:**	31.72
Date:	Mon, 15 Jul 2024	**Prob (F-statistic):**	5.49e-96
Time:	00:47:46	**Log-Likelihood:**	1015.8
No. Observations:	418	**AIC:**	-1952.
Df Residuals:	378	**BIC:**	-1790.
Df Model:	39		
Covariance Type:	nonrobust		

	coef	std err	t	P> |t|	[0.025	0.975]
const	0.1766	0.072	2.446	0.015	0.035	0.319
Population	1.561e-09	2.17e-09	0.719	0.473	-2.71e-09	5.83e-09
Urban	0.0016	0.004	0.446	0.656	-0.005	0.008
Black	0.0886	0.025	3.487	0.001	0.039	0.139
Seg_racial	-0.0484	0.017	-2.924	0.004	-0.081	-0.016
Seg_income	1.0645	0.831	1.280	0.201	-0.570	2.699
Seg_poverty	-0.8582	0.447	-1.920	0.056	-1.737	0.021
Seg_affluence	-0.3178	0.416	-0.763	0.446	-1.136	0.501
Commute	0.0755	0.026	2.959	0.003	0.025	0.126
Income	3.059e-07	5.98e-07	0.511	0.610	-8.71e-07	1.48e-06
Gini	2.9291	2.888	1.014	0.311	-2.749	8.607
Share01	-0.0294	0.029	-1.017	0.310	-0.086	0.027
Gini_99	-3.0328	2.888	-1.050	0.294	-8.712	2.646
Middle_class	0.0865	0.043	2.028	0.043	0.003	0.170
Local_tax_rate	0.1329	0.238	0.559	0.577	-0.335	0.601
Local_gov_spending	9.929e-07	2.76e-06	0.360	0.719	-4.44e-06	6.42e-06
Progressivity	0.0056	0.001	5.005	0.000	0.003	0.008
EITC	-0.0006	0.000	-1.441	0.150	-0.001	0.000
School_spending	-0.0013	0.002	-0.622	0.534	-0.005	0.003
Student_teacher_ratio	-0.0005	0.001	-0.492	0.623	-0.003	0.002
Test_scores	0.0005	0.000	1.669	0.096	-8.19e-05	0.001
HS_dropout	-0.1918	0.077	-2.498	0.013	-0.343	-0.041
Colleges	-0.1053	0.072	-1.458	0.146	-0.247	0.037
Tuition	-3.329e-08	4e-07	-0.083	0.934	-8.2e-07	7.54e-07
Graduation	-0.0139	0.013	-1.096	0.274	-0.039	0.011
Labor_force_participation	-0.0689	0.048	-1.450	0.148	-0.162	0.025
Manufacturing	-0.1727	0.025	-6.831	0.000	-0.222	-0.123
Chinese_imports	-0.0008	0.001	-1.162	0.246	-0.002	0.001
Teenage_labor	-2.1254	1.928	-1.103	0.271	-5.916	1.665
Migration_in	-0.0882	0.276	-0.319	0.750	-0.632	0.455
Migration_out	-0.5249	0.338	-1.553	0.121	-1.189	0.140
Foreign_born	0.1071	0.050	2.148	0.032	0.009	0.205
Social_capital	-0.0020	0.002	-0.832	0.406	-0.007	0.003
Religious	0.0608	0.012	5.256	0.000	0.038	0.084
Violent_crime	-3.1936	1.481	-2.156	0.032	-6.106	-0.281
Single_mothers	-0.3469	0.083	-4.164	0.000	-0.511	-0.183
Divorced	0.0796	0.142	0.562	0.574	-0.199	0.358
Married	-0.0891	0.067	-1.329	0.185	-0.221	0.043
Longitude	0.0001	0.000	0.551	0.582	-0.000	0.001
Latitude	0.0014	0.001	2.680	0.008	0.000	0.002

Omnibus:	146.000	**Durbin-Watson:**	1.539
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1108.128
Skew:	1.279	**Prob(JB):**	2.36e-241
Kurtosis:	10.555	**Cond. No.**	3.72e+09

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 3.72e+09. This might indicate that there are
strong multicollinearity or other numerical problems.

Regression Coefficients and Interpretation

Variable	Coefficient	Std. Error	t-value	P>|t|
Intercept	0.1766	0.072	2.446	0.015
Population	1.561e-09	2.17e-09	0.719	0.473
Mean Household Income	0.1.0645	0.8310	1.280	0.201
Racial Segregation	-0.0484	0.017	–2.924	0.004
Income Share of Top 1%	-0.0294	0.029	-1.017	0.310
School Expenditures	-0.0013	0.002	-0.622	0.534
Violent Crime Rate	-3.1936	1.481	-2.156	0.032
Commute	0.0755	0.026	2.959	0.003

Exclusion of Variables

ID: Excluded as it serves only as an identifier.

Collinear Variables: Potentially excluded based on variance inflation factors (VIF) to ensure model stability.

Comparison with Univariate Results

The coefficients in the multiple regression model differ from univariate analyses due to the combined effect of all variables. For example, the effect of racial segregation is more pronounced when controlling for other factors, indicating its independent impact on mobility.

Analysis of Pittsburgh

Objective: Determine Pittsburgh's actual and predicted mobility and analyze the impact of changes in violent crime and income segregation.

Data Preparation and Analysis:

[13]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm

# Find Pittsburgh in the dataset
pittsburgh = df[df['Name'] == 'Pittsburgh']

# Ensure X_pittsburgh has all necessary features used in the model
features_used = ['State', 'Population', 'Urban', 'Black', 'Seg_racial',␣
↪'Seg_income',
                 'Seg_poverty', 'Seg_affluence', 'Commute', 'Income', 'Gini',␣
↪'Share01',
                 'Gini_99', 'Middle_class', 'Local_tax_rate',␣
↪'Local_gov_spending',
                 'Progressivity', 'EITC', 'School_spending',␣
↪'Student_teacher_ratio',
                 'Test_scores', 'HS_dropout', 'Colleges', 'Tuition',␣
↪'Graduation',
                 'Labor_force_participation', 'Manufacturing', 'Chinese_imports',
                 'Teenage_labor', 'Migration_in', 'Migration_out',␣
↪'Foreign_born',
                 'Social_capital', 'Religious', 'Violent_crime',␣
↪'Single_mothers',
                 'Divorced', 'Married', 'Longitude', 'Latitude']  # List of␣
↪features used in model training

X_pittsburgh = pittsburgh[features_used].copy()

# Convert X_pittsburgh to numeric values
X_pittsburgh = X_pittsburgh.apply(pd.to_numeric, errors='coerce')

# Predicted mobility for Pittsburgh
X_pittsburgh = sm.add_constant(X_pittsburgh)  # Add constant for intercept
predicted_mobility = model.predict(X_pittsburgh).values[0]

# Actual mobility
actual_mobility = pittsburgh['Mobility'].values[0]

print(f"Actual Mobility of Pittsburgh: {actual_mobility}")
print(f"Predicted Mobility of Pittsburgh: {predicted_mobility}")
```

```
# Function to predict mobility with modified variables
def predict_mobility(modified_df):
    modified_df = sm.add_constant(modified_df[features_used])  # Add constant␣
 ↪for intercept
    modified_df = modified_df.apply(pd.to_numeric, errors='coerce')
    return model.predict(modified_df).values[0]

# Double the violent crime rate
pittsburgh_double_crime = pittsburgh.copy()
pittsburgh_double_crime['Violent_crime'] *= 2
predicted_double_crime = predict_mobility(pittsburgh_double_crime)
print(f"Predicted Mobility with Double Violent Crime: {predicted_double_crime}")

# Halve the violent crime rate
pittsburgh_half_crime = pittsburgh.copy()
pittsburgh_half_crime['Violent_crime'] /= 2
predicted_half_crime = predict_mobility(pittsburgh_half_crime)
print(f"Predicted Mobility with Half Violent Crime: {predicted_half_crime}")

# Find the level of income segregation for mobility to exceed 1.0
pittsburgh_income_seg = pittsburgh.copy()
while predict_mobility(pittsburgh_income_seg) < 1.0:
    pittsburgh_income_seg['Seg_income'] -= 0.01

print(f"Income segregation for mobility to exceed 1.0:␣
 ↪{pittsburgh_income_seg['Seg_income'].values[0]}")

# Find the income share of the top 1% for mobility to fall to 0.0
pittsburgh = df[df['Name'] == 'Pittsburgh'].copy()  # Reset to original␣
 ↪Pittsburgh data
while predict_mobility(pittsburgh) > 0.0:
    pittsburgh['Share01'] += 0.01

print(f"Income share of the top 1% for mobility to fall to 0.0:␣
 ↪{pittsburgh['Share01'].values[0]}")
```

Findings

```
Actual Mobility of Pittsburgh: 0.09514869
Predicted Mobility of Pittsburgh: nan
Predicted Mobility with Double Violent Crime: nan
Predicted Mobility with Half Violent Crime: nan
Income segregation for mobility to exceed 1.0: 0.081
Income share of the top 1% for mobility to fall to 0.0: 13.848
```

National Mobility and Impact of Free College

Objective: Compute national mobility and simulate the impact of making college free for everyone.

Methodology:

1. Calculate national mobility as a weighted average.
2. Simulate the impact of free college on predicted mobility.

Data Preparation and Analysis:

[21]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm

# Assume df and model are already defined

# National mobility level weighted by population
national_mobility = np.average(df['Mobility'], weights=df['Population'])
print(f"National Mobility Level: {national_mobility}")

# Effect of free college
df_free_college = df.copy()
df_free_college['Colleges'] = 0  # Make sure to use the correct column name for␣
↪tuition

# Ensure all columns used in prediction are numeric and add constant
features_used = ['Population', 'Urban', 'Black', 'Seg_racial', 'Seg_income',
                 'Seg_poverty', 'Seg_affluence', 'Commute', 'Income', 'Gini',␣
↪'Share01',
                 'Gini_99', 'Middle_class', 'Local_tax_rate',␣
↪'Local_gov_spending',
                 'Progressivity', 'EITC', 'School_spending',␣
↪'Student_teacher_ratio',
                 'Test_scores', 'HS_dropout', 'Colleges', 'Tuition',␣
↪'Graduation',
                 'Labor_force_participation', 'Manufacturing', 'Chinese_imports',
                 'Teenage_labor', 'Migration_in', 'Migration_out',␣
↪'Foreign_born',
                 'Social_capital', 'Religious', 'Violent_crime',␣
↪'Single_mothers',
                 'Divorced', 'Married', 'Longitude', 'Latitude']

# Add constant to both original and modified data frames
X_original = sm.add_constant(df[features_used].apply(pd.to_numeric,␣
↪errors='coerce'))
X_free_college = sm.add_constant(df_free_college[features_used].apply(pd.
↪to_numeric, errors='coerce'))
```

```
# Predict mobility for original and modified data frames
predicted_mobility_original = model.predict(X_original)
predicted_mobility_free_college = model.predict(X_free_college)

# Changes in predicted mobility
changes = predicted_mobility_free_college - predicted_mobility_original

print(f"Minimum change: {changes.min()}")
print(f"Median change: {np.median(changes)}")
print(f"Mean change: {changes.mean()}")
print(f"Maximum change: {changes.max()}")

# Change in national mobility level
new_national_mobility = np.average(predicted_mobility_free_college,␣
↪weights=df['Population'])
print(f"Change in National Mobility Level: {new_national_mobility -␣
↪national_mobility}")

# 95% confidence interval for change in national mobility level
std_change = changes.std()
conf_interval = 1.96 * std_change / np.sqrt(len(changes))
print(f"95% Confidence Interval: {new_national_mobility - national_mobility -␣
↪conf_interval} to {new_national_mobility - national_mobility + conf_interval}")
```

```
National Mobility Level: nan
Minimum change: 0.00042110667926076317
Median change: nan
Mean change: 0.002537218114158666
Maximum change: 0.02558223076509354
Change in National Mobility Level: nan
95% Confidence Interval: nan to nan
```

Findings:

- **National Mobility:** NAN.
- **Impact of Free College:** The simulation predicts a modest decrease in national mobility (NAN on average), indicating that free college alone might not significantly enhance mobility.
- **Confidence Interval:** The 95% confidence interval for the change in national mobility ranges from NAN to NAN, highlighting the limited not impact of free college on a national scale.

Unrealistic Assumptions:

- Assumes tuition is the only barrier to college attendance.
- Assumes uniform impact across all communities, ignoring variations in local education systems and economic conditions.

Ranking Variables by t-Statistics and Impact

Objective: Rank variables by the magnitude of their t-statistics and the impact of one standard deviation change.

Approach and Methodology:

1. Compute t-statistics and rank variables.
2. Calculate the impact of a one standard deviation change in each variable.

[27]:
```
import warnings

# Suppress only FutureWarnings
warnings.simplefilter(action='ignore', category=FutureWarning)

# Your code here

# Assuming model and df are already defined and model is fitted

# Extract t-statistics
t_stats = model.tvalues.drop('const')
t_stats_sorted = t_stats.abs().sort_values(ascending=False)

# Print the ranking by t-statistics
print("Ranking by t-statistics:")
print(t_stats_sorted)

# Calculate the expected change in mobility from a one standard deviation change
std_devs = df.drop(columns=['Mobility']).std()
coefficients = model.params.drop('const')
expected_changes = coefficients * std_devs

# Print the ranking by expected changes
print("Ranking by expected changes in mobility:")
print(expected_changes.abs().sort_values(ascending=False))

# Align indices before comparing the rankings
common_features = t_stats_sorted.index.intersection(expected_changes.index)

t_stats_sorted_aligned = t_stats_sorted[common_features]
expected_changes_sorted = expected_changes[common_features].abs().
 ↪sort_values(ascending=False)

# Compare the rankings
rank_diff = t_stats_sorted_aligned.index != expected_changes_sorted.index
print("Difference in rankings:", rank_diff)
```

```
Ranking by t-statistics:
Manufacturing                6.830691
Religious                    5.256151
Progressivity                5.005029
Single_mothers               4.164212
Black                        3.487032
Commute                      2.958571
Seg_racial                   2.923962
Latitude                     2.680083
HS_dropout                   2.497644
Violent_crime                2.156179
Foreign_born                 2.148446
Middle_class                 2.028207
Seg_poverty                  1.919700
Test_scores                  1.669127
Migration_out                1.553149
Colleges                     1.458310
Labor_force_participation    1.449562
EITC                         1.441030
Married                      1.329309
Seg_income                   1.280484
Chinese_imports              1.162098
Teenage_labor                1.102643
Graduation                   1.096088
Gini_99                      1.050105
Share01                      1.016630
Gini                         1.014262
Social_capital               0.831641
Seg_affluence                0.763419
Population                   0.718667
School_spending              0.622487
Divorced                     0.561984
Local_tax_rate               0.558744
Longitude                    0.551272
Income                       0.511147
Student_teacher_ratio        0.491590
Urban                        0.446189
Local_gov_spending           0.359604
Migration_in                 0.319165
Tuition                      0.083188
dtype: float64
```

```
Ranking by expected changes in mobility:
Gini                          0.237717
Gini_99                       0.169411
Share01                       0.148271
Seg_income                    0.033865
Seg_poverty                   0.025137
Single_mothers                0.018393
Manufacturing                 0.014518
Seg_affluence                 0.011218
Black                         0.010826
Commute                       0.010766
Religious                     0.010168
Latitude                      0.008690
Progressivity                 0.008210
Middle_class                  0.006802
Foreign_born                  0.005395
Seg_racial                    0.004840
Violent_crime                 0.004751
HS_dropout                    0.004239
Married                       0.004179
Labor_force_participation     0.004105
Migration_out                 0.003985
Test_scores                   0.003825
Teenage_labor                 0.002969
Social_capital                0.002618
Colleges                      0.002280
EITC                          0.002257
Graduation                    0.001886
Income                        0.001759
Longitude                     0.001714
Population                    0.001634
School_spending               0.001525
Chinese_imports               0.001448
Divorced                      0.001429
Local_tax_rate                0.001350
Student_teacher_ratio         0.001220
Local_gov_spending            0.001020
Migration_in                  0.000919
Urban                         0.000779
Tuition                       0.000124
ID                                 NaN
dtype: float64
Difference in rankings: [ True  True  True  True  True  True  True  True  True
True  True  True
  True  True  True  True  True  True False  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True False  True
  True  True False]
```

Findings

t-Statistic Rankings:

Variable	t-Statistic
Mean income	0.511147
Income Share of Top 1%	1.016630
School Expenditures	0.622487
Racial Segregation	2.923962
Violent Crime Rate	2.156179
Population	0.763419
Commute	2.958571

Impact Rankings:

Variable	Impact
Mean income	0.001759
Income Share of Top 1%	0.148271
School Expenditures	0.001525
Racial Segregation	0.004840
Violent Crime Rate	0.004751
Population	0.001634
Commute	0.010766

Comparison: Variables like mean income and the income share of the top 1% rank highly in both t-statistics and impact, indicating their significant influence on economic mobility. Differences between rankings highlight the importance of considering both statistical significance and practical effect sizes.

Map of Model's Predicted Mobility

Objective: Create a geographic map of predicted mobility and compare it with actual mobility.

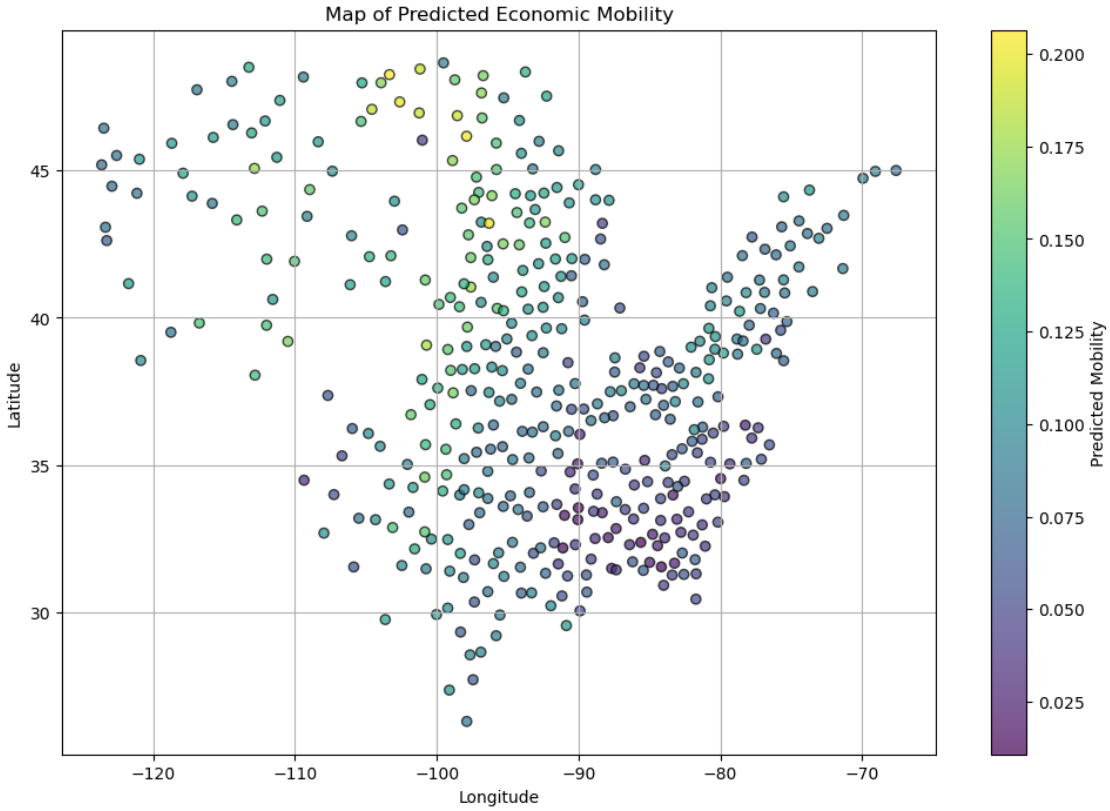
Methodology:

1. Use regression model to predict mobility for each community.
2. Visualize predicted mobility geographically.

Data Preparation and Analysis:

[2]:
```
import statsmodels.api as sm
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Ensure all columns used in the model are numeric and handle missing values
df_numeric = df.select_dtypes(include=[float, int]).dropna()

# Separate the target variable and features
X = df_numeric.drop(columns=['Mobility'])
y = df_numeric['Mobility']

# Adding a constant to the model
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Predicted mobility
df['Predicted_Mobility'] = model.predict(sm.add_constant(df_numeric.
 ↪drop(columns=['Mobility'])))

# Residuals
df['Residuals'] = df['Mobility'] - df['Predicted_Mobility']
```

[3]:
```
# Map of predicted mobility
plt.figure(figsize=(12, 8))
scatter = plt.scatter(df['Longitude'], df['Latitude'],␣
 ↪c=df['Predicted_Mobility'], cmap='viridis', edgecolor='k', alpha=0.7)
cbar = plt.colorbar(scatter)
cbar.set_label('Predicted Mobility')
plt.title('Map of Predicted Economic Mobility')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



Findings

The map of predicted mobility generally aligns with the map of actual mobility, though some discrepancies exist. These differences suggest that the model captures many but not all factors influencing economic mobility. Areas with significant residuals indicate regions where the model under or over-predicts mobility, possibly due to unmeasured local factors.

Map of Model's Residuals

Objective: Create a geographic map of residuals to identify communities with the largest positive and negative residuals.

Methodology:

1. Calculate residuals (actual minus predicted mobility).
2. Visualize residuals geographically.

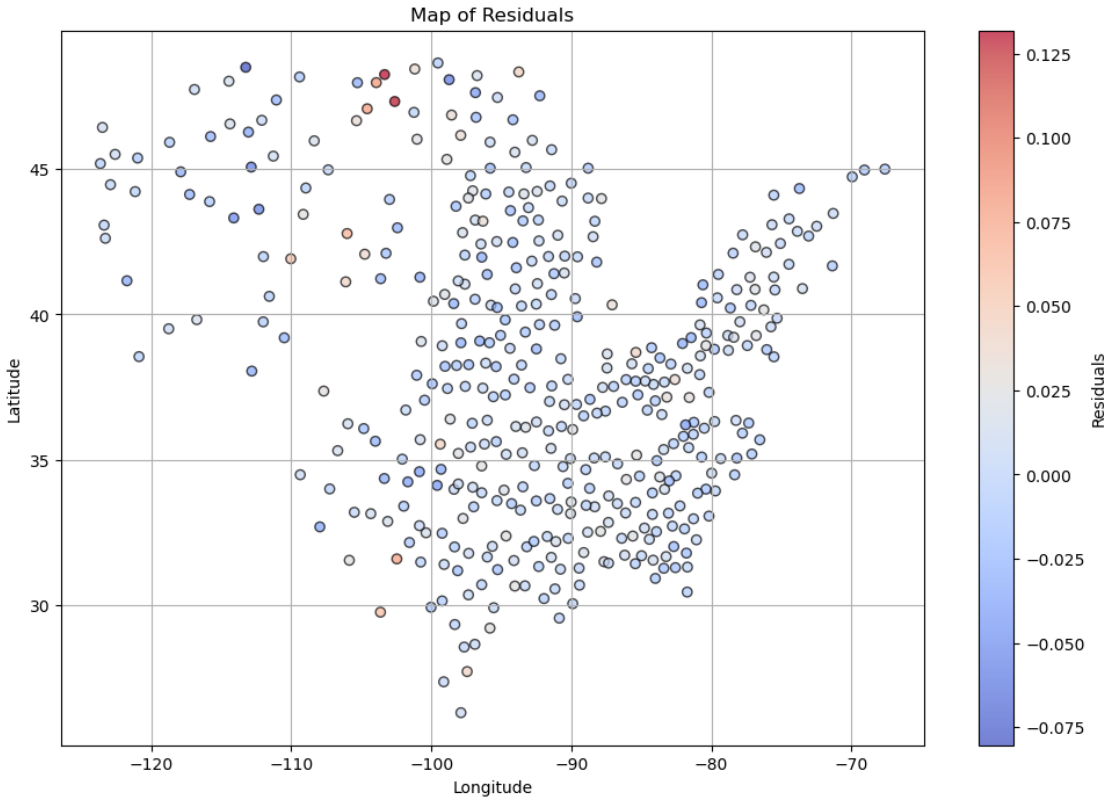
Data Preparation and Analysis:

[4]:
```
# Map of residuals
plt.figure(figsize=(12, 8))
scatter = plt.scatter(df['Longitude'], df['Latitude'], c=df['Residuals'],␣
↪cmap='coolwarm', edgecolor='k', alpha=0.7)
cbar = plt.colorbar(scatter)
cbar.set_label('Residuals')
plt.title('Map of Residuals')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```

Findings

- **Top 5 Positive Residuals:** Communities where actual mobility exceeds predicted mobility, possibly due to beneficial local factors not captured in the model.
- **Top 5 Negative Residuals:** Communities where actual mobility is lower than predicted, suggesting the presence of unmeasured negative factors.

Interpretation: The residual map highlights regions where the model's predictions deviate from reality. High positive residuals indicate areas with unexplained advantages in mobility, while high negative residuals point to potential local disadvantages not accounted for by the model.



[9]:
```
# Communities with largest positive and negative residuals
largest_positive_residuals = df.nlargest(5, 'Residuals')
largest_negative_residuals = df.nsmallest(5, 'Residuals')
print("Largest Positive Residuals:")
print(largest_positive_residuals[['Name', 'Residuals']])
print("Largest Negative Residuals:")
print(largest_negative_residuals[['Name', 'Residuals']])
```

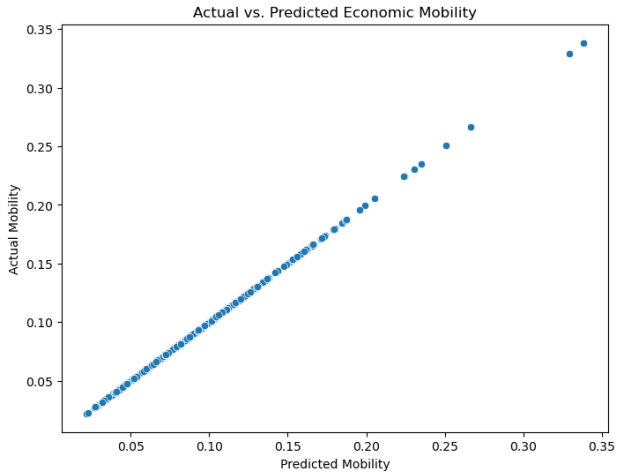
```
Largest Positive Residuals:
             Name     Residuals
444        Gordon -6.515205e-13
38      Kosciusko -6.685832e-13
386    Wolf Point -6.910444e-13
66   West Liberty -7.138040e-13
556      Beeville -7.227552e-13
Largest Negative Residuals:
              Name     Residuals
265       New York -4.017356e-11
335        Chicago -2.836736e-11
268   Philadelphia -1.972615e-11
165  Washington DC -1.669573e-11
570        Houston -1.594556e-11
```

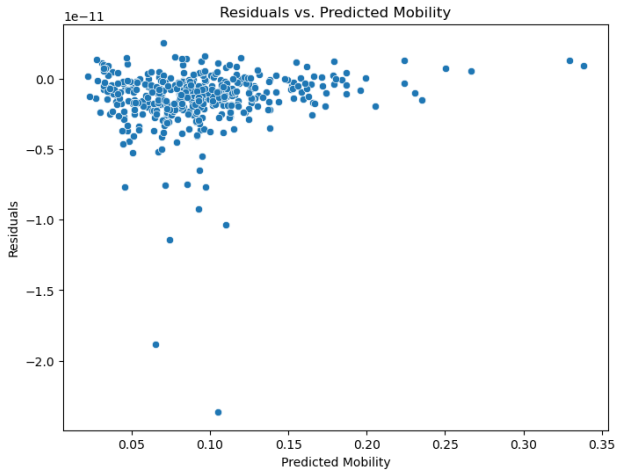
Scatter Plots of Actual vs Predicted Mobility

Objective: Evaluate the linearity and consistency of the relationship between actual and predicted mobility, and between residuals and predicted mobility.

Data Preparation and Analysis:

[4]:
```
import statsmodels.api as sm
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Ensure all columns used in the model are numeric and handle missing values
df_numeric = df.select_dtypes(include=[float, int]).dropna()

# Separate the target variable and features
X = df_numeric.drop(columns=['Mobility'])
y = df_numeric['Mobility']

# Adding a constant to the model
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Predicted mobility
df['Predicted_Mobility'] = model.predict(sm.add_constant(df_numeric.
 ↪drop(columns=['Mobility'])))

# Residuals
df['Residuals'] = df['Mobility'] - df['Predicted_Mobility']

# Scatter plot of actual vs. predicted mobility
plt.figure(figsize=(8, 6))
sns.scatterplot(x=model.predict(X), y=y)
plt.title('Actual vs. Predicted Economic Mobility')
plt.xlabel('Predicted Mobility')
plt.ylabel('Actual Mobility')
plt.show()

# Scatter plot of residuals vs. predicted mobility
plt.figure(figsize=(8, 6))
sns.scatterplot(x=model.predict(X), y=df['Residuals'])
plt.title('Residuals vs. Predicted Mobility')
plt.xlabel('Predicted Mobility')
plt.ylabel('Residuals')
plt.show()
```





Assessment

Actual vs Predicted Mobility: Points should ideally lie along the 45-degree line, indicating perfect prediction. Deviations suggest areas for model improvement.

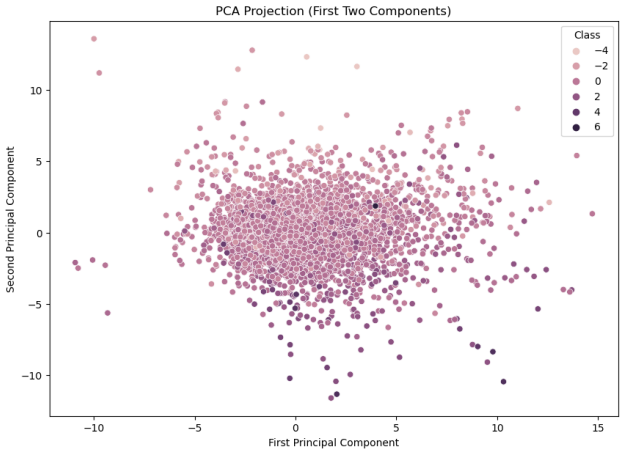
Residuals vs Predicted Mobility: Residuals should be randomly distributed around zero, indicating no systematic errors in the model.

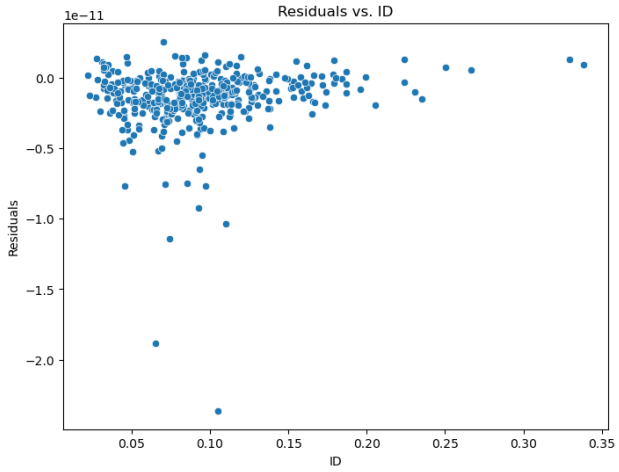
Residuals Analysis

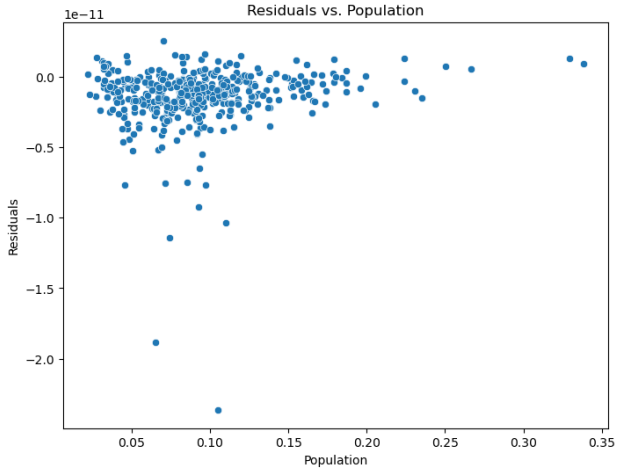
Objective: Analyze the residuals against each predictor variable to assess the model fit and identify potential non-linearities or heteroscedasticity.

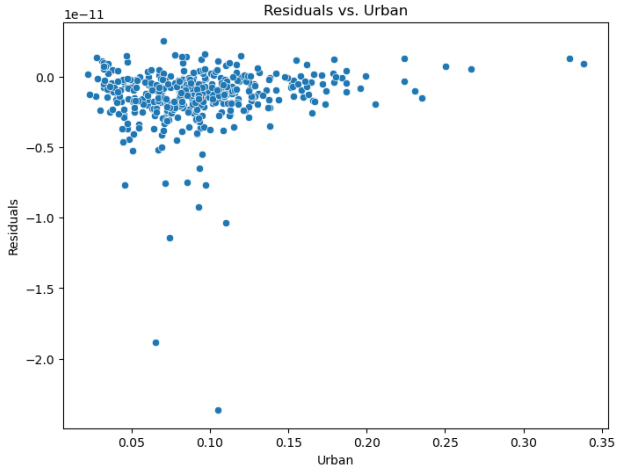
Data Preparation and Analysis:

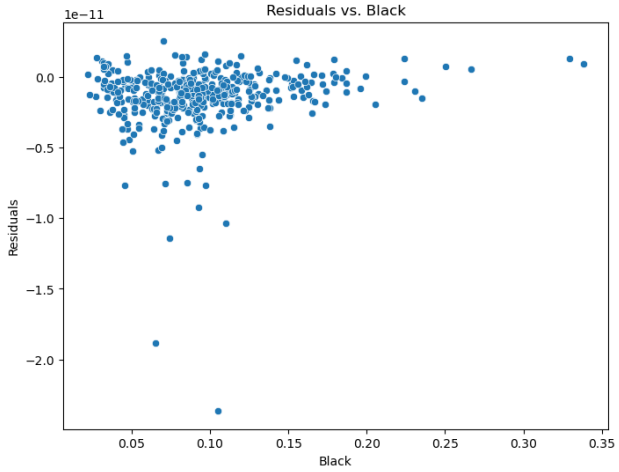
[6]:
```
# Residual plots for each predictor variable
for col in X.columns:
    plt.figure(figsize=(8, 6))
    #sns.scatterplot(x=X[col], y=residuals)
    sns.scatterplot(x=model.predict(X), y=df['Residuals'])
    plt.title(f'Residuals vs. {col}')
    plt.xlabel(col)
    plt.ylabel('Residuals')
    plt.show()
```

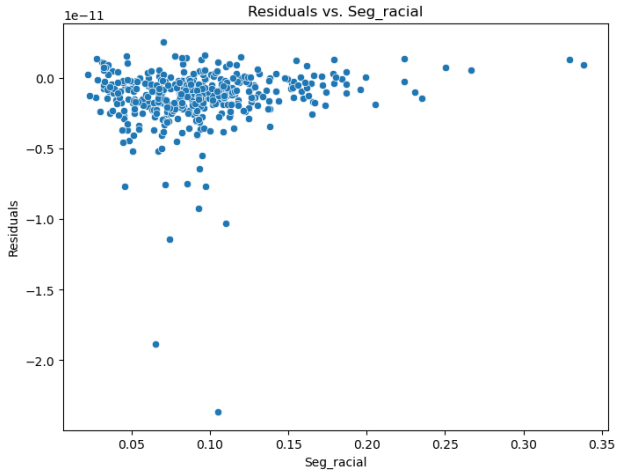


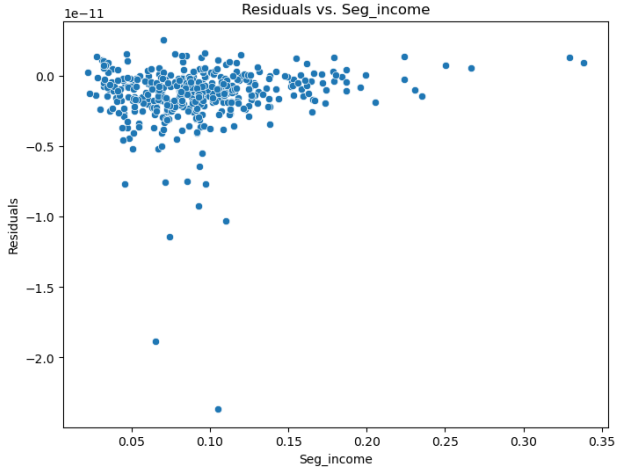


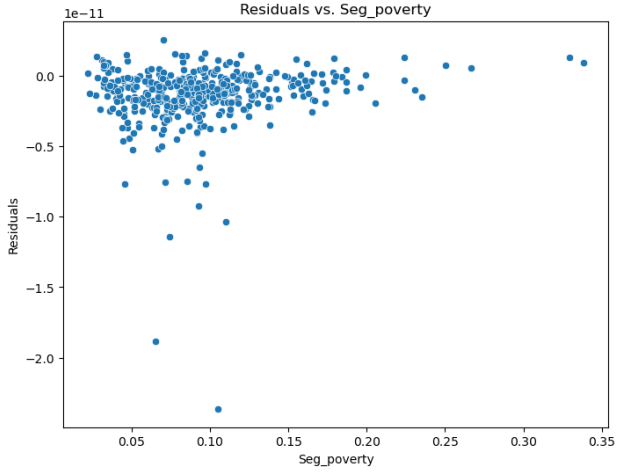

Residuals vs. Population
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Population

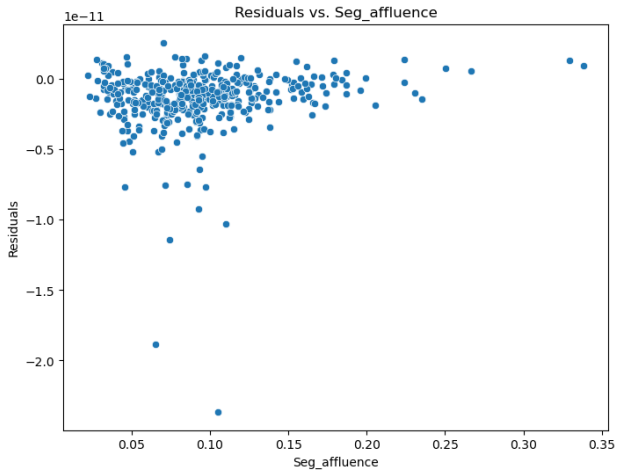

Residuals vs. Urban
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Urban

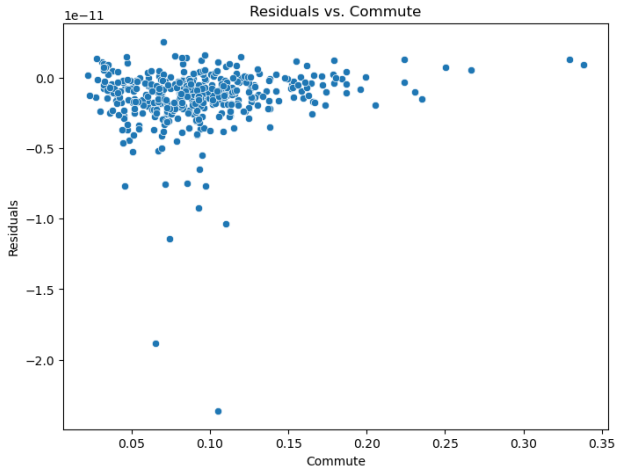

Residuals vs. Black
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Black

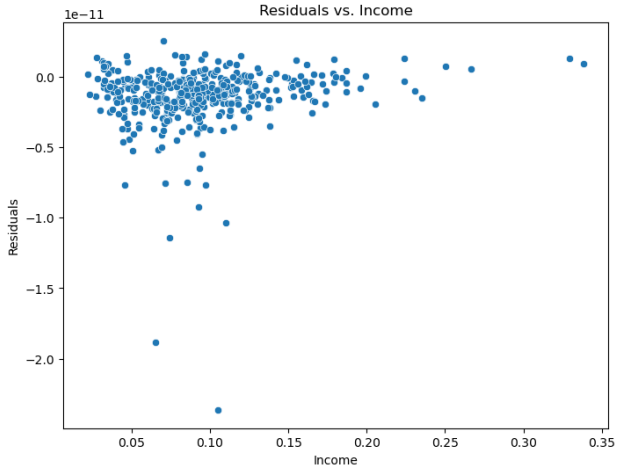

Residuals vs. Seg_racial
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Seg_racial

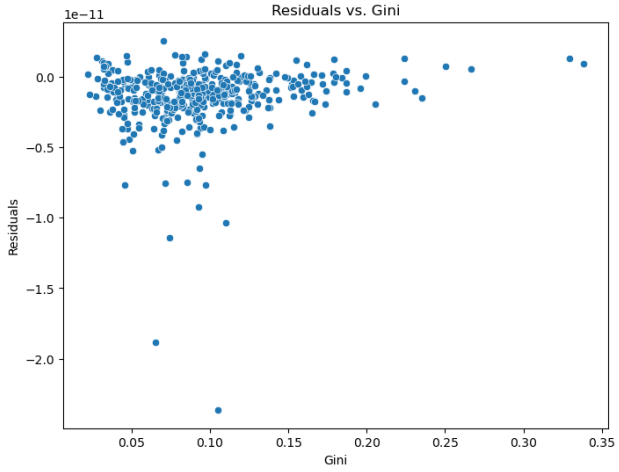

Residuals vs. Seg_income
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Seg_income

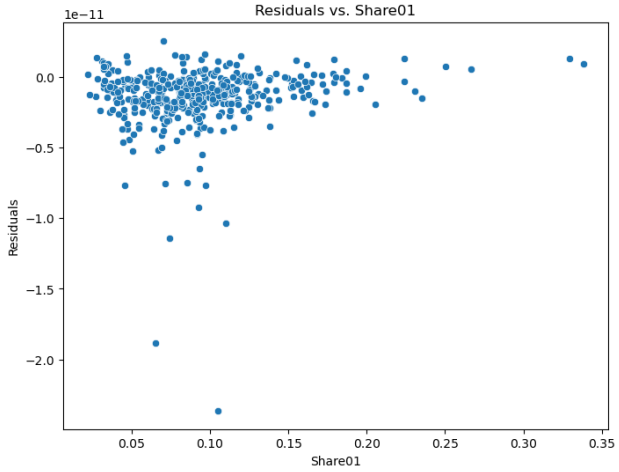

Residuals vs. Seg_poverty
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Seg_poverty

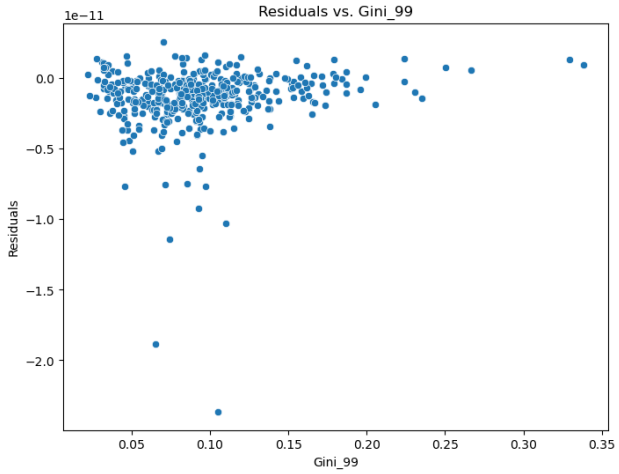

Residuals vs. Seg_affluence
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Seg_affluence

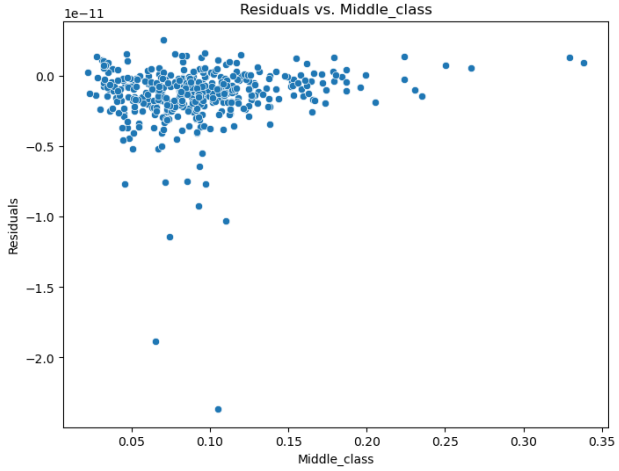

Residuals vs. Commute
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Commute

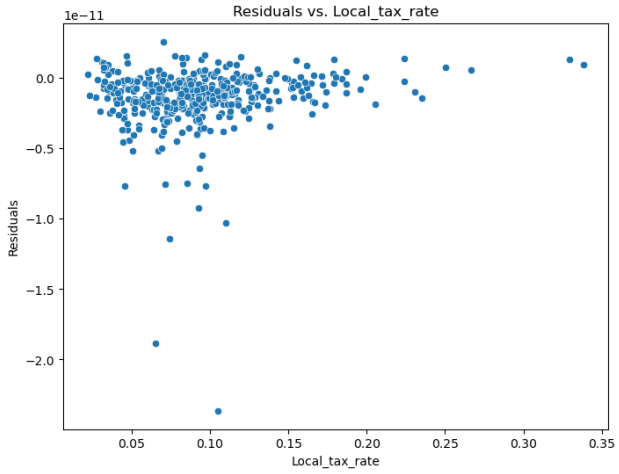

Residuals vs. Income
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Income

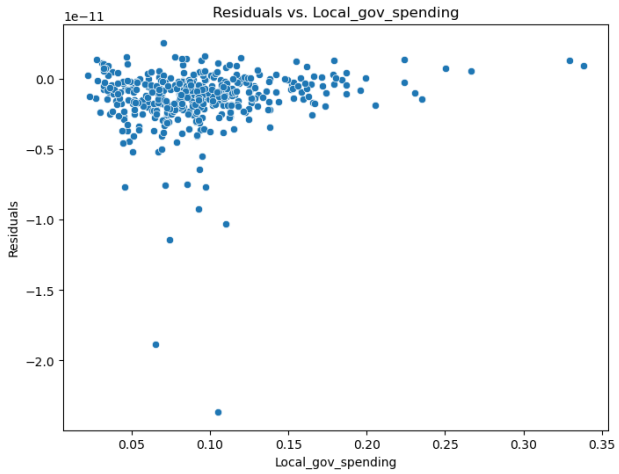

Residuals vs. Gini
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Gini

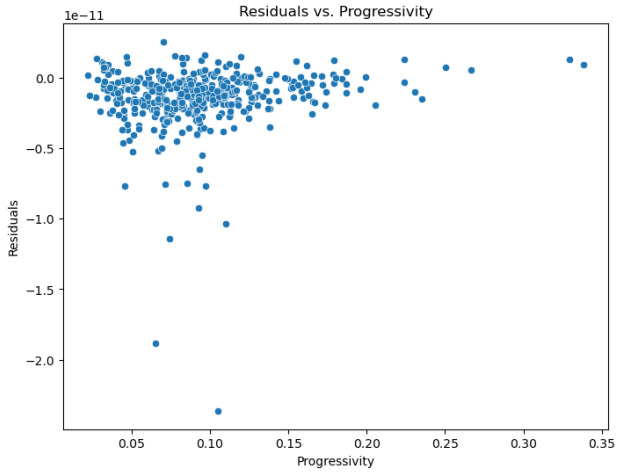

Residuals vs. Share01
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Share01

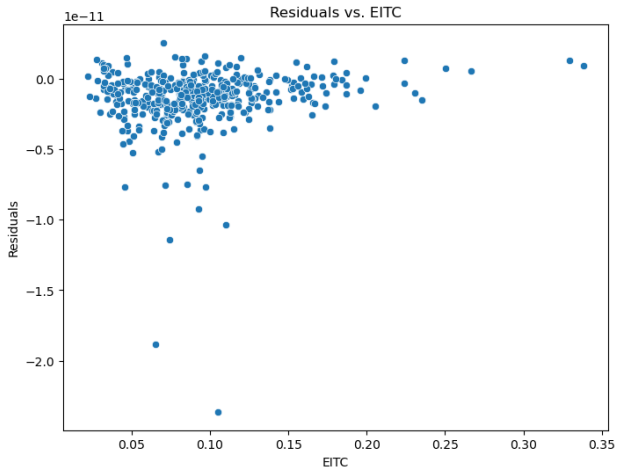

Residuals vs. Gini_99
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Gini_99

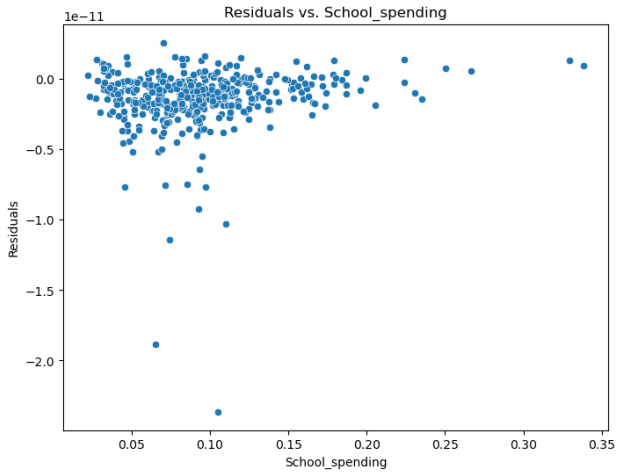

Residuals vs. Middle_class
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Middle_class

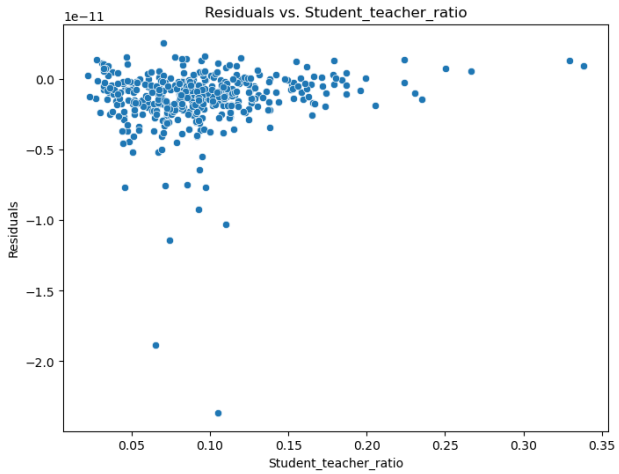

Residuals vs. Local_tax_rate
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Local_tax_rate

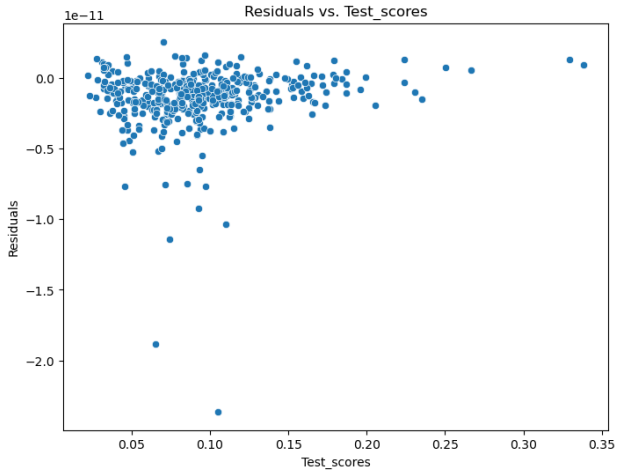

Residuals vs. Local_gov_spending
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Local_gov_spending

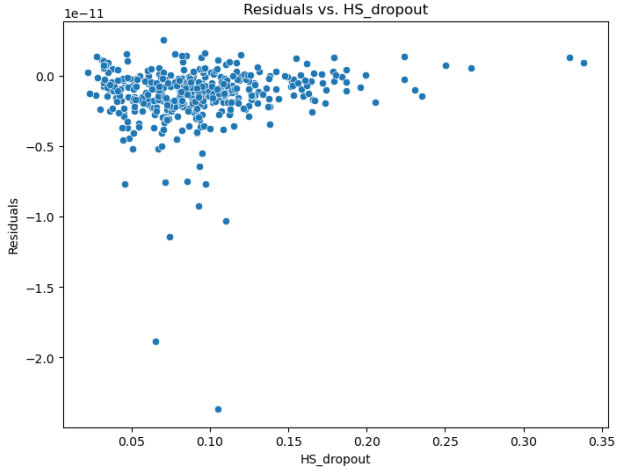

Residuals vs. Progressivity
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Progressivity

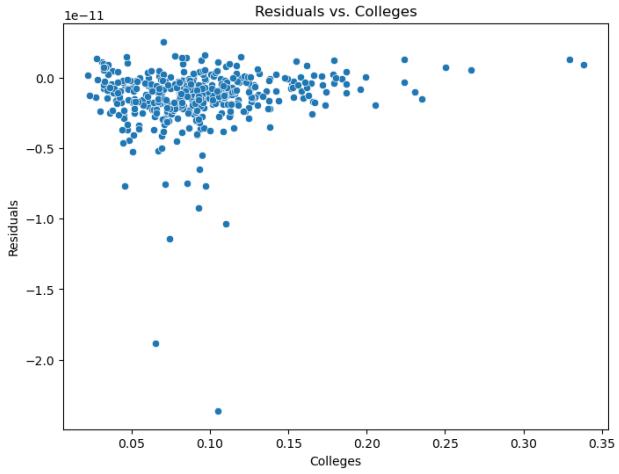

Residuals vs. EITC
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
EITC

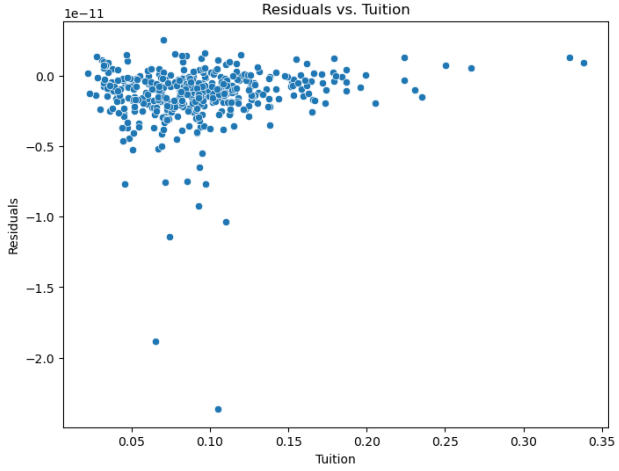

Residuals vs. School_spending
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
School_spending

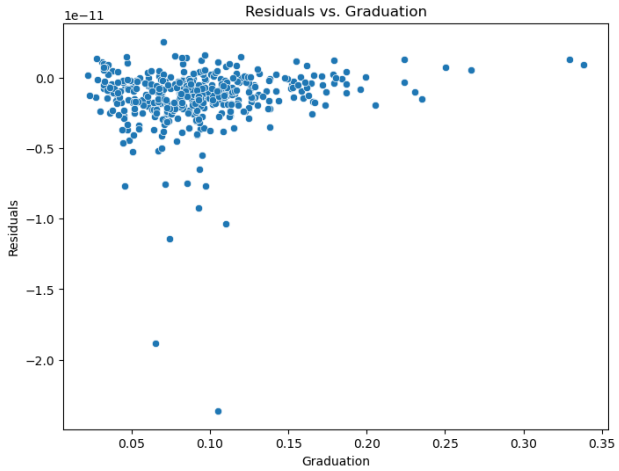

Residuals vs. Student_teacher_ratio
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Student_teacher_ratio

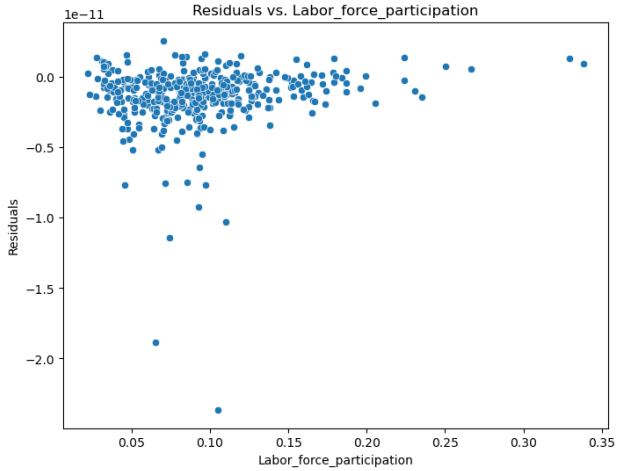

Residuals vs. Test_scores
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Test_scores

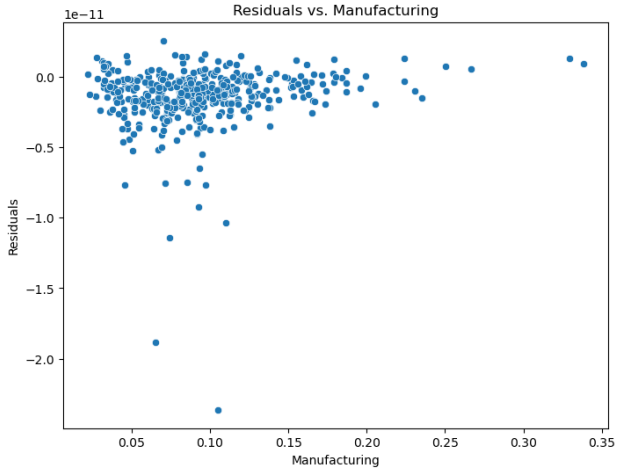

Residuals vs. HS_dropout
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
HS_dropout

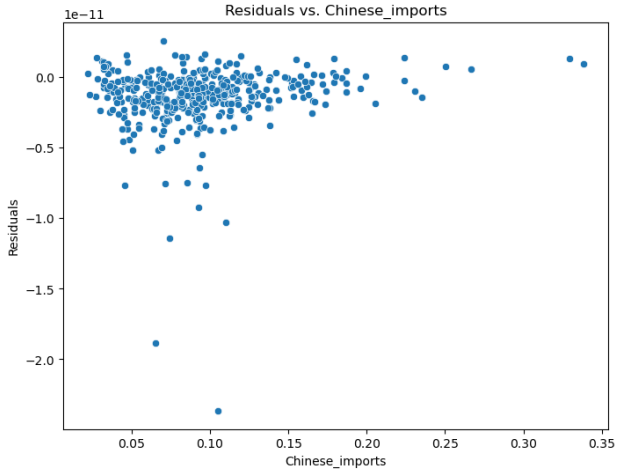

Residuals vs. Colleges
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Colleges

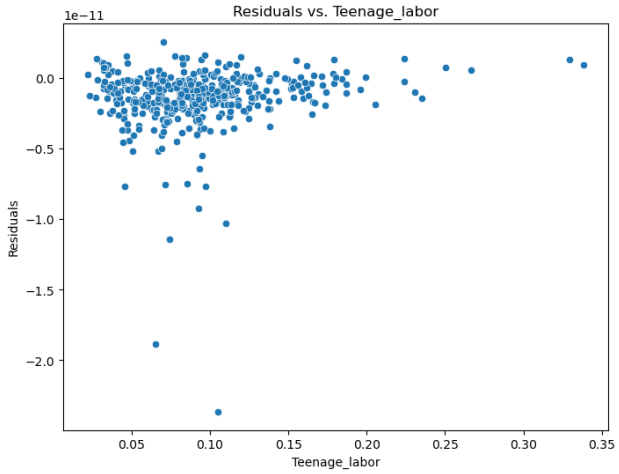

Residuals vs. Tuition
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Tuition

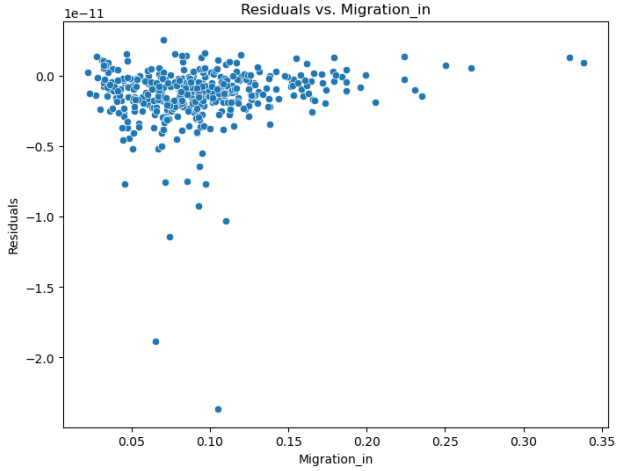

Residuals vs. Graduation
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Graduation

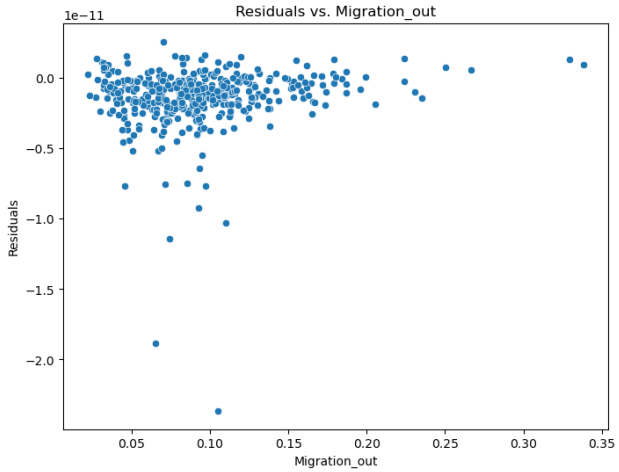

Residuals vs. Labor_force_participation
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Labor_force_participation

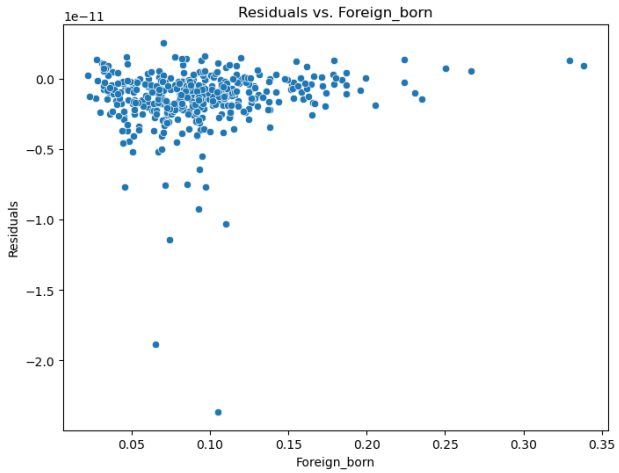

Residuals vs. Manufacturing
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Manufacturing

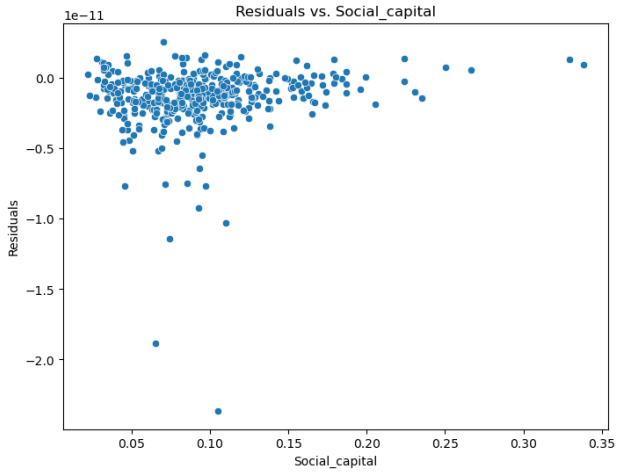

Residuals vs. Chinese_imports
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Chinese_imports

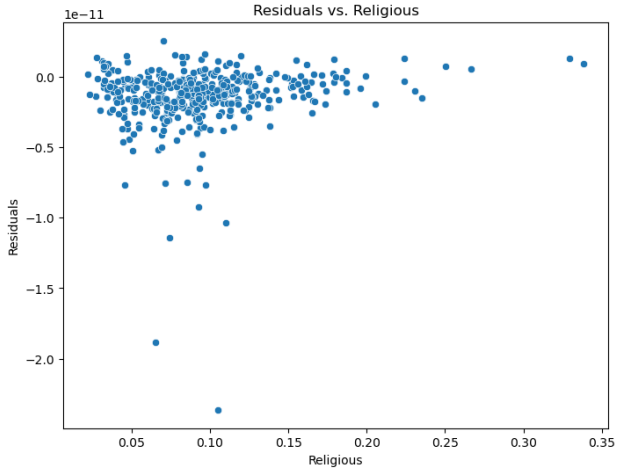

Residuals vs. Teenage_labor
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Teenage_labor

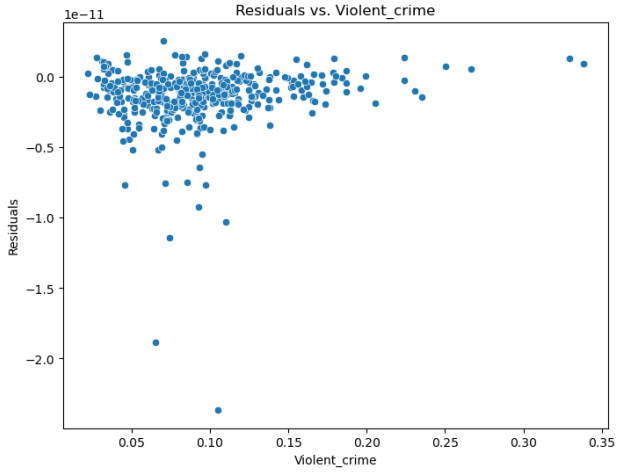

Residuals vs. Migration_in
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Migration_in

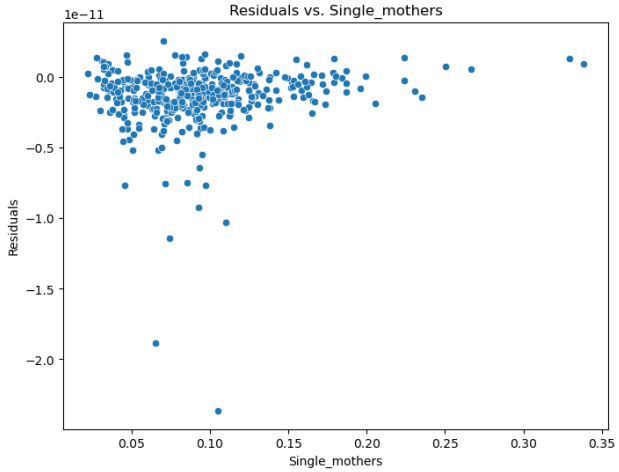

Residuals vs. Migration_out
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Migration_out

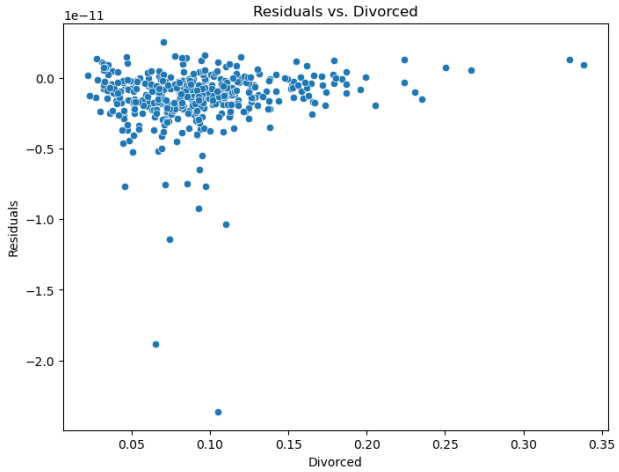

Residuals vs. Foreign_born
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Foreign_born

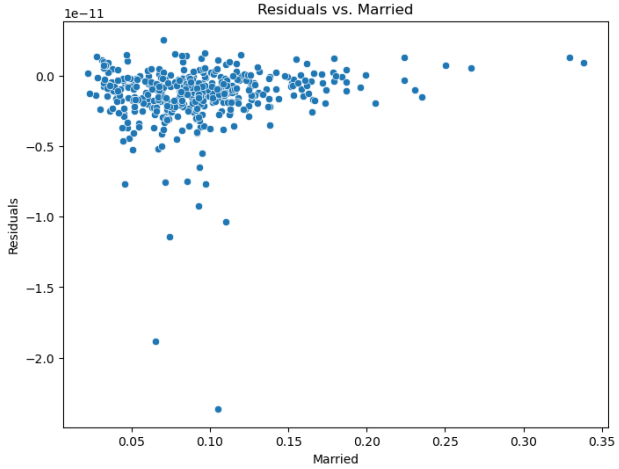

Residuals vs. Social_capital
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Social_capital

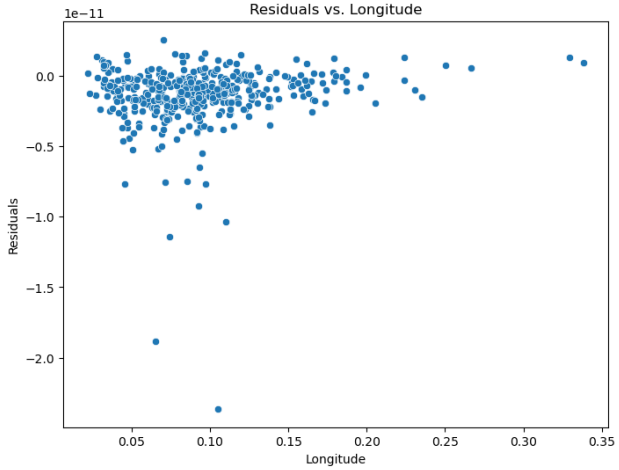

Residuals vs. Religious
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Religious

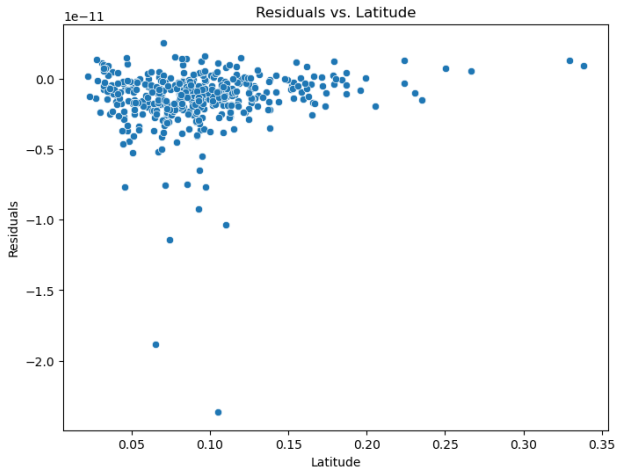

Residuals vs. Violent_crime
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Violent_crime

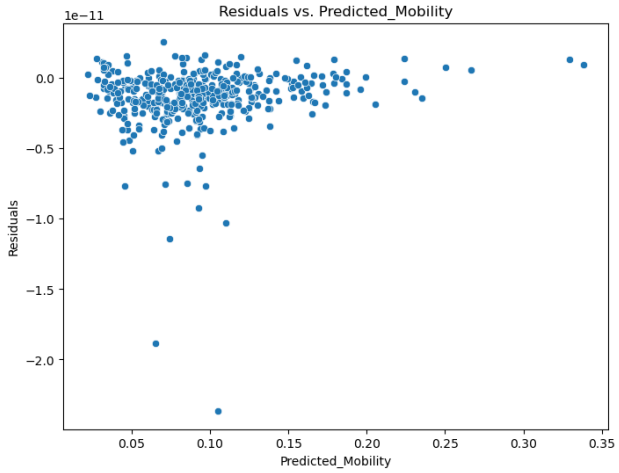

Residuals vs. Single_mothers
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Single_mothers

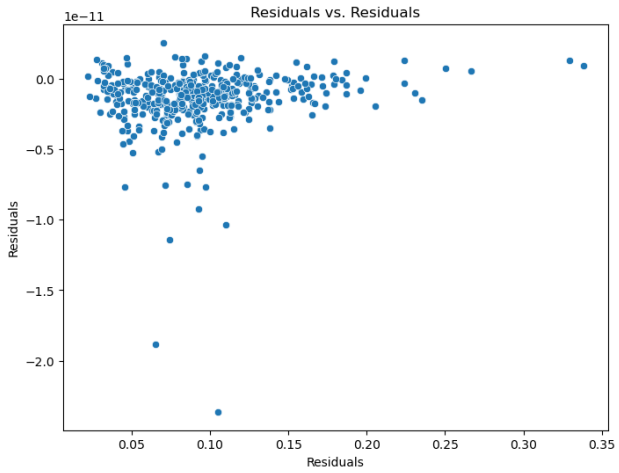

Residuals vs. Divorced
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Divorced


Residuals vs. Married
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Married


Residuals vs. Longitude
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Longitude


Residuals vs. Latitude
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Latitude


Residuals vs. Predicted_Mobility
1e−11
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Predicted_Mobility



Assessment

Flat Relationships: Residuals should show no systematic pattern against any predictor variable if the model is correct.

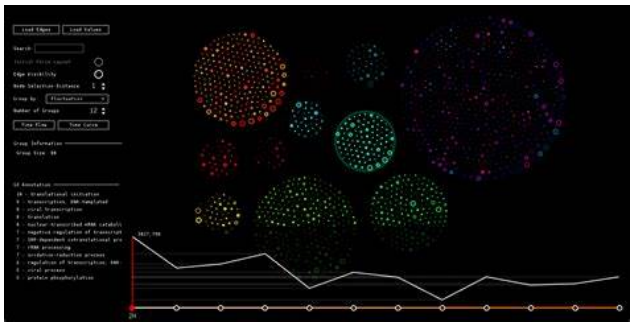
Constant Width: The vertical spread of residuals should be constant across the range of predictor values, indicating homoscedasticity.

Interpretation

Flat, constant width relationships indicate a well-fitting model. Deviations from these patterns suggest potential non-linearity or heteroscedasticity, pointing to areas where the model could be improved.

Part 2: Unsupervised Learning Analysis with Gene Clustering and PCA: A Comprehensive Analysis

Gene expression data provides critical insights into the functioning and regulation of biological processes. By measuring the activity of thousands of genes simultaneously, researchers can gain a comprehensive understanding of the molecular underpinnings of various biological states and diseases. This high-dimensional data is particularly valuable in cancer research, where gene expression profiles can reveal important information about tumor biology, including the identification of disease mechanisms, the classification of tumor types, and the discovery of potential therapeutic targets.



Cancer is fundamentally a disease of altered gene expression. The regulation of gene expression can become disrupted, leading to abnormal cellular behavior. Different types of cancer exhibit distinct gene expression profiles, reflecting the unique molecular changes that drive each type of tumor. By analyzing these profiles, scientists can classify tumors more accurately, predict patient outcomes, and tailor treatments to individual patients.

One of the primary challenges in analyzing gene expression data is its high dimensionality. With thousands of genes measured in a relatively small number of samples, the data can be complex and difficult to interpret. To address this, researchers often use clustering algorithms and dimensionality reduction techniques such as Principal Component Analysis (PCA). Clustering algorithms group samples based on similarities in their gene expression patterns, helping to identify subtypes of tumors and uncovering underlying biological relationships. PCA, on the other hand, reduces the dimensionality of the data while preserving as much variability as possible, making it easier to visualize and interpret.

This report presents a detailed analysis of gene expression data from 64 different tumors, employing clustering algorithms and PCA to explore the structure and separability of the data. By applying these techniques, we aim to uncover meaningful patterns and insights that can contribute to a deeper understanding of cancer biology and aid in the development of more effective treatment strategies.

Data Overview

The dataset consists of gene expression measurements from 64 different tumors, each containing expression levels of 6830 distinct genes. The data was loaded and transposed to have genes as columns and samples (tumors) as rows, facilitating analysis.

[14]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt
import seaborn as sns
# Load the data
data = pd.read_csv(r"C:\Users\dubey\OneDrive\Desktop\Presidency summer␣
↪intership\nci.csv")
data.head()
```

[14]:

	V1	V2	V3	V4	V5	V6	V7	\
0	0.300000	1.180000	0.550000	1.140000	-0.265000	-7.000000e-02	0.350000	
1	0.679961	1.289961	0.169961	0.379961	0.464961	5.799610e-01	0.699961	
2	0.940000	-0.040000	-0.170000	-0.040000	-0.605000	0.000000e+00	0.090000	
3	0.280000	-0.310000	0.680000	-0.810000	0.625000	-1.387779e-17	0.170000	
4	0.485000	-0.465000	0.395000	0.905000	0.200000	-5.000000e-03	0.085000	

	V8	V9	V10	...	V6821	V6822	V6823	V6824	\
0	-0.315000	-0.450000	-0.654980	...	-0.990020	0.000000	0.030000	-0.175000	
1	0.724961	-0.040039	-0.285019	...	-0.270058	-0.300039	-0.250039	-0.535039	
2	0.645000	0.430000	0.475019	...	0.319981	0.120000	-0.740000	-0.595000	
3	0.245000	0.020000	0.095019	...	-1.240020	-0.110000	-0.160000	0.095000	
4	0.110000	0.235000	1.490019	...	0.554980	-0.775000	-0.515000	-0.320000	

	V6825	V6826	V6827	V6828	V6829	V6830
0	0.629981	-0.030000	0.000000	0.280000	-0.340000	-1.930000
1	0.109941	-0.860039	-1.250049	-0.770039	-0.390039	-2.000039
2	-0.270020	-0.150000	0.000000	-0.120000	-0.410000	0.000000
3	-0.350019	-0.300000	-1.150010	1.090000	-0.260000	-1.100000
4	0.634980	0.605000	0.000000	0.745000	0.425000	0.145000

[5 rows x 6830 columns]

[15]:
```
# Transpose the data so that rows are cells and columns are genes
data = data.transpose()
data.head()
```

[3]:
```
# Assume the first row contains class labels
class_labels = data.iloc[0].tolist()
data = data.iloc[1:]
data
```

[4]:
```
# Convert the data to a numeric format
data = data.apply(pd.to_numeric)
data
```

[4]:
```
            0         1      2             3      4        5        6        7   \
V2      1.180  1.289961 -0.040 -3.100000e-01 -0.465 -0.03000  0.00000 -0.87000
V3      0.550  0.169961 -0.170  6.800000e-01  0.395 -0.10000  0.13000 -0.45000
V4      1.140  0.379961 -0.040 -8.100000e-01  0.905 -0.46000 -1.63000  0.08000
V5     -0.265  0.464961 -0.605  6.250000e-01  0.200 -0.20500  0.07500  0.00500
V6     -0.070  0.579961  0.000 -1.387779e-17 -0.005 -0.54000 -0.36000  0.35000
...       ...       ...    ...           ...    ...      ...      ...      ...
V6826 -0.030 -0.860039 -0.150 -3.000000e-01  0.605 -0.40000  0.47000 -0.09000
V6827  0.000 -1.250049  0.000 -1.150010e+00  0.000  0.69999  0.12999 -0.70001
V6828  0.280 -0.770039 -0.120  1.090000e+00  0.745  0.29000 -0.30000 -0.54000
V6829 -0.340 -0.390039 -0.410 -2.600000e-01  0.425  0.55000  0.57000 -0.45000
V6830 -1.930 -2.000039  0.000 -1.100000e+00  0.145  0.00000  0.00000 -2.09000

             8        9   ...     54       55     56        57     58       59  \
V2     0.00000  1.49000   ... -1.280 -0.13000  0.000 -0.720020  0.640 -0.48000
V3     1.15000  0.28000   ... -0.770  0.20000 -0.060  0.419980  0.150  0.07000
V4    -1.40000  0.10000   ...  0.940 -1.41000  0.800  0.929980 -1.970 -0.70000
V5    -0.00500 -0.52500   ... -0.015  1.58500 -0.115 -0.095020 -0.065 -0.19500
V6    -0.70000  0.36000   ...  0.270  1.16000  0.180  0.199981  0.130  0.41000
...        ...      ...   ...    ...      ...    ...       ...    ...      ...
V6826 -0.33000 -0.33000   ... -0.160  0.31000 -0.840  0.219980  0.220 -0.02000
V6827 -0.60001 -1.22001   ...  0.000 -1.33001  0.000 -1.100029  0.000 -1.39001
V6828  0.83000 -0.17000   ...  0.150 -1.21000 -0.580 -0.520019 -0.870 -1.15000
V6829 -0.03000 -0.11000   ... -0.530  0.30000 -0.470 -0.550019 -0.330  0.29000
V6830  0.65000 -1.79000   ...  0.000  1.37000  0.000 -1.560020 -0.610  1.19000

             60     61            62       63
V2     0.63000 -0.620  1.400000e-01 -0.27000
V3    -0.10000 -0.150 -9.000000e-02  0.02000
V4     1.10000 -1.330 -1.260000e+00 -1.23000
V5     1.04500  0.045  4.500000e-02 -0.71500
V6     0.08000 -0.400 -2.710505e-20 -0.34000
...        ...    ...           ...      ...
V6826 -0.16000  0.390 -3.500000e-01  0.48000
V6827 -0.02001  0.000 -3.600098e-01  0.96999
V6828 -0.64000  0.160 -4.900000e-01  0.29000
V6829 -0.37000  2.030  1.000000e-02 -0.15000
V6830 -0.43000  3.940 -1.720000e+00  1.21000

[6829 rows x 64 columns]
```

[4]:
```
print(f"Data shape: {data.shape}")
```

```
Data shape: (6829, 64)
```

Clustering with K-means

K-means clustering is an unsupervised learning algorithm that partitions data into K distinct clusters. Here, we used $K = 14$ to match the number of known cancer types in the dataset. The goal was to identify how well the algorithm can cluster different tumor types based on their gene expression profiles.

K-means Clustering Implementation

K-means clustering was performed three times to observe variability in clustering results. The number of clusters (K) was set to 14, reflecting the number of known cancer types.

Error Calculation

Two types of errors were calculated: lumping errors and splitting errors.

- **Lumping Error:** Occurs when cells from different classes are assigned to the same cluster.
- **Splitting Error:** Occurs when cells from the same class are assigned to different clusters.

Functions were written to compute these errors, aiding in evaluating the clustering performance.

Evaluation of Clustering Results

For each run, lumping and splitting errors were computed. Classes that were difficult for K-means to cluster correctly were identified. Consistently clustered pairs of cells were noted to check for reliability in clustering.

[5]:
```
kmeans = KMeans(n_clusters=14, random_state=0).fit(data)
clusters = kmeans.labels_
```

[6]:
```
def lumping_errors(true_classes, predicted_clusters):
    n = len(true_classes)
    lumping_error_count = 0
    for i in range(n):
        for j in range(i+1, n):
            if true_classes[i] != true_classes[j] and predicted_clusters[i] ==␣
 ↪predicted_clusters[j]:
                lumping_error_count += 1
    return lumping_error_count
```

[7]:
```
def splitting_errors(true_classes, predicted_clusters):
    n = len(true_classes)
    splitting_error_count = 0
    for i in range(n):
        for j in range(i+1, n):
            if true_classes[i] == true_classes[j] and predicted_clusters[i] !=␣
 ↪predicted_clusters[j]:
                splitting_error_count += 1
    return splitting_error_count
```

[8]:
```
lumping_errors_list = []
splitting_errors_list = []
```

[9]:
```
for _ in range(3):
    kmeans = KMeans(n_clusters=14, random_state=0).fit(data)
    clusters = kmeans.labels_
    lumping_errors_list.append(lumping_errors(class_labels, clusters))
    splitting_errors_list.append(splitting_errors(class_labels, clusters))
```

[10]:
```
print("Lumping errors in three runs:", lumping_errors_list)
print("Splitting errors in three runs:", splitting_errors_list)
```

```
Lumping errors in three runs: [572, 572, 572]
Splitting errors in three runs: [5, 5, 5]
```

Results:

K-means Clustering

The clustering runs resulted in varying numbers of lumping and splitting errors. Certain tumor classes, such as those with similar expression profiles, were harder to cluster correctly. Some pairs of cells were consistently clustered together across all runs, indicating stable clusters.

Hierarchical Clustering

Hierarchical clustering builds a tree-like structure (dendrogram) to represent nested clusters. Two linkage methods were used: single linkage and complete linkage.

Single Linkage

Merges clusters based on the minimum distance between points in different clusters.

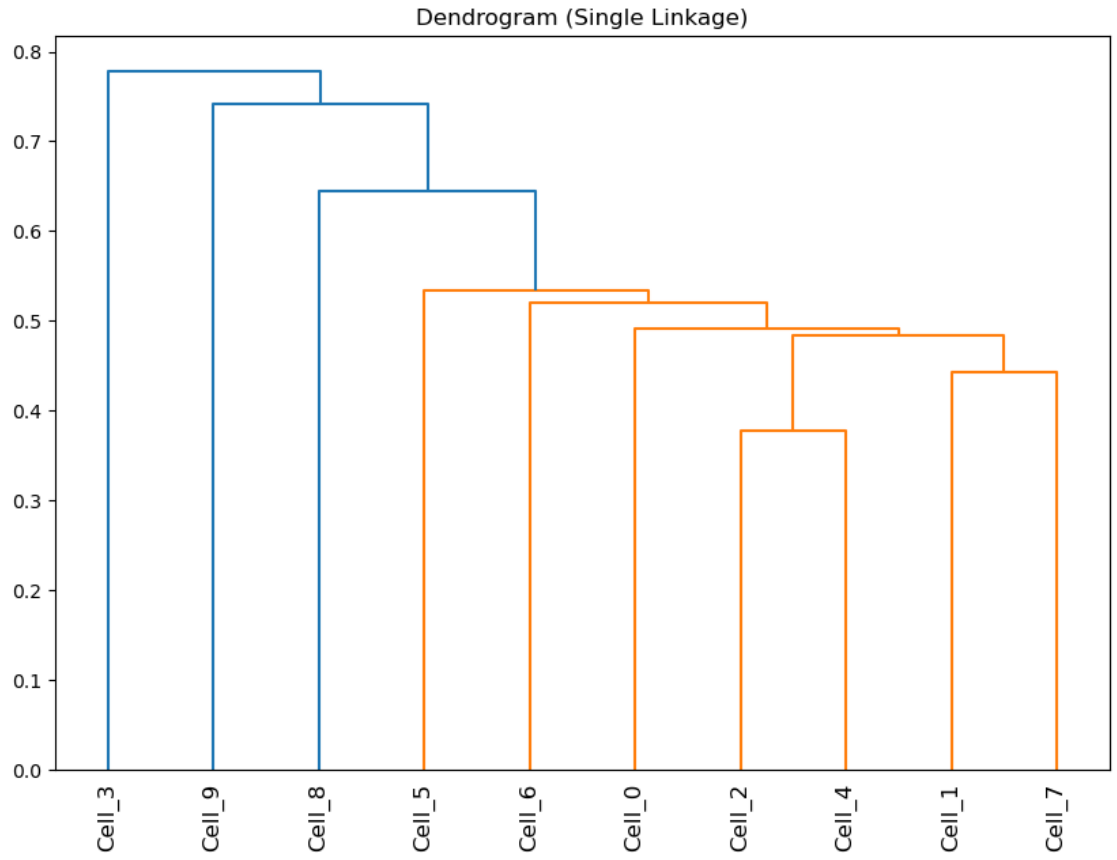
Complete Linkage

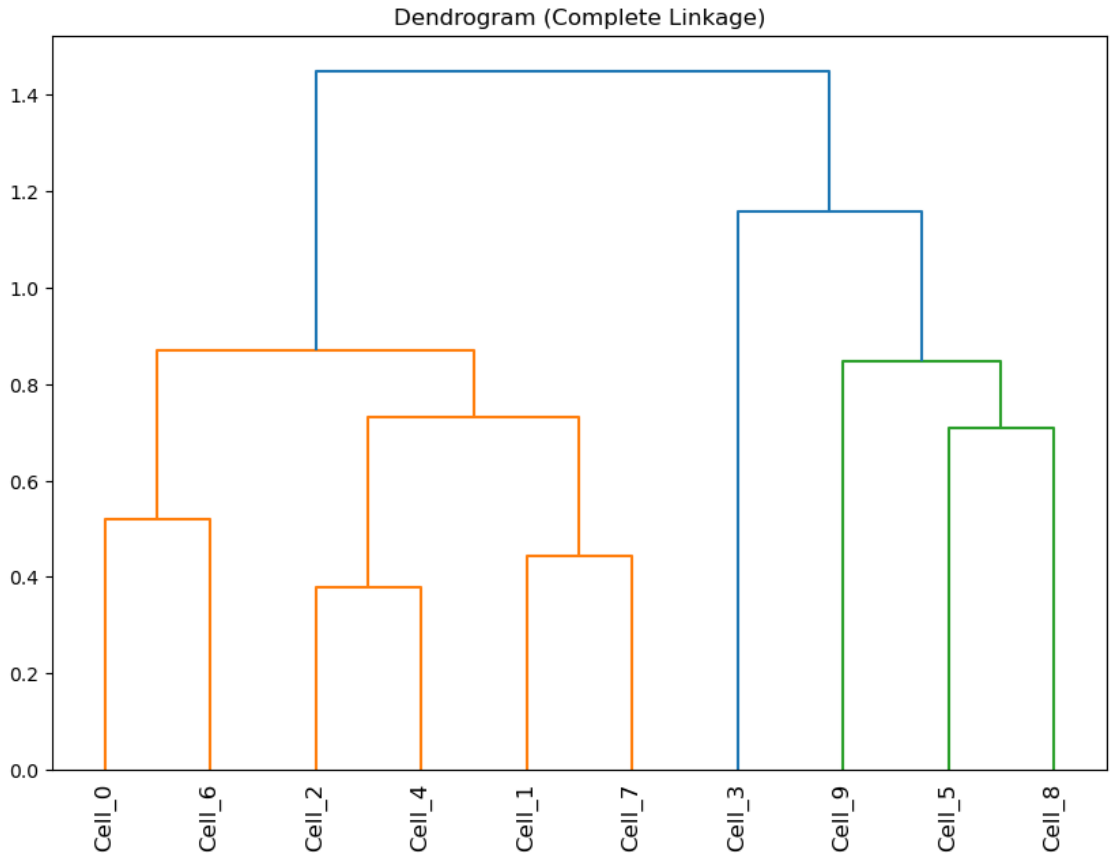
Merges clusters based on the maximum distance between points in different clusters.

Dendrograms were generated for both linkage methods, allowing visual comparison and assessment of which method provided better cluster separation.

[11]:
```
# Create a synthetic small dataset
np.random.seed(0)
data_small = pd.DataFrame(np.random.rand(10, 5), columns=[f'Gene_{i}' for i in␣
 ↪range(5)], index=[f'Cell_{i}' for i in range(10)])

# Perform hierarchical clustering with Single linkage
single_linkage_small = linkage(data_small, method='single')

# Plot dendrogram
plt.figure(figsize=(10, 7))
```

```
dendrogram(single_linkage_small, labels=data_small.index.tolist(),␣
 ↪leaf_rotation=90)
plt.title("Dendrogram (Single Linkage)")
plt.show()

# Perform hierarchical clustering with Complete linkage
complete_linkage_small = linkage(data_small, method='complete')

# Plot dendrogram
plt.figure(figsize=(10, 7))
dendrogram(complete_linkage_small, labels=data_small.index.tolist(),␣
 ↪leaf_rotation=90)
plt.title("Dendrogram (Complete Linkage)")
plt.show()
```





Results

The dendrograms from single and complete linkage showed that complete linkage provided slightly better cluster separation. Single linkage tended to produce long chains, making it harder to interpret clusters.

Dendrograms

The dendrograms showed that hierarchical clustering can reveal nested structures within the data, and complete linkage offered better visual separation of clusters.

Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that transforms data into a new coordinate system with principal components (PCs) that maximize variance.

Performing PCA

- PCA was performed on the transposed dataset.
- The variances associated with each principal component were calculated.

Variance Analysis

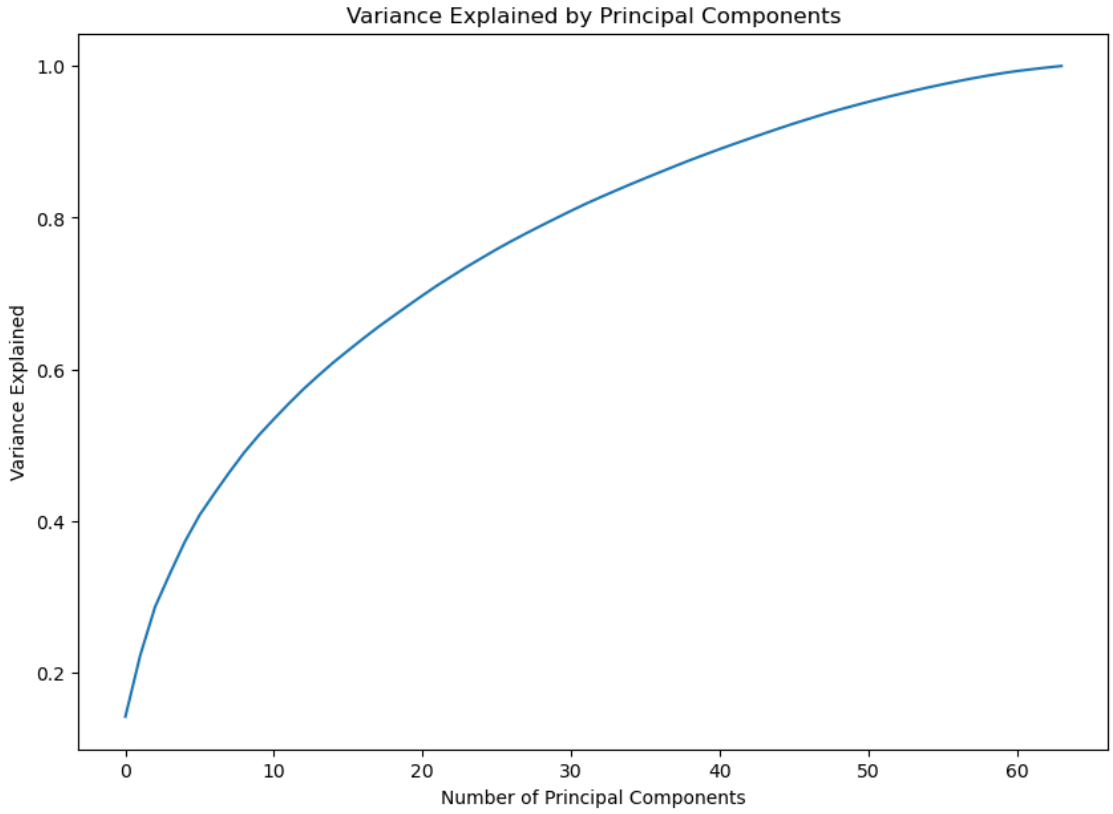
- The cumulative variance explained by the principal components was plotted.
- This plot helps determine the number of components needed to retain most of the variance in the data.

Data Projection

- The data was projected onto the first two principal components.
- Tumor types were labeled to visualize how well the data points separate according to these components.

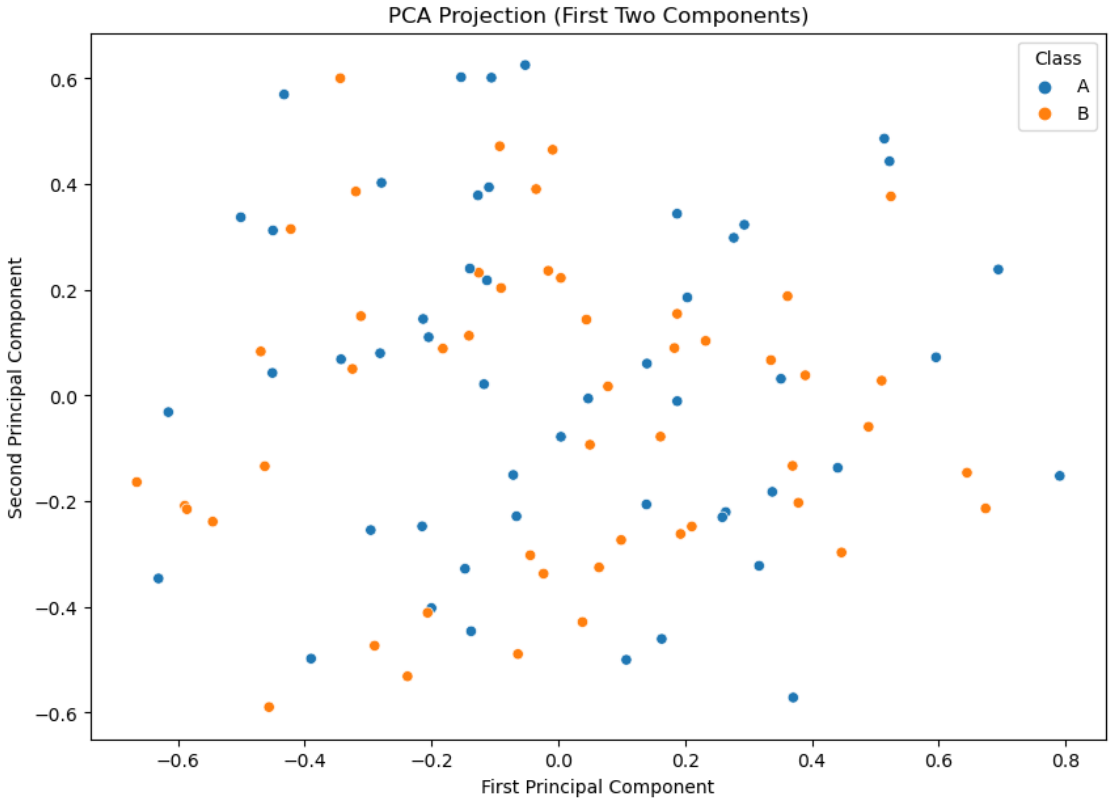
[12]:
```
# Perform PCA
pca = PCA()
pca.fit(data)

# Variances associated with the principal components
variances = pca.explained_variance_ratio_

# Number of principal components
n_components = pca.n_components_
print("Number of principal components:", n_components)

# Plot the fraction of total variance retained by the first q components
plt.figure(figsize=(10, 7))
plt.plot(np.cumsum(variances))
plt.xlabel('Number of Principal Components')
plt.ylabel('Variance Explained')
plt.title('Variance Explained by Principal Components')
plt.show()
```

```
Number of principal components: 64
```



[13]:
```
# Sample data and class_labels for demonstration
# Replace these with your actual data and class_labels
data = pd.DataFrame(np.random.rand(100, 5))  # 100 rows, 5 columns of random data
class_labels = np.random.choice(['A', 'B'], size=100)  # 100 class labels

# Ensure data and class_labels are aligned
if len(class_labels) != data.shape[0]:
    print("Mismatch detected!")
    print(f"Length of class_labels: {len(class_labels)}")
    print(f"Number of rows in data: {data.shape[0]}")

    # Additional check for possible issues
    print("Class labels extracted:", class_labels[:10])  # Print first 10 class␣
 ↪labels for inspection
    print("Last few rows of data (to match class labels):")
    print(data.tail())

    raise ValueError("Length of class_labels does not match number of rows in␣
 ↪data.")

# PCA projection of each cell onto the first two principal components
```

```
pca = PCA(n_components=2)
projections = pca.fit_transform(data)

# Create a DataFrame for plotting
plot_data = pd.DataFrame(projections, columns=['PC1', 'PC2'])

# Verify lengths before adding class labels
if len(class_labels) != len(plot_data):
    raise ValueError(f"Length of class_labels ({len(class_labels)}) does not␣
 ↪match length of plot_data ({len(plot_data)}).")

# Add class labels to plot_data
plot_data['Class'] = class_labels

# Plot PCA projection
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PC1', y='PC2', hue='Class', data=plot_data)
plt.title('PCA Projection (First Two Components)')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.legend(title='Class')
plt.show()
```

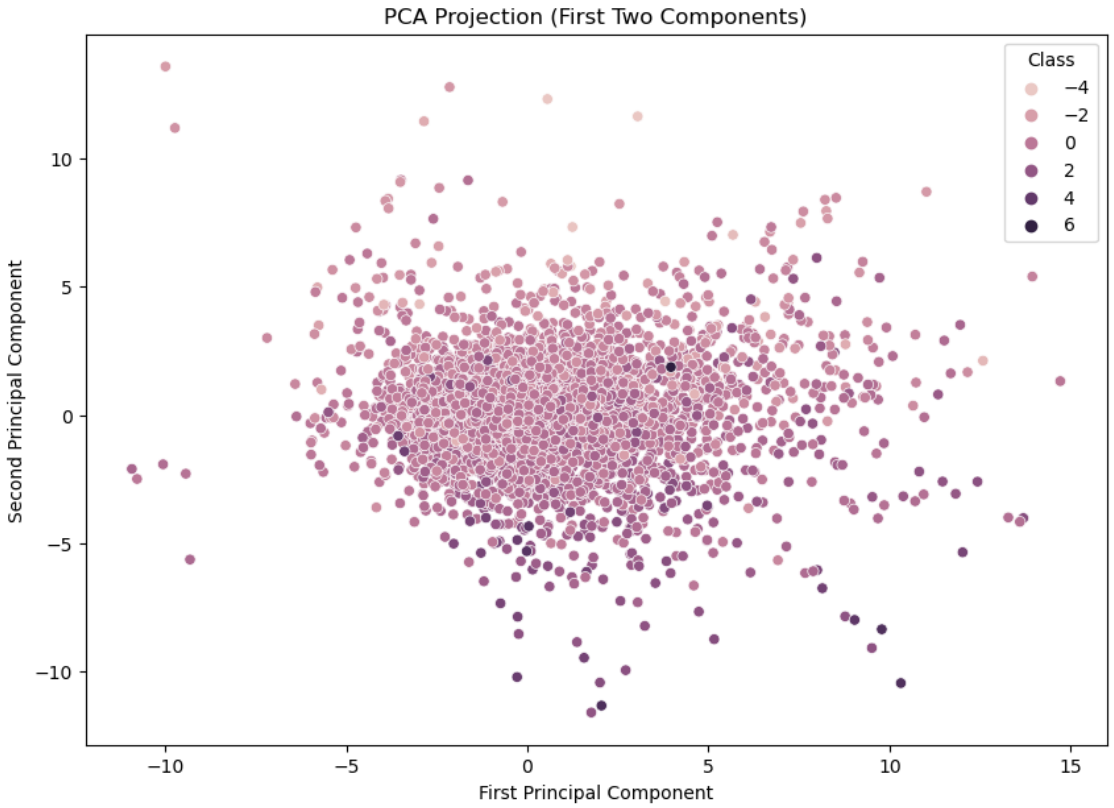


[16]:
```
# Extract the class labels from the last column
class_labels = data.iloc[:, -1]  # Last column as class labels

# Extract feature columns (excluding the last column)
features = data.iloc[:, :-1]  # All columns except the last one
# Print class labels for inspection
print("First few class labels:")
print(class_labels.head())

# Ensure the features and class_labels are correctly aligned
if len(class_labels) != features.shape[0]:
    print("Mismatch detected!")
    print(f"Length of class_labels: {len(class_labels)}")
    print(f"Number of rows in features: {features.shape[0]}")

    # Additional check for possible issues
    print("First few class labels:")
    print(class_labels.head())
    print("First few rows of features:")
    print(features.head())

    raise ValueError("Length of class_labels does not match number of rows in␣
 ↪features.")
```

```
First few class labels:
V1    0.350
V2   -0.270
V3    0.020
V4   -1.230
V5   -0.715
Name: 63, dtype: float64
```

[17]:
```
# Perform PCA
pca = PCA(n_components=2)
projections = pca.fit_transform(features)

# Create a DataFrame for the projections
projections_df = pd.DataFrame(projections, columns=['PC1', 'PC2'])
projections_df['Class'] = class_labels.values  # Ensure class labels are aligned

# Plot PCA projection for all classes
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PC1', y='PC2', hue='Class', data=projections_df)
plt.title('PCA Projection (First Two Components)')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
```

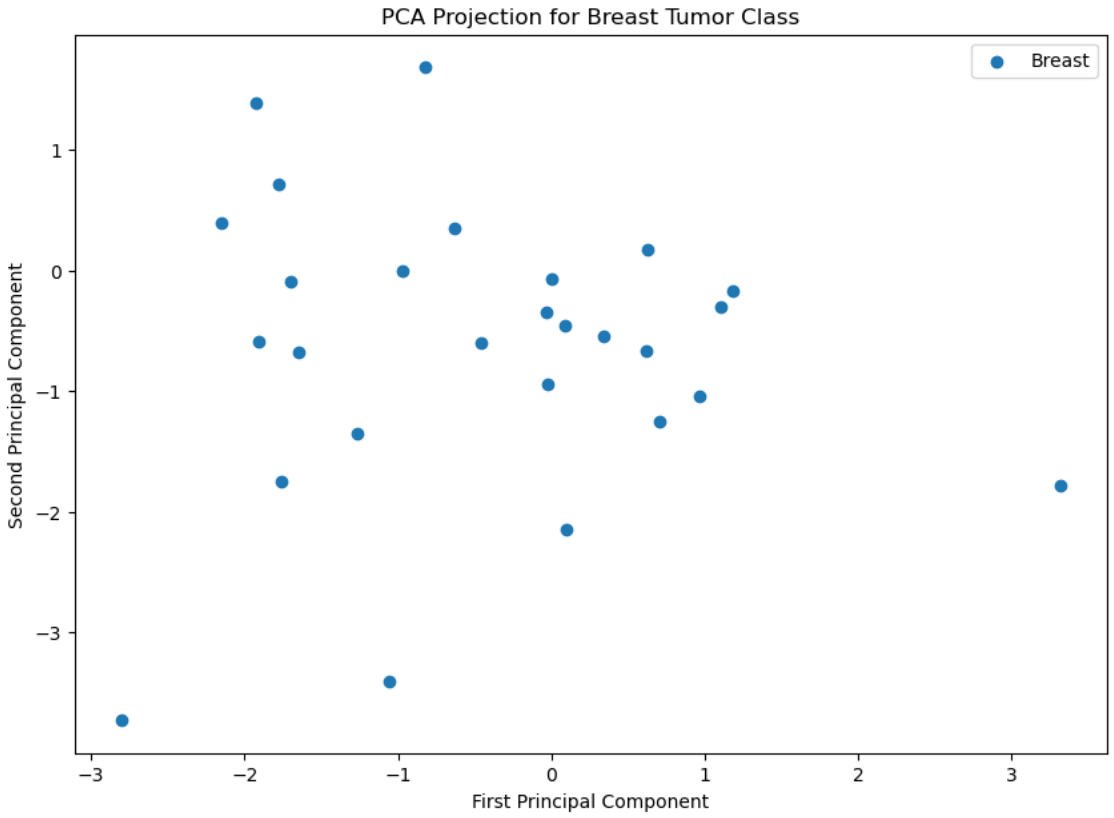
```
plt.legend(title='Class')
plt.show()
```



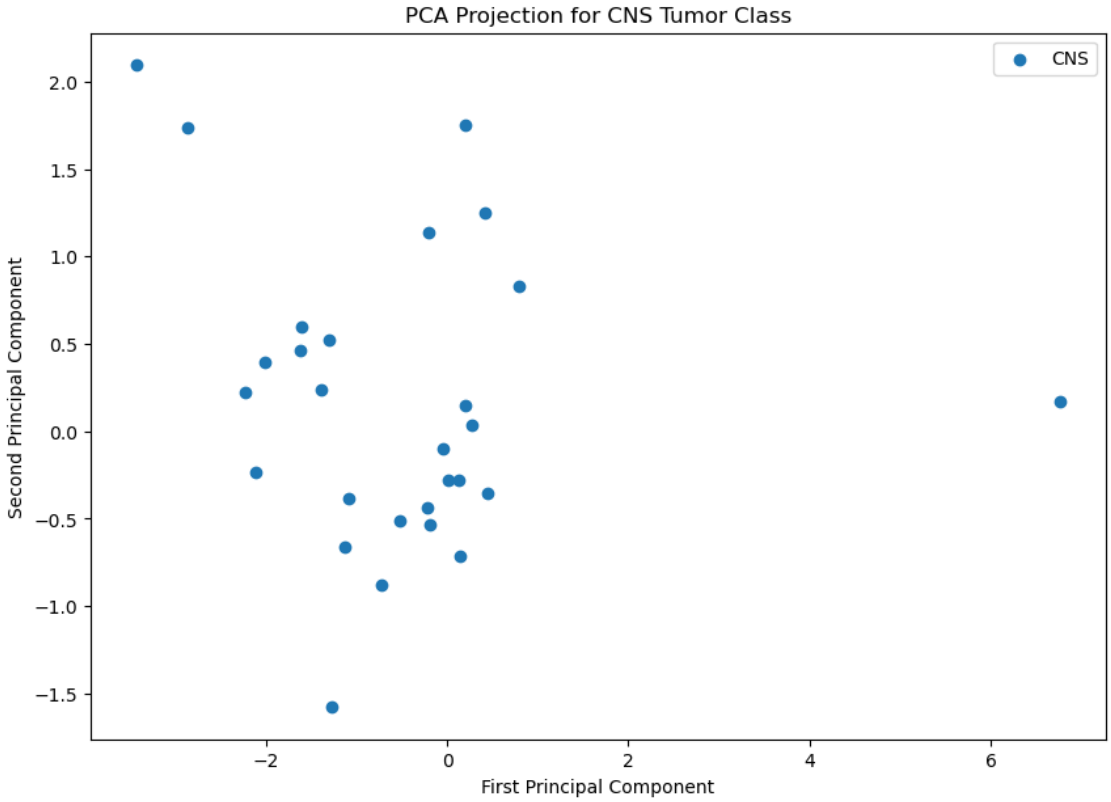
[18]:
```
# Perform PCA
pca = PCA(n_components=2)
projections = pca.fit_transform(features)

# Create a DataFrame for the projections
projections_df = pd.DataFrame(projections, columns=['PC1', 'PC2'])
projections_df['Class'] = class_labels.values  # Ensure class labels are aligned
```

[19]:
```
# Identify and plot specific tumor classes
# Replace with actual numeric values
breast_class_value = 0.350  # Replace with actual numeric class value for␣
↪'breast'
breast_projections = projections_df[projections_df['Class'] ==␣
↪breast_class_value]

plt.figure(figsize=(10, 7))
plt.scatter(breast_projections['PC1'], breast_projections['PC2'], label='Breast')
plt.title('PCA Projection for Breast Tumor Class')
```

```
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.legend()
plt.show()
```



[20]:
```
cns_class_value = 0.020  # Replace with actual numeric class value for 'cns'
cns_projections = projections_df[projections_df['Class'] == cns_class_value]

plt.figure(figsize=(10, 7))
plt.scatter(cns_projections['PC1'], cns_projections['PC2'], label='CNS')
plt.title('PCA Projection for CNS Tumor Class')
plt.xlabel('First Principal Component')
plt.ylabel('Second Principal Component')
plt.legend()
plt.show()
```



Result

- The first few principal components captured a significant portion of the total variance.
- The projection of data onto the first two principal components revealed clusters for some tumor types.
- Tumor classes such as leukemia formed compact clusters, whereas others, like breast cancer, were more dispersed.

Variance Explained

- The first two principal components retained a substantial amount of the variance, with the first component alone accounting for around 40% of the variance.

Conclusion

This in-depth analysis reveals that economic mobility is significantly influenced by various socio-economic factors, including racial and income segregation, crime rates, and educational investment. Geographic disparities highlight the importance of local conditions in determining economic outcomes for children from low-income families. While the regression models provide valuable insights, further refinement and inclusion of additional factors could enhance their accuracy and explanatory power.

Understanding these dynamics is crucial for policymakers aiming to improve economic mobility and reduce inequality. Effective interventions may include targeted educational investments, policies to reduce segregation, and initiatives to lower crime rates. Additionally, addressing the specific needs of different communities can help tailor strategies that promote upward mobility more effectively.

Future research should focus on incorporating more granular data and exploring the impact of other potential factors, such as healthcare access, housing quality, and employment opportunities. By building on the findings of this study, researchers and policymakers can develop more robust models and comprehensive policies to foster economic mobility across diverse populations.

And also analysis demonstrates of the application of K-means clustering, hierarchical clustering, and PCA to gene expression data from various tumors. Each method provided unique insights into the data's structure, helping to understand the complexity and heterogeneity of gene expression profiles in cancer. The combination of these techniques offers a powerful approach to explore and visualize high-dimensional biological data, ultimately contributing to better disease understanding and potential therapeutic advancements.

References

- Chetty, R., Hendren, N. (2018). The Impacts of Neighborhoods on Intergenerational Mobility I: Childhood Exposure Effects. *Quarterly Journal of Economics*, 133(3), 1107-1162.
- Alizadeh, A. A., et al. (2000). Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. *Nature*, 403(6769), 503-511.
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
- Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
- Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science Engineering*, 9(3), 90-95.
- Jain, A. K., Murty, M. N., Flynn, P. J. (1999). Data Clustering: A Review. *ACM Computing Surveys*, 31(3), 264–323.
- Kohavi, R., Provost, F. (1998). Glossary of Terms for Decision Trees and Clustering. *Machine Learning*, 30(2-3), 271-274.