Empirical Project 4 (Using Google DataCommons to Predict Social Mobility)

Helen (Yingying) Huang

Part 1: Data set up

In [59]:
```
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import zscore, norm
import statsmodels.api as sm
from statsmodels.formula.api import ols
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, make_scorer
from sklearn.model_selection import cross_val_score, KFold
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

%matplotlib inline
warnings.filterwarnings('ignore')
```

1. Go to Google DataCommons and select at least 10 county-level variables that you think might be useful in predicting the statistic that we are using to describe intergenerational mobility which is the variable kfr_pooled_p25.

Value:Count_Person_BachelorOfArtsHumanitiesAndOtherMajor

Value:Count_Person_BachelorOfBusinessMajor

Value:Count_Person_BachelorOfEducationMajor

Value:Count_Person_BachelorOfScienceAndEngineeringMajor

Value:Count_Person_BachelorOfScienceAndEngineeringRelatedMajor

Value:Count_HousingUnit_WithCashRent

Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome

Value:Median_Age_Person

Value:Median_Income_Person

Value:StandardizedPrecipitationIndex_Atmosphere

2. Select and download at least 10 predictors in DataCommons for all counties in the United States. First, select a geography and choose predictors. Next, click “Get Code/Data”. Then, click “Bulk Download data.” Picking a particular year will generate a .csv file that contains the data for all counties. (Note that some data are only available in certain years, so you should pick a year where the variables you want to use are available).

I selected 10 predictors listed in the first question, using data from the year 2015, at the county level in the US.

3. Merge these data with the atlas_training.dta data file.

Datasets (df1 and df2), which contain the 10 predictors downloaded from DataCommons, were merged with the atlas_training.dta (df3) data file as follows:

In [60]:
```
### Prepare the data for the set of 10 predictors
## Load the datasets

df1 = pd.read_csv('1-5predictors.csv')
df2 = pd.read_csv('6-10predictors.csv')
df3 = pd.read_stata('atlas_training.dta')
```

In [61]:
```
df1.head()
```

Out[61]:

	placeDcid	placeName	Date:Count_Person_BachelorOfArtsHumanitiesAndOtherMajor	Value:Count_Person_BachelorOfArtsHumanitiesAndOthe
0	geoId/01001	Autauga County	2015	
1	geoId/01003	Baldwin County	2015	
2	geoId/01005	Barbour County	2015	
3	geoId/01007	Bibb County	2015	
4	geoId/01009	Blount County	2015	

In [62]:
```
df2.head()
```

Out[62]:

	placeDcid	placeName	Date:Count_HousingUnit_WithCashRent	Value:Count_HousingUnit_WithCashRent	Source:Count_HousingUnit_WithCa
0	geoId/01001	Autauga County	2015	4796	https://www.census.gov/prc surveys/a
1	geoId/01003	Baldwin County	2015	18880	https://www.census.gov/prc surveys/a
2	geoId/01005	Barbour County	2015	2855	https://www.census.gov/prc surveys/a
3	geoId/01007	Bibb County	2015	1414	https://www.census.gov/prc surveys/a
4	geoId/01009	Blount County	2015	3599	https://www.census.gov/prc surveys/a

In [63]:
```
df3.head()
```

Out[63]:

	geoid	place	pop	housing	kfr_pooled_p25	test	training	P_1	P_2	P_3	...	P_112	P_113	P_114	P_115
0	1003.0	Baldwin County	187114	104061	0.388847	0.0	1.0	82.847946	98.593452	101.776711	...	0.0	19.100000	0.00	2.01
1	1005.0	Barbour County	27321	11829	0.349386	0.0	1.0	76.313896	93.878723	90.702942	...	0.0	45.160000	0.00	4.84
2	1007.0	Bibb County	22754	8981	0.363391	0.0	1.0	73.765617	104.868469	82.129547	...	0.0	30.910000	0.00	7.27
3	1013.0	Butler County	20624	9964	0.357249	0.0	1.0	92.096672	121.073296	117.823196	...	0.0	41.070000	0.00	5.36
4	1015.0	Calhoun County	117714	53289	0.361847	0.0	1.0	76.938210	95.478249	98.326622	...	0.0	18.790001	0.61	3.03

5 rows × 128 columns

In [64]:
```
### Prepare the data for the set of 10 predictors
## Merge data

def rename_columns(df, rename_dict):
    return df.rename(columns=rename_dict, inplace=True)

def remove_prefix_suffix(df, col_name, prefix=None, suffix=None):
    if prefix:
        df[col_name] = df[col_name].str.replace(prefix, "")
    if suffix:
        df[col_name] = df[col_name].str.replace(suffix, "")

rename_dict = {"placeDcid": "identifier", "placeName": "county_name"}
rename_columns(df1, rename_dict)
rename_columns(df2, rename_dict)

rename_dict = {"geoid": "identifier", "place": "county_name"}
rename_columns(df3, rename_dict)

df3['identifier'] = df3['identifier'].astype(str)

remove_prefix_suffix(df1, "identifier", prefix="geoId/")
remove_prefix_suffix(df2, "identifier", prefix="geoId/")
remove_prefix_suffix(df3, "identifier", suffix=".0")

cb1 = df3.merge(df1, on=["identifier", "county_name"], how="inner")
cb = cb1.merge(df2, on=["identifier", "county_name"], how="inner")

cb
```

Out[64]:

	identifier	county_name	pop	housing	kfr_pooled_p25	test	training	P_1	P_2	P_3	...	Source:GenderIncomeInec
0	12007	Bradford County	27981	11011	0.354766	0.0	1.0	28.083626	35.928146	27.845144	...	
1	12015	Charlotte County	161276	100632	0.413865	0.0	1.0	44.377594	61.532696	65.234047	...	
2	12017	Citrus County	140214	78026	0.394591	0.0	1.0	38.371532	47.172661	55.010231	...	
3	12027	DeSoto County	34651	14590	0.356809	0.0	1.0	21.147446	31.037828	35.652672	...	
4	12031	Duval County	872598	388486	0.349491	0.0	1.0	58.278805	75.047768	77.089523	...	
...	...	...	...	...	...	...	...	...	...	...	...	
2238	72115	Quebradillas Municipio	25738	10754	NaN	1.0	0.0	42.974659	53.096306	52.796627	...	
2239	72121	Sabana Grande Municipio	24974	10958	NaN	1.0	0.0	42.974659	53.096306	52.796627	...	
2240	72123	Salinas Municipio	30807	14380	NaN	1.0	0.0	42.974659	53.096306	52.796627	...	
2241	72137	Toa Baja Municipio	88195	36546	NaN	1.0	0.0	42.974659	53.096306	52.796627	...	
2242	72141	Utuado Municipio	32593	14192	NaN	1.0	0.0	42.974659	53.096306	52.796627	...	

2243 rows × 158 columns

4. Many of the Google DataCommons variables are counts (e.g., total number of female residents of a county or owner-occupied housing units). Replace these counts with rates (e.g., percent female or fraction of owner-occupied housing units) by dividing by the population and housing variables given to you in atlas_training.dta. (Note that Google DataCommons is still under development; although you can draw graphs with per capita figures, only the counts can be downloaded via the Bulk Downloads).

Counts were converted to rates as follows:

In [65]:

```
### Prepare the data for the set of 10 predictors
## Calculate rates for the count variables

count_columns = [col for col in cb.columns if "Value:Count_" in col]
for count_col in count_columns:
    denominator = 'housing' if "Housing" in count_col else 'pop'
    rate_col = count_col.replace("Count", "Rate")
    cb[rate_col] = cb[count_col] / cb[denominator]
cb.drop(columns=count_columns, inplace=True)

cb.head()
```

Out[65]:

	identifier	county_name	pop	housing	kfr_pooled_p25	test	training	P_1	P_2	P_3	...	Source:Median_Income_Pers
0	12007	Bradford County	27981	11011	0.354766	0.0	1.0	28.083626	35.928146	27.845144	...	https://www.census.gov/progran surveys/acs/da
1	12015	Charlotte County	161276	100632	0.413865	0.0	1.0	44.377594	61.532696	65.234047	...	https://www.census.gov/progran surveys/acs/da
2	12017	Citrus County	140214	78026	0.394591	0.0	1.0	38.371532	47.172661	55.010231	...	https://www.census.gov/progran surveys/acs/da
3	12027	DeSoto County	34651	14590	0.356809	0.0	1.0	21.147446	31.037828	35.652672	...	https://www.census.gov/progran surveys/acs/da
4	12031	Duval County	872598	388486	0.349491	0.0	1.0	58.278805	75.047768	77.089523	...	https://www.census.gov/progran surveys/acs/da

5 rows × 158 columns

5. Produce simple summary statistics for the 10 predictors you selected from DataCommons and krf_pooled_p25 in the combined data set for observations that exist in both data sets.

Simple summary statistics:

In [66]:

```
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Produce simple summary statistics for the 10 predictors an krf_pooled_p25

selected_columns_10 = ['kfr_pooled_p25'] + [col for col in cb.columns if "Value:" in col]
cb_selected_10 = cb[selected_columns_10]
cb_selected_10.describe()
```

Out[66]:

	kfr_pooled_p25	Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	Value:Median_Age_Person	Value:Median_Income_Per
count	1126.000000	2241.000000	2243.000000	2242.000
mean	0.414503	0.232215	40.091306	24117.000
std	0.052216	0.061199	4.790646	5492.881
min	0.212865	-0.160444	22.300000	9399.000
25%	0.379837	0.196398	37.400000	20782.250
50%	0.411379	0.232735	40.300000	23626.500
75%	0.443173	0.268950	42.900000	26855.500
max	0.614030	0.501838	65.300000	61012.000

6. Run a linear regression of kfr_pooled_p25 on the 10 predictors (converted to rates when appropriate) from Google DataCommons, inspect the results, and comment on what you find.

After experimenting with several linear regression models — with and without outliers, and with and without PCA — as shown in the code blocks and outputs below, I chose the non-PCA outlier-included model to inspect the results.

The OLS regression results indicate:

- R-squared (0.502)
- Adj. R-squared (0.498)
- F-statistic (112.6)
- Coefficients (Gender Income Inequality: 0.1626, Median Age: -0.0011, Median Income: 5.693e-06, Standardized Precipitation Index: -0.0029, Rate of BA in Humanities & Other: -0.3953, Rate of BA in Business: -1.6816, Rate of BA in Education: 2.3793, Rate of BS/BE in Science & Engineering: 0.2039, Rate of BS/BE in Science & Engineering Related: 0.6639, Rate of Housing Unit With Cash Rent: -0.0596)
- P-values (all the coefficients are significant ($p<0.05$))
- Durbin-Watson statistic (1.308)
- Skew (0.357)
- Kurtosis (3.658)

Comments:

Statistical Significance

- All predictors have p-values less than 0.05, making them statistically significant at the 5% significance level. This means that there is strong evidence against the null hypothesis for each predictor, and they are considered to have a statistically significant relationship with the dependent variable kfr_pooled_p25.

Goodness-of-Fit

- The R-squared value of 0.502 suggests that approximately 50.2% of the variability in kfr_pooled_p25 can be explained by the model's predictors. This leaves almost half of the variation unexplained, which may suggest the need for a more complex model or that there is a lot of inherent variability in the outcome that cannot be captured by any model.

- The Adjusted R-squared (which accounts for the number of predictors) is 0.498, very close to the R-squared, suggesting that the inclusion of the number of predictors is appropriate and not leading to significant overfitting.

Predictive Power

- Given that the R-squared is over 0.5, the model has relatively low predictive power as a significant proportion of the variance in the dependent variable is still unexplained by the model.

Limitations

- The large condition number suggests multicollinearity, which can undermine the reliability and clarity of the coefficient estimates. Despite the model's predictive ability, concerns arise from multicollinearity, non-normal residuals, and autocorrelation, which may affect its validity.

Practical Implications

- Practically, these results suggest that factors like gender income inequality and the composition of educational qualifications in the population have more significant associations with the variable kfr_pooled_p25.

7. How well does your linear regression predict krf_pooled_p25 in-sample?

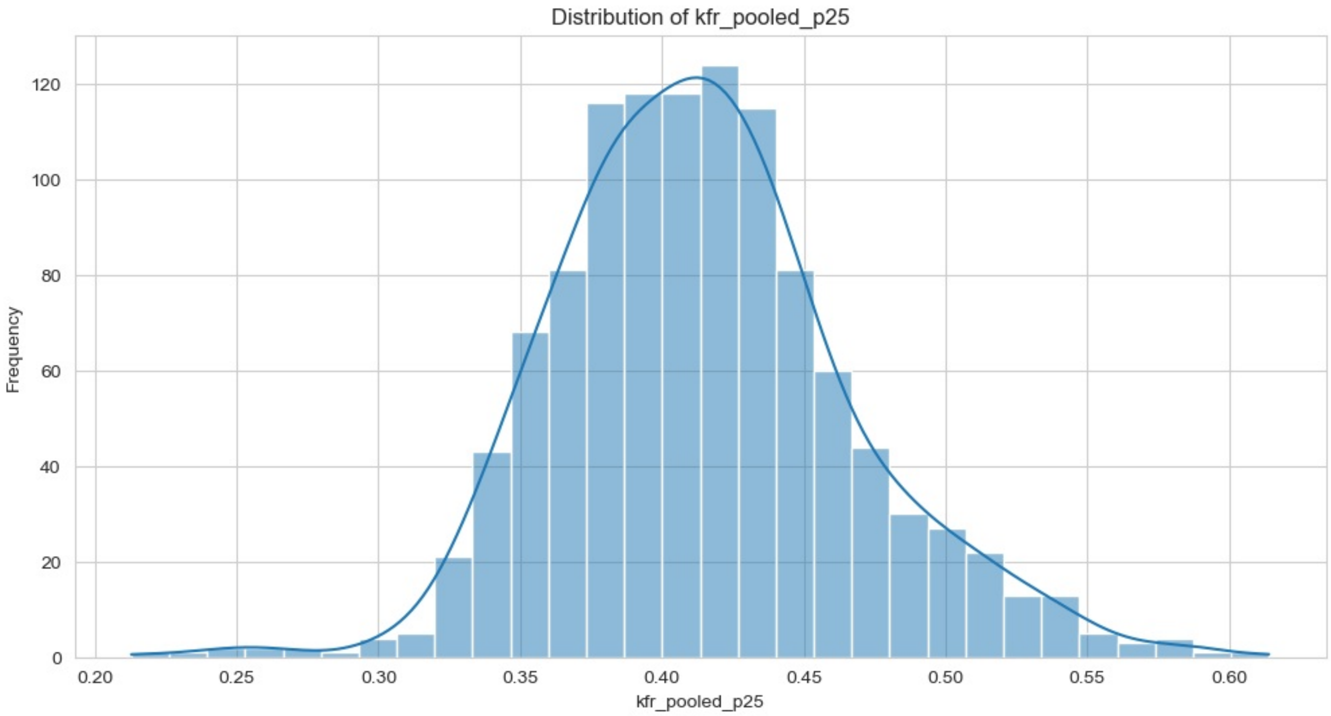
The predictive performance of the linear regression model (non-PCA outlier-included) on kfr_pooled_p25 is as follows:

- MSE: 0.0014

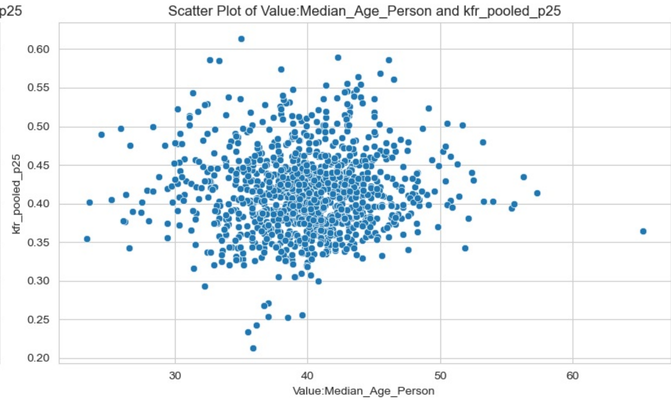
- MAE: 0.0285

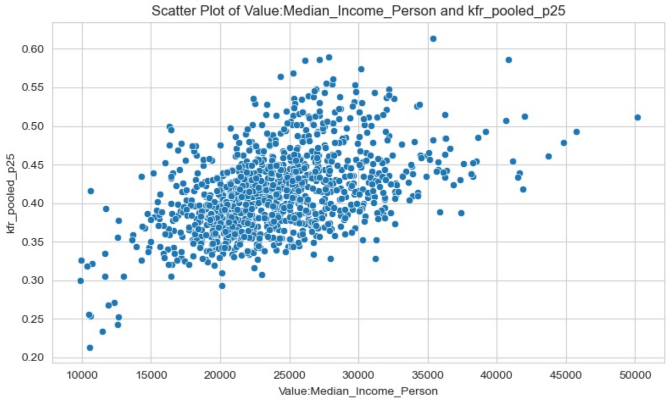
Based on these metrics, the errors are relatively small, suggesting that the model offers reasonably accurate in-sample predictions.

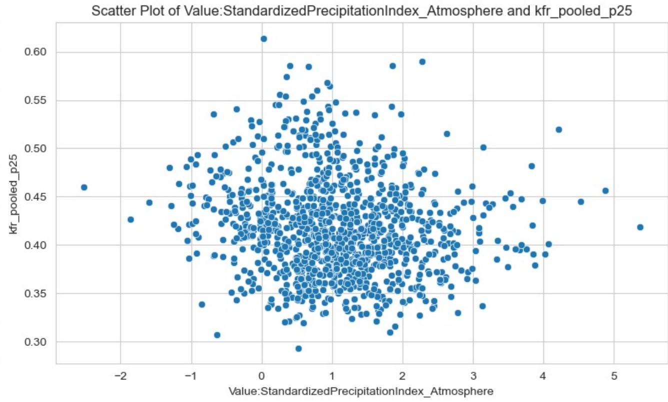
In [67]:
```python
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Inspect the distribution of the target variable
# The distribution of "kfr_pooled_p25" appears to be mostly bell-shaped with a slight right skew
# This is a good sign for linear regression, as a normal-like distribution of the dependent variable often lead

plt.figure(figsize=(12, 6))

sns.histplot(cb['kfr_pooled_p25'], bins=30, kde=True)
plt.title('Distribution of kfr_pooled_p25')
plt.xlabel('kfr_pooled_p25')
plt.ylabel('Frequency')

plt.show()
```

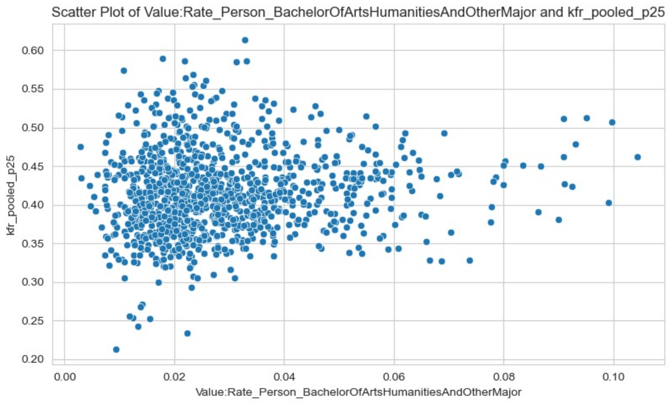


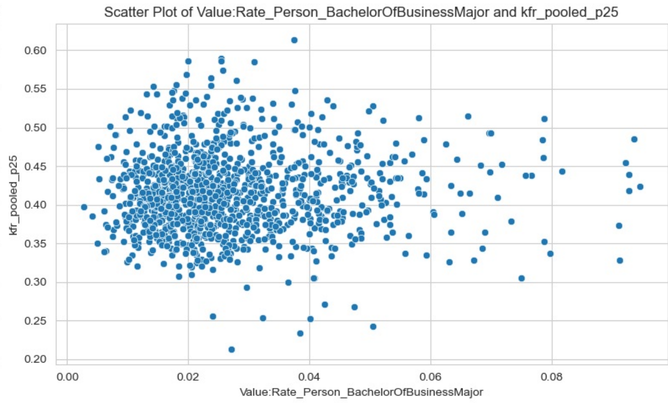
In [68]:
```python
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Inspect the distribution of the predictor variables
# None of these plots show a strong, clear linear trend, which suggests that simple linear regression may not b
# While a few plots exhibit some outliers, they are not too pronounced

fig, axes = plt.subplots(5, 2, figsize=(16, 24))
axes = axes.flatten()

var_columns_10 = cb[selected_columns_10].drop(columns='kfr_pooled_p25').columns

for i, ax in enumerate(axes):
    var = var_columns_10[i]
    sns.scatterplot(x=var, y='kfr_pooled_p25', data=cb, ax=ax)
    ax.set_title(f'Scatter Plot of {var} and kfr_pooled_p25')
    ax.set_xlabel(var)
    ax.set_ylabel('kfr_pooled_p25')

plt.tight_layout()
plt.show()
```

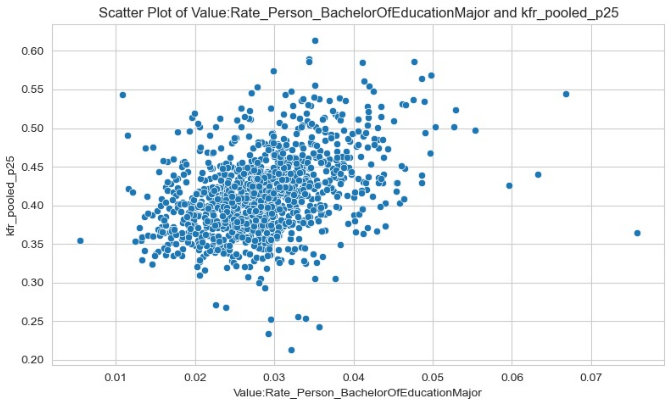


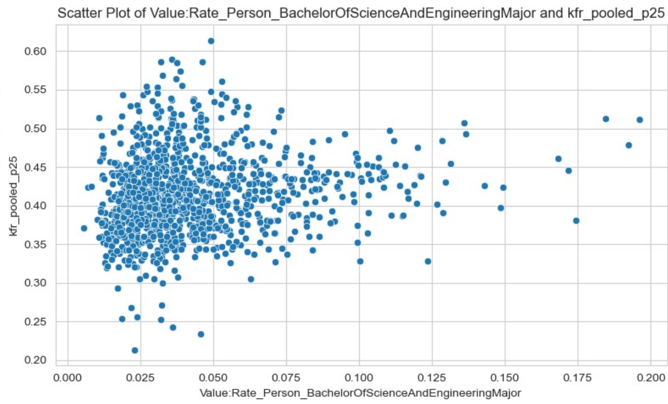


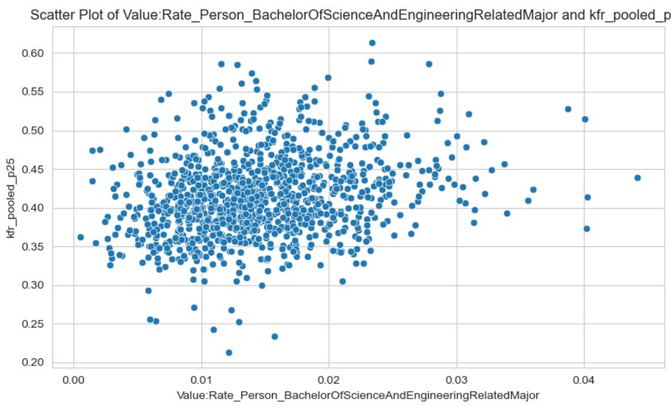


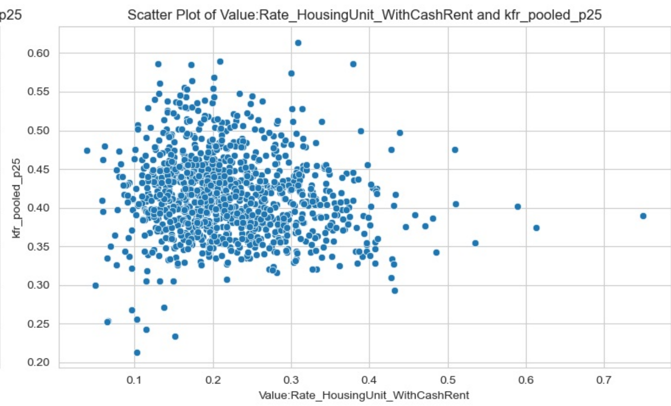












In [69]:
```
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Inspect the missing values
# (A significant portion of the kfr_pooled_p25 values are missing because the other half of the data is in the
# There are more missing values in the VSPIA., while the missing values in other two valuables are minimal
```

```
missing_values_10 = cb_selected_10.isnull().sum()
missing_values_10 = missing_values_10[missing_values_10 > 0]
missing_percentage_10 = (missing_values_10 / len(cb_selected_10)) * 100
missing_df_10 = pd.DataFrame({
    'Missing Values': missing_values_10,
    'Percentage (%)': missing_percentage_10
}).sort_values(by='Percentage (%)', ascending=False)

missing_df_10
```

Out[69]:

	Missing Values	Percentage (%)
kfr_pooled_p25	1117	49.799376
Value:StandardizedPrecipitationIndex_Atmosphere	63	2.808738
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	2	0.089166
Value:Median_Income_Person	1	0.044583

In [70]:

```
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Handle the missing values
# Drop rows with missing kfr_pooled_p25 values
# For the other variables, given the observed concentrations and potential skewness for the majority of the sca

cb_selected_cleaned_10 = cb_selected_10.dropna(subset=['kfr_pooled_p25'])
cb_selected_cleaned_10 = cb_selected_cleaned_10.fillna(cb_selected_cleaned_10.median())
missing_values_final_10 = cb_selected_cleaned_10.isnull().sum()
missing_values_final_10 = missing_values_final_10[missing_values_final_10 > 0]
missing_values_final_10
```

Out[70]:

```
Series([], dtype: int64)
```

In [71]:

```
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Identify potential outliers using Z-score method

df_zscore_10 = cb_selected_cleaned_10.apply(zscore)
outliers_10 = (df_zscore_10.abs() > 3).sum()
outliers_10
```

Out[71]:

```
kfr_pooled_p25                                                   12
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome     12
Value:Median_Age_Person                                           9
Value:Median_Income_Person                                       11
Value:StandardizedPrecipitationIndex_Atmosphere                   9
Value:Rate_Person_BachelorOfArtsHumanitiesAndOtherMajor          20
Value:Rate_Person_BachelorOfBusinessMajor                        19
Value:Rate_Person_BachelorOfEducationMajor                        8
Value:Rate_Person_BachelorOfScienceAndEngineeringMajor           18
Value:Rate_Person_BachelorOfScienceAndEngineeringRelatedMajor     9
Value:Rate_HousingUnit_WithCashRent                               9
dtype: int64
```

In [72]:

```
### Experiment with linear regression (Non-PCA outlier-excluded model)
## Run the linear regression without outliers
# Due to the lack of sufficient context, I simply made the decision by comparing the results from running the l

df_no_outliers_10 = cb_selected_cleaned_10[(df_zscore_10.abs() <= 3).all(axis=1)]
X_10no = df_no_outliers_10.drop(columns='kfr_pooled_p25')
y_10no = df_no_outliers_10['kfr_pooled_p25']
X_10no = sm.add_constant(X_10no)
lin_reg_10no = sm.OLS(y_10no, X_10no).fit()

print(lin_reg_10no.summary())

y_pred_10no = lin_reg_10no.predict(X_10no)

r2_10no = r2_score(y_10no, y_pred_10no)
adj_r2_10no = 1 - (1-r2_10no)*(len(y_10no)-1)/(len(y_10no)-X_10no.shape[1]-1)
mse_10no = mean_squared_error(y_10no, y_pred_10no)
mae_10no = mean_absolute_error(y_10no, y_pred_10no)
rse_10no = np.sqrt(mse_10no)

print(f'R^2: {r2_10no}')
print(f'Adj. R^2: {adj_r2_10no}')
print(f'MSE: {mse_10no}')
print(f'MAE: {mae_10no}')
print(f'RSE: {rse_10no}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:          kfr_pooled_p25   R-squared:                       0.489
Model:                             OLS   Adj. R-squared:                  0.484
Method:                  Least Squares   F-statistic:                     97.17
Date:                 Tue, 05 Dec 2023   Prob (F-statistic):          1.55e-140
Time:                         11:12:31   Log-Likelihood:                 1993.3
No. Observations:                 1026   AIC:                            -3965.
Df Residuals:                     1015   BIC:                            -3910.
Df Model:                           10
Covariance Type:             nonrobust
==============================================================================================================
==================
                                                                 coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------------------------------------
------------------
const                                                          0.2820      0.018     15.634      0.000
0.247       0.317
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome      0.1616      0.022      7.383      0.000
0.119       0.205
Value:Median_Age_Person                                       -0.0012      0.000     -3.667      0.000
-0.002      -0.001
Value:Median_Income_Person                                  5.193e-06   3.35e-07     15.519      0.000    4.
54e-06    5.85e-06
Value:StandardizedPrecipitationIndex_Atmosphere               -0.0034      0.001     -2.637      0.009
-0.006      -0.001
Value:Rate_Person_BachelorOfArtsHumanitiesAndOtherMajor       -0.5325      0.198     -2.688      0.007
-0.921      -0.144
Value:Rate_Person_BachelorOfBusinessMajor                     -1.6374      0.154    -10.645      0.000
-1.939      -1.336
Value:Rate_Person_BachelorOfEducationMajor                     2.6989      0.186     14.518      0.000
2.334       3.064
Value:Rate_Person_BachelorOfScienceAndEngineeringMajor          0.2673      0.123      2.169      0.030
0.025       0.509
Value:Rate_Person_BachelorOfScienceAndEngineeringRelatedMajor    0.7334      0.320      2.295      0.022
0.106       1.361
Value:Rate_HousingUnit_WithCashRent                           -0.0885      0.022     -4.065      0.000
-0.131      -0.046
==============================================================================
Omnibus:                       15.815   Durbin-Watson:                   1.270
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               16.120
Skew:                           0.294   Prob(JB):                     0.000316
Kurtosis:                       3.176   Cond. No.                     7.30e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.3e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
R^2: 0.4891043310378249
Adj. R^2: 0.4835620703291623
MSE: 0.001202422407229398
MAE: 0.02721144057184908
RSE: 0.03467596296037643
```

In [73]:

```python
### Experiment with linear regression (Non-PCA outlier-included model)
## Run the linear regression with outliers
# After experimenting with the two models, one with outliers and one without, there is no significant change in
# Since the outliers have minimal impact on the model, it is acceptable to either remove or retain them
# Therefore, I will conduct further analysis of the set of 10 predictors, including outliers

X2_10with = cb_selected_cleaned_10.drop(columns='kfr_pooled_p25')
y2_10with = cb_selected_cleaned_10['kfr_pooled_p25']
X2_10with = sm.add_constant(X2_10with)
lin_reg_10with = sm.OLS(y2_10with, X2_10with).fit()

print(lin_reg_10with.summary())

y2_pred_10with = lin_reg_10with.predict(X2_10with)

r2_10with = r2_score(y2_10with, y2_pred_10with)
adj_r2_10with = 1 - (1-r2_10with)*(len(y2_10with)-1)/(len(y2_10with)-X2_10with.shape[1]-1)
mse_10with = mean_squared_error(y2_10with, y2_pred_10with)
mae_10with = mean_absolute_error(y2_10with, y2_pred_10with)
rse_10with = np.sqrt(mse_10with)

print(f'R^2: {r2_10with}')
print(f'Adj. R^2: {adj_r2_10with}')
print(f'MSE: {mse_10with}')
print(f'MAE: {mae_10with}')
print(f'RSE: {rse_10with}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         kfr_pooled_p25   R-squared:                       0.502
Model:                            OLS   Adj. R-squared:                  0.498
Method:                 Least Squares   F-statistic:                     112.6
Date:                Tue, 05 Dec 2023   Prob (F-statistic):          2.66e-161
Time:                        11:12:32   Log-Likelihood:                 2120.1
No. Observations:                1126   AIC:                            -4218.
Df Residuals:                    1115   BIC:                            -4163.
Df Model:                          10
Covariance Type:            nonrobust
===============================================================================================================
==================
                                                                     coef    std err          t      P>|t|
[0.025      0.975]
---------------------------------------------------------------------------------------------------------------
------------------
const                                                              0.2713      0.017     16.326      0.000
0.239       0.304
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome       0.1626      0.020      8.031      0.000
0.123       0.202
Value:Median_Age_Person                                           -0.0011      0.000     -3.682      0.000
-0.002      -0.001
Value:Median_Income_Person                                      5.693e-06   3.15e-07     18.071      0.000      5.
08e-06    6.31e-06
Value:StandardizedPrecipitationIndex_Atmosphere                   -0.0029      0.001     -2.397      0.017
-0.005      -0.001
Value:Rate_Person_BachelorOfArtsHumanitiesAndOtherMajor           -0.3953      0.170     -2.328      0.020
-0.729      -0.062
Value:Rate_Person_BachelorOfBusinessMajor                         -1.6816      0.129    -13.033      0.000
-1.935      -1.428
Value:Rate_Person_BachelorOfEducationMajor                         2.3793      0.178     13.404      0.000
2.031       2.728
Value:Rate_Person_BachelorOfScienceAndEngineeringMajor             0.2039      0.103      1.988      0.047
0.003       0.405
Value:Rate_Person_BachelorOfScienceAndEngineeringRelatedMajor      0.6639      0.302      2.197      0.028
0.071       1.257
Value:Rate_HousingUnit_WithCashRent                               -0.0596      0.019     -3.066      0.002
-0.098      -0.021
==============================================================================
Omnibus:                       35.681   Durbin-Watson:                   1.308
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               44.196
Skew:                           0.357   Prob(JB):                     2.53e-10
Kurtosis:                       3.658   Cond. No.                     6.98e+06
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.98e+06. This might indicate that there are
strong multicollinearity or other numerical problems.
R^2: 0.5024121568397615
Adj. R^2: 0.4974988118893462
MSE: 0.0013554769765591547
MAE: 0.02850761991620012
RSE: 0.03681680291061616
```

In [74]:

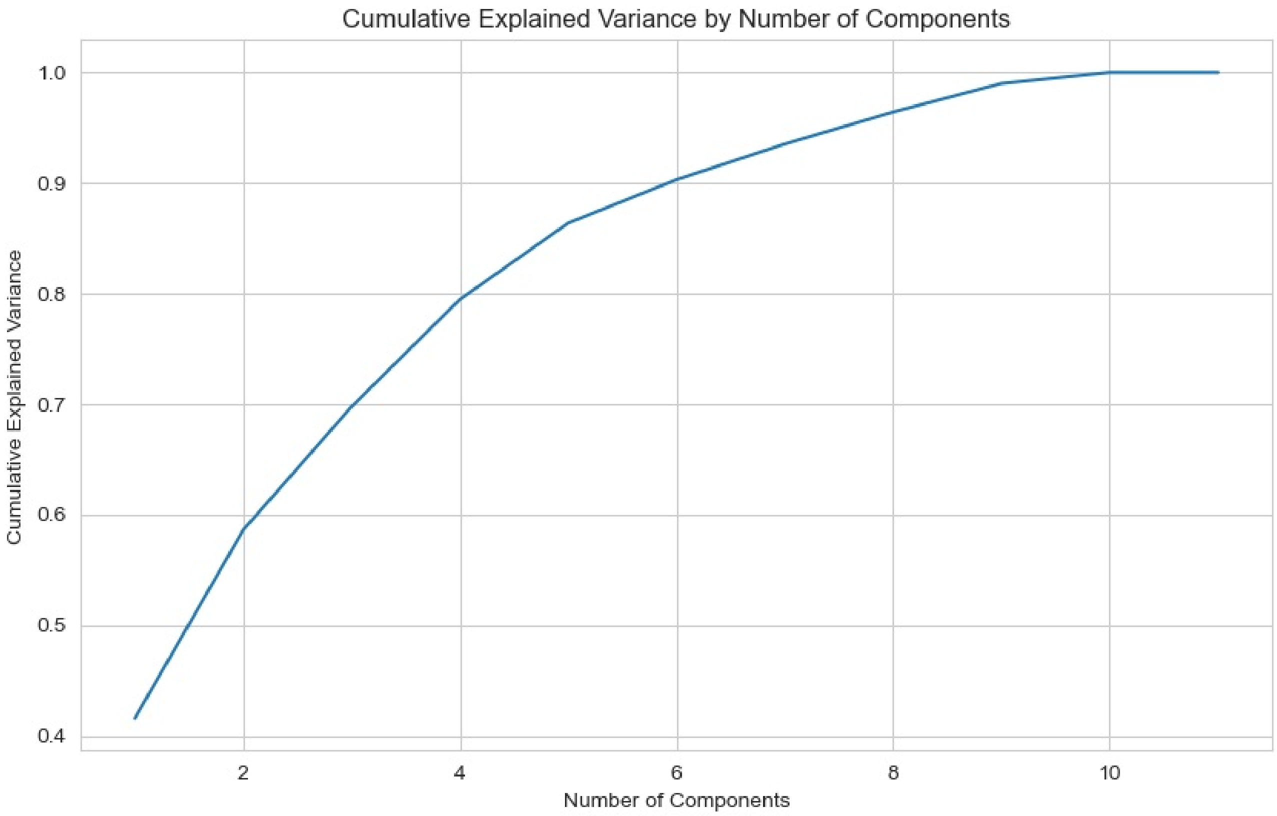
```
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Inspect multicollinearity using VIF method
# While the 'const' term has a very high VIF, this might not be a problem depending on the context
# For the other variables, the VIFs are mostly below 10, with only a couple of variables slightly above 5

X_const_10 = sm.add_constant(cb_selected_cleaned_10)

vif_data_10 = pd.DataFrame()
vif_data_10["feature"] = X_const_10.columns
vif_data_10["VIF"] = [variance_inflation_factor(X_const_10.values, i) for i in range(len(X_const_10.columns))]

vif_data_sorted_10 = vif_data_10.sort_values(by="VIF", ascending=False)
vif_data_sorted_10.head(10)
```

Out[74]:

	feature	VIF
0	const	281.485329
9	Value:Rate_Person_BachelorOfScienceAndEngineer...	5.948565
6	Value:Rate_Person_BachelorOfArtsHumanitiesAndO...	5.767734
7	Value:Rate_Person_BachelorOfBusinessMajor	3.260051
4	Value:Median_Income_Person	2.862011
10	Value:Rate_Person_BachelorOfScienceAndEngineer...	2.852620
11	Value:Rate_HousingUnit_WithCashRent	2.047308
1	kfr_pooled_p25	2.009695
3	Value:Median_Age_Person	1.747693
8	Value:Rate_Person_BachelorOfEducationMajor	1.676368

In [75]:

```
### Get a sense of the underlying structure of the set of 10 predictors and preprocess the data
## Standardize the variables and Apply PCA
# I didn't choose to remove variables (such as by using feature engineering techniques) as the questions did no
# Another way to enhance predictions and address multicollinearity is through Principal Component Analysis (PCA
# While PCA can reduce a model's interpretability and does not always enhance model performance, let's try it t

scaler = StandardScaler()
X_scaled_10 = scaler.fit_transform(X2_10with)

pca = PCA()
X_pca_10 = pca.fit_transform(X_scaled_10)

cumulative_variance_10 = np.cumsum(pca.explained_variance_ratio_)
n_components_95 = np.where(cumulative_variance_10 >= 0.95)[0][0] + 1

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_variance_10) + 1), cumulative_variance_10)
plt.title('Cumulative Explained Variance by Number of Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.show()

n_components_95, cumulative_variance_10[:n_components_95]
```



Out[75]:

```
(8,
 array([0.41557248, 0.58589669, 0.69747615, 0.79456217, 0.86366344,
        0.90295289, 0.93522157, 0.96417871]))
```

In [76]:

```
### Experiment with linear regression (PCA model with outliers included)
## Make predictions using the first 8 principal components that explain at least 95% of the variance
# Based on pre-PCA and post-PCA metrics, the PCA doesn't enhance the model performance, though the condition nu
# The decrease in both R-squared and Adj. R-squared alongside increased error metrics (MSE, MAE, and RSE) sugge
# the principal components used don't capture enough of the variability in the data that is relevant for predic
# While a lower condition number is indicative of a model that has potentially addressed multicollinearity, it
# The final choice of model should be based on a holistic evaluation of model performance
# Hence, the Non-PCA outlier-included model statistically outperforms the PCA model with outliers included

X_pca_reduced_10 = X_pca_10[:, :n_components_95]
y3_10 = cb_selected_cleaned_10['kfr_pooled_p25'].values
X_pca_reduced_10 = sm.add_constant(X_pca_reduced_10)
lin_reg_pca_10 = sm.OLS(y3_10, X_pca_reduced_10).fit()

print(lin_reg_pca_10.summary())

y3_pred_10 = lin_reg_pca_10.predict(X_pca_reduced_10)

r2_pca_10 = r2_score(y3_10, y3_pred_10)
adj_r2_pca_10 = 1 - (1-r2_pca_10)*(len(y3_10)-1)/(len(y3_10)-X_pca_reduced_10.shape[1]-1)
mse_pca_10 = mean_squared_error(y3_10, y3_pred_10)
mae_pca_10 = mean_absolute_error(y3_10, y3_pred_10)
rse_pca_10 = np.sqrt(mse_pca_10)

print(f'R^2: {r2_pca_10}')
print(f'Adj. R^2: {adj_r2_pca_10}')
```

```
print(f'MSE: {mse_pca_10}')
print(f'MAE: {mae_pca_10}')
print(f'RSE: {rse_pca_10}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.442
Model:                            OLS   Adj. R-squared:                  0.438
Method:                 Least Squares   F-statistic:                     110.7
Date:                Tue, 05 Dec 2023   Prob (F-statistic):          6.03e-136
Time:                        11:12:35   Log-Likelihood:                 2055.9
No. Observations:                1126   AIC:                            -4094.
Df Residuals:                    1117   BIC:                            -4049.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4145      0.001    355.415      0.000       0.412       0.417
x1             0.0060      0.001     10.493      0.000       0.005       0.007
x2             0.0137      0.001     15.321      0.000       0.012       0.015
x3            -0.0205      0.001    -18.566      0.000      -0.023      -0.018
x4             0.0077      0.001      6.473      0.000       0.005       0.010
x5             0.0049      0.001      3.494      0.000       0.002       0.008
x6            -0.0035      0.002     -1.898      0.058      -0.007       0.000
x7            -0.0228      0.002    -11.103      0.000      -0.027      -0.019
x8             0.0085      0.002      3.905      0.000       0.004       0.013
==============================================================================
Omnibus:                       34.555   Durbin-Watson:                   1.237
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               44.970
Skew:                           0.331   Prob(JB):                     1.72e-10
Kurtosis:                       3.721   Cond. No.                         3.79
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R^2: 0.4422813886231114
Adj. R^2: 0.437783657886201
MSE: 0.0015192789524732582
MAE: 0.030390644358175338
RSE: 0.03897792904289886
```

Part 2: Prediction Challenge

8. Run a linear regression of krf_pooled_p25 on the full predictor set (consisting of the 10 predictors you chose from DataCommons and the 121 predictors included in the training data). Interpret one of the coefficients. Obtain predictions of kfr_pooled_p25.

After experimenting with several linear regression models — with and without outliers, and with and without PCA — as shown in the code blocks and outputs below, I chose non-PCA outlier-excluded model to inspect the results.

The OLS regression results indicate:

- R-squared (0.878)
- Adj. R-squared (0.860)
- F-statistic (49.03)
- Coefficients (Top 5 significant positive predictors include `P_26` Fraction of Residents w/ a College Degree or More in 2000, `P_32` Share Below Poverty Line 2000, `Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome`, `P_47` Employment Rate 2000, `P_46` Share of Working Adults w/ Commute Time of 15 Minutes Or Less in 2006-2010 ACS. Top 5 significant negative predictors are the `Value:Rate_Person_BachelorOfScienceAndEngineeringMajor`, `P_45` Share of Single-Headed Households with Children 2000, `P_49` Log wage growth for HS Grad., 2005-2014, `P_56` Mentally Unhealthy Days per Month (Persons 18 Years and Over), `P_10` % of Individuals Earning < 138% of the FPL without Insurance in 2013. The presence of multicollinearity suggests caution in interpreting individual coefficients. Further analysis would be required to address this issue.)
- P-values (a number of predictors have p-values greater than 0.05)
- Durbin-Watson statistic (1.908)
- Skew (-0.015)
- Kurtosis (3.245)

Comments:

Statistical Significance

- The F-statistic of 49.03 with an associated p-value of essentially 0 suggests that the statistical test has found strong evidence to indicate that the group means are not all equal. It implies that the differences observed are statistically significant and unlikely to be due to random chance.

Goodness-of-Fit

- The R-squared value is 0.878, which means that approximately 87.8% of the variance in the dependent variable can be explained by the model. This is a high R-squared value and suggests a good fit. The adjusted R-squared value is 0.860, which adjusts for the number of predictors in the model and is also high, confirming that the model fits the data well.

Predictive Power

- A high R-squared value suggests that the model has good predictive power. However, the true test of predictive power is how well the model performs on new, unseen data.

Limitations

- The large condition number (1.75e+09), suggesting potential multicollinearity, can make the interpretation of individual coefficients problematic.
- The model includes many predictors (131), which is a lot more than the PCA model. This increases the risk of overfitting and may reduce the model's generalizability. With many predictors, the risk of Type I error (false positives) increases, and some predictors may appear significant by chance.

Practical Implications

- Statistically significant predictors with larger coefficients might be areas where policy interventions or further research could be focused to understand their impact on kfr_pooled_p25. For example, a significant positive coefficient for P_26 suggests that increasing the proportion of residents with a college degree could have a positive impact on the dependent variable. Similarly, a significant negative coefficient for P_45 suggests that reducing the share of single-headed households with children might be associated with an increase in the dependent variable. It is also important to consider the practical significance of the predictors, which involves understanding the actual impact in the real-world context, not just whether an effect exists statistically.
- While the model shows a good fit statistically, one must be cautious about its practical application due to potential multicollinearity and overfitting. Further model diagnostics, validation on test data, and consideration of practical significance are necessary to ensure the robustness and applicability of the model's findings.

Here are the in-sample predictions of kfr_pooled_p25 using the full set for the first five observations:

- [0.371851 0.413179 0.396186 0.360396 0.368280]

In [77]:
```
### Prepare the data for the full set

value_columns = [col for col in cb.columns if "Value:" in col]
p_columns = [col for col in cb.columns if "P_" in col]
selected_columns = ['kfr_pooled_p25'] + value_columns + p_columns
cb_selected = cb[selected_columns]

cb_selected.describe()
```

Out[77]:

	kfr_pooled_p25	Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	Value:Median_Age_Person	Value:Median_Income_Per
count	1126.000000	2241.000000	2243.000000	2242.000
mean	0.414503	0.232215	40.091306	24117.000
std	0.052216	0.061199	4.790646	5492.881
min	0.212865	-0.160444	22.300000	9399.000
25%	0.379837	0.196398	37.400000	20782.250
50%	0.411379	0.232735	40.300000	23626.500
75%	0.443173	0.268950	42.900000	26855.500
max	0.614030	0.501838	65.300000	61012.000

8 rows × 132 columns

In [78]:
```
### Get a sense of the underlying structure of the full set and preprocess the data
## Inspect the missing values
# (A significant portion of the kfr_pooled_p25 values are missing because the other half of the data is in the
# There are more missing values in the VSPIA., while the missing values in other two valuables are minimal

missing_values = cb_selected.isnull().sum()
missing_values = missing_values[missing_values > 0]
missing_percentage = (missing_values / len(cb_selected)) * 100
missing_df = pd.DataFrame({
    'Missing Values': missing_values,
    'Percentage (%)': missing_percentage
}).sort_values(by='Percentage (%)', ascending=False)

missing_df
```

Out[78]:

	Missing Values	Percentage (%)
kfr_pooled_p25	1117	49.799376
Value:StandardizedPrecipitationIndex_Atmosphere	63	2.808738
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	2	0.089166
Value:Median_Income_Person	1	0.044583

In [79]:

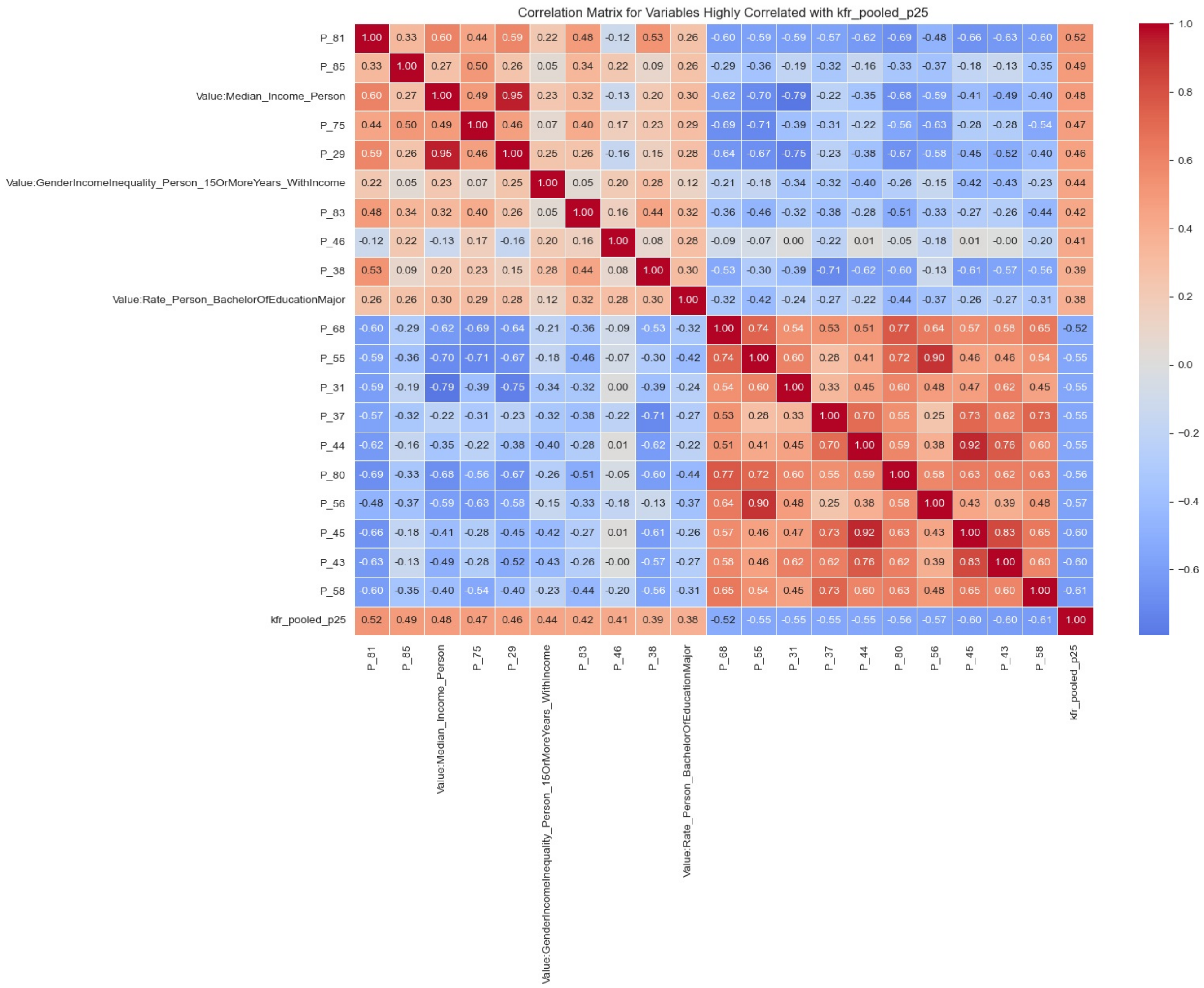
```
### Get a sense of the underlying structure of the full set and preprocess the data
## Handle the missing values
# Drop rows with missing kfr_pooled_p25 values
# For the other variables, given the observed concentrations and potential skewness for the majority of the sca

cb_selected_cleaned = cb_selected.dropna(subset=['kfr_pooled_p25'])
cb_selected_cleaned = cb_selected_cleaned.fillna(cb_selected.median())
missing_values_final = cb_selected_cleaned.isnull().sum()
missing_values_final = missing_values_final[missing_values_final > 0]
missing_values_final
```

Out[79]:

```
Series([], dtype: int64)
```

In [80]:

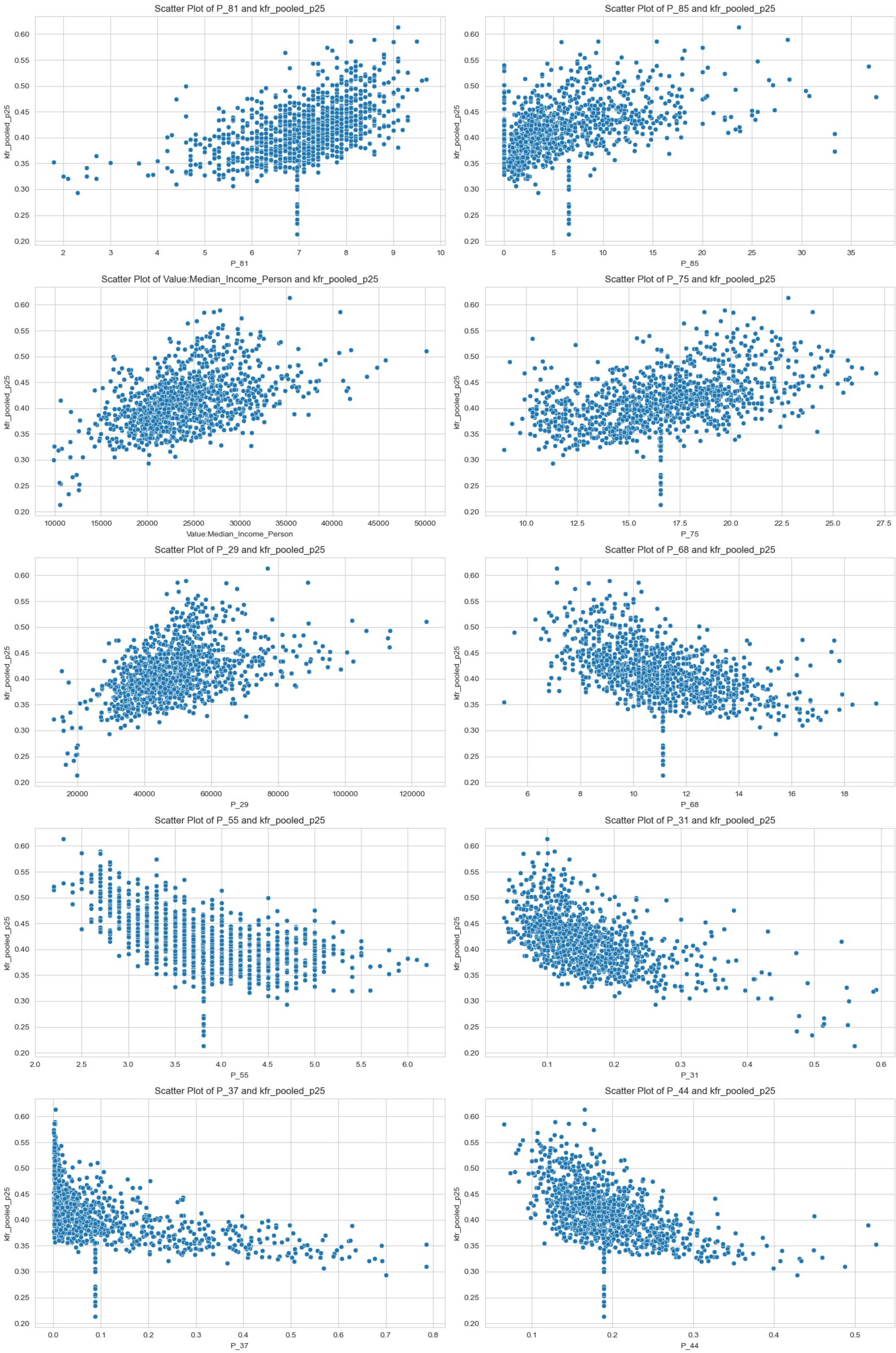
```
### Get a sense of the underlying structure of the full set and preprocess the data
## Inspect the correlation coefficient and identify multicollinearity

correlation_matrix = cb_selected_cleaned.corr()
correlations_with_target = correlation_matrix['kfr_pooled_p25'].sort_values(ascending=False)
top_10_corr = correlations_with_target.head(11)[1:]
bottom_10_corr = correlations_with_target.tail(10)
selected_vars = top_10_corr.index.tolist() + bottom_10_corr.index.tolist() + ['kfr_pooled_p25']
selected_corr_matrix = cb_selected_cleaned[selected_vars].corr()
plt.figure(figsize=(15, 10))
sns.heatmap(selected_corr_matrix, annot=True, cmap='coolwarm', center=0, linewidths=.5, fmt=".2f")
plt.title('Correlation Matrix for Variables Highly Correlated with kfr_pooled_p25')
plt.show()
```



In [81]:

```
### Get a sense of the underlying structure of the full set and preprocess the data
## Assess the linearity of the relationships of top 5 positively and top 5 negatively correlated variables

selected_vars = top_10_corr.index.tolist()[:5] + bottom_10_corr.index.tolist()[:5]
fig, axes = plt.subplots(5, 2, figsize=(16, 24))
```

```
axes = axes.flatten()

for i, var in enumerate(selected_vars):
    sns.scatterplot(x=var, y='kfr_pooled_p25', data=cb_selected_cleaned, ax=axes[i])
    axes[i].set_title(f'Scatter Plot of {var} and kfr_pooled_p25')
    axes[i].set_xlabel(var)
    axes[i].set_ylabel('kfr_pooled_p25')

plt.tight_layout()
plt.show()
```



In [82]:

```
### Get a sense of the underlying structure of the full set and preprocess the data
## Identify potential outliers using Z-score method

df_zscore = cb_selected_cleaned[selected_vars + ['kfr_pooled_p25']].apply(zscore)
outliers = (df_zscore.abs() > 3).sum()
```

```
outliers
```

Out[82]:

```
P_81                          12
P_85                          22
Value:Median_Income_Person    11
P_75                           1
P_29                          16
P_68                           7
P_55                           6
P_31                          23
P_37                          27
P_44                          15
kfr_pooled_p25                12
dtype: int64
```

In [84]:

```
### Experiment with linear regression (Non-PCA outlier-excluded model)
## Run the linear regression without outliers
# Due to the lack of sufficient context, I simply made the decision by comparing the results from running the l

df_no_outliers = cb_selected_cleaned[(df_zscore.abs() <= 3).all(axis=1)]
value_columns = [col for col in df_no_outliers.columns if "Value:" in col]
p_columns = [col for col in df_no_outliers.columns if "P_" in col]
selected_columns = value_columns + p_columns

X = df_no_outliers[selected_columns]
y = df_no_outliers['kfr_pooled_p25']
X = sm.add_constant(X)
lin_reg = sm.OLS(y, X).fit()

print(lin_reg.summary())

y_pred = lin_reg.predict(X)

r2_no_outliers = r2_score(y, y_pred)
adj_r2_no_outliers = 1 - (1-r2_no_outliers)*(len(y)-1)/(len(y)-X.shape[1]-1)
mse_no_outliers = mean_squared_error(y, y_pred)
mae_no_outliers = mean_absolute_error(y, y_pred)
rse_no_outliers = np.sqrt(mse_no_outliers)

print(f'R^2: {r2_no_outliers}')
print(f'Adj. R^2: {adj_r2_no_outliers}')
print(f'MSE: {mse_no_outliers}')
print(f'MAE: {mae_no_outliers}')
print(f'RSE: {rse_no_outliers}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         kfr_pooled_p25   R-squared:                       0.878
Model:                            OLS   Adj. R-squared:                  0.860
Method:                 Least Squares   F-statistic:                     49.03
Date:                Tue, 05 Dec 2023   Prob (F-statistic):               0.00
Time:                        11:13:03   Log-Likelihood:                 2756.3
No. Observations:                1021   AIC:                            -5249.
Df Residuals:                     889   BIC:                            -4598.
Df Model:                         131
Covariance Type:            nonrobust
=====================================================================================================================
==================
                                                                      coef    std err          t      P>|t|
[0.025      0.975]
---------------------------------------------------------------------------------------------------------------------
------------------
const                                                               0.3067     10.315      0.030      0.976      -
19.939      20.552
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome        0.0684      0.014      4.954      0.000
0.041       0.095
Value:Median_Age_Person                                             0.0011      0.000      2.529      0.012
0.000       0.002
Value:Median_Income_Person                                       2.397e-06   4.92e-07      4.874      0.000     1.
43e-06    3.36e-06
Value:StandardizedPrecipitationIndex_Atmosphere                    -0.0002      0.001     -0.231      0.817
-0.002       0.001
Value:Rate_Person_BachelorOfArtsHumanitiesAndOtherMajor            -0.1988      0.115     -1.725      0.085
-0.425       0.027
Value:Rate_Person_BachelorOfBusinessMajor                          -0.0966      0.112     -0.865      0.387
-0.316       0.123
Value:Rate_Person_BachelorOfEducationMajor                          0.1200      0.125      0.957      0.339
-0.126       0.366
Value:Rate_Person_BachelorOfScienceAndEngineeringMajor             -0.4358      0.090     -4.849      0.000
-0.612      -0.259
Value:Rate_Person_BachelorOfScienceAndEngineeringRelatedMajor      -0.0814      0.185     -0.441      0.660
-0.444       0.281
Value:Rate_HousingUnit_WithCashRent                                -0.0076      0.019     -0.409      0.683
-0.044       0.029
P_1                                                                -0.0002   8.21e-05     -2.203      0.028
-0.000   -1.98e-05
P_2                                                             -4.508e-06    9.1e-05     -0.050      0.961
-0.000       0.000
P_3                                                               1.54e-05   9.28e-05      0.166      0.868
```

```
-0.000       0.000
P_4                                                              9.577e-05      0.000      0.919      0.358
-0.000       0.000
P_5                                                             -9.633e-05      0.000     -0.864      0.388
-0.000       0.000
P_6                                                             -3.591e-05      0.000     -0.300      0.765
-0.000       0.000
P_7                                                              1.721e-05   7.14e-05      0.241      0.809
-0.000       0.000
P_8                                                              6.387e-05      0.000      0.489      0.625
-0.000       0.000
P_9                                                              7.398e-05      0.000      0.620      0.535
-0.000       0.000
P_10                                                               -0.0065      0.003     -2.124      0.034
-0.012      -0.000
P_11                                                               -0.0018      0.004     -0.475      0.635
-0.009       0.006
P_12                                                               -0.0004      0.001     -0.761      0.447
-0.001       0.001
P_13                                                                0.0007      0.001      1.170      0.242
-0.000       0.002
P_14                                                                0.0004      0.001      0.703      0.482
-0.001       0.001
P_15                                                                0.0004      0.001      0.800      0.424
-0.001       0.002
P_16                                                                0.0004      0.001      0.770      0.441
-0.001       0.001
P_17                                                                0.0002      0.001      0.172      0.864
-0.002       0.002
P_18                                                                0.0002      0.001      0.370      0.712
-0.001       0.002
P_19                                                                0.0002      0.001      0.379      0.705
-0.001       0.002
P_20                                                                0.0003      0.001      0.440      0.660
-0.001       0.002
P_21                                                                0.0003      0.000      0.543      0.588
-0.001       0.001
P_22                                                               -0.0002      0.000     -0.497      0.619
-0.001       0.001
P_23                                                               -0.0003      0.000     -0.554      0.579
-0.001       0.001
P_24                                                            -1.277e-06   2.52e-07     -5.070      0.000     -1.
77e-06   -7.83e-07
P_25                                                               -0.0002      0.000     -0.759      0.448
-0.001       0.000
P_26                                                                0.2208      0.044      4.963      0.000
0.133       0.308
P_27                                                               -0.0610      0.042     -1.442      0.150
-0.144       0.022
P_28                                                                0.0474      0.034      1.404      0.161
-0.019       0.114
P_29                                                             5.097e-07   2.61e-07      1.956      0.051     -1.
61e-09    1.02e-06
P_30                                                            -1.828e-06   4.93e-07     -3.709      0.000      -2
.8e-06   -8.61e-07
P_31                                                               -0.0188      0.033     -0.576      0.565
-0.083       0.045
P_32                                                                0.2049      0.042      4.842      0.000
0.122       0.288
P_33                                                               -0.0742      0.038     -1.931      0.054
-0.150       0.001
P_34                                                               -0.0986      0.162     -0.608      0.544
-0.417       0.220
P_35                                                               -0.2135      0.156     -1.372      0.171
-0.519       0.092
P_36                                                                0.0517      0.261      0.198      0.843
-0.461       0.564
P_37                                                                0.1400      0.174      0.804      0.422
-0.202       0.482
P_38                                                                0.2165      0.171      1.267      0.205
-0.119       0.552
P_39                                                                0.2100      0.171      1.232      0.218
-0.125       0.545
P_40                                                                0.1167      0.287      0.407      0.684
-0.446       0.680
P_41                                                               -0.0005      0.001     -0.413      0.680
-0.003       0.002
P_42                                                              -6.11e-06   8.95e-06     -0.683      0.495     -2.
37e-05    1.15e-05
P_43                                                               -0.0277      0.016     -1.722      0.085
-0.059       0.004
P_44                                                               -0.0091      0.031     -0.294      0.769
-0.070       0.052
P_45                                                               -0.2229      0.036     -6.139      0.000
-0.294      -0.152
P_46                                                                0.0379      0.012      3.144      0.002
0.014       0.062
P_47                                                                0.0576      0.019      2.993      0.003
0.020       0.095
```

P_48	1.618e-05	0.000	0.110	0.912	-0.000	0.000
P_49	-0.0182	0.006	-2.882	0.004	-0.031	-0.006
P_50	0.1785	0.160	1.118	0.264	-0.135	0.492
P_51	-1.172e-06	8.29e-06	-0.141	0.888	-1.75e-05	1.51e-05
P_52	-1.001e-05	4.07e-06	-2.463	0.014	-1.8e-05	-2.03e-06
P_53	-0.0485	0.065	-0.747	0.455	-0.176	0.079
P_54	3.104e-05	2.31e-05	1.344	0.179	-1.43e-05	7.64e-05
P_55	-0.0058	0.005	-1.213	0.226	-0.015	0.004
P_56	-0.0157	0.004	-4.246	0.000	-0.023	-0.008
P_57	0.0029	0.001	3.507	0.000	0.001	0.005
P_58	-0.0012	0.001	-1.754	0.080	-0.003	0.000
P_59	4.788e-05	3.16e-05	1.513	0.131	-1.42e-05	0.000
P_60	7.995e-06	6.42e-06	1.245	0.213	-4.61e-06	2.06e-05
P_61	4.708e-05	3.6e-05	1.307	0.191	-2.36e-05	0.000
P_62	8.152e-07	7.11e-07	1.147	0.252	-5.8e-07	2.21e-06
P_63	0.0036	0.001	3.591	0.000	0.002	0.006
P_64	0.0046	0.002	2.083	0.038	0.000	0.009
P_65	-0.0041	0.004	-0.988	0.324	-0.012	0.004
P_66	-2.439e-05	1.7e-05	-1.435	0.152	-5.77e-05	8.96e-06
P_67	-5.236e-05	0.000	-0.473	0.637	-0.000	0.000
P_68	0.0011	0.001	0.832	0.406	-0.001	0.004
P_69	-0.0002	0.000	-1.136	0.256	-0.000	0.000
P_70	-4.761e-08	3.02e-07	-0.158	0.875	-6.4e-07	5.45e-07
P_71	-0.0003	8.52e-05	-3.728	0.000	-0.000	-0.000
P_72	-3.559e-06	6.24e-06	-0.571	0.568	-1.58e-05	8.68e-06
P_73	3.32e-06	3.88e-06	0.857	0.392	-4.29e-06	1.09e-05
P_74	-0.0002	0.000	-0.427	0.669	-0.001	0.001
P_75	0.0012	0.000	3.035	0.002	0.000	0.002
P_76	0.0007	0.000	2.686	0.007	0.000	0.001
P_77	-6.404e-05	4.52e-05	-1.416	0.157	-0.000	2.47e-05
P_78	-0.0001	0.000	-0.421	0.674	-0.001	0.000
P_79	0.0001	0.000	0.469	0.639	-0.000	0.001
P_80	-0.0002	0.000	-2.317	0.021	-0.000	-3.78e-05
P_81	0.0040	0.002	1.866	0.062	-0.000	0.008
P_82	0.0938	0.065	1.433	0.152	-0.035	0.222
P_83	0.0940	0.065	1.436	0.151	-0.034	0.222
P_84	0.0924	0.065	1.412	0.158	-0.036	0.221
P_85	0.0949	0.065	1.450	0.147	-0.034	0.223
P_86	0.0969	0.065	1.481	0.139	-0.032	0.225
P_87	0.0941	0.065	1.439	0.151	-0.034	0.223
P_88	0.0918	0.066	1.399	0.162	-0.037	0.221
P_89	0.0911	0.066	1.389	0.165	-0.038	0.220
P_90	0.0970	0.066	1.480	0.139	-0.032	0.226
P_91	0.0951	0.066	1.451	0.147	-0.034	0.224
P_92	0.0941	0.065	1.438	0.151		

```
-0.034       0.223
P_93                                                                      0.0937      0.065      1.431      0.153
-0.035       0.222
P_94                                                                     -0.0945      0.080     -1.189      0.235
-0.251       0.062
P_95                                                                     -0.0945      0.080     -1.188      0.235
-0.251       0.062
P_96                                                                     -0.0940      0.080     -1.182      0.237
-0.250       0.062
P_97                                                                     -0.0945      0.080     -1.188      0.235
-0.251       0.062
P_98                                                                     -0.0964      0.080     -1.210      0.227
-0.253       0.060
P_99                                                                     -0.0943      0.079     -1.186      0.236
-0.250       0.062
P_100                                                                    -0.0905      0.080     -1.130      0.259
-0.248       0.067
P_101                                                                    -0.0921      0.080     -1.159      0.247
-0.248       0.064
P_102                                                                    -0.0979      0.079     -1.234      0.218
-0.254       0.058
P_103                                                                    -0.0946      0.079     -1.190      0.234
-0.251       0.061
P_104                                                                    -0.0948      0.080     -1.192      0.234
-0.251       0.061
P_105                                                                    -0.0943      0.080     -1.185      0.236
-0.250       0.062
P_106                                                                    -0.0009      0.001     -1.148      0.251
-0.002       0.001
P_107                                                                    -0.0012      0.001     -1.320      0.187
-0.003       0.001
P_108                                                                     0.0007      0.001      0.936      0.349
-0.001       0.002
P_109                                                                  4.808e-05      0.000      0.197      0.844
-0.000       0.001
P_110                                                                     0.0021      0.003      0.823      0.411
-0.003       0.007
P_111                                                                 -4.455e-05      0.000     -0.106      0.916
-0.001       0.001
P_112                                                                     0.0007      0.000      2.344      0.019
0.000       0.001
P_113                                                                    -0.0002   7.44e-05     -2.071      0.039
-0.000   -8.03e-06
P_114                                                                    -0.0001      0.000     -0.295      0.768
-0.001       0.001
P_115                                                                    -0.0005      0.000     -2.088      0.037
-0.001   -3.09e-05
P_116                                                                    -0.0015      0.001     -1.803      0.072
-0.003       0.000
P_117                                                                    -0.0003      0.000     -2.699      0.007
-0.001   -8.03e-05
P_118                                                                     0.0013      0.000      3.649      0.000
0.001       0.002
P_119                                                                    -0.0003      0.000     -1.384      0.167
-0.001       0.000
P_120                                                                     0.0003      0.000      1.234      0.217
-0.000       0.001
P_121                                                                    -0.0010      0.001     -1.228      0.220
-0.003       0.001
==============================================================================
Omnibus:                        2.465   Durbin-Watson:                   1.908
Prob(Omnibus):                  0.292   Jarque-Bera (JB):                2.593
Skew:                          -0.015   Prob(JB):                        0.273
Kurtosis:                       3.245   Cond. No.                     1.75e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.75e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
R^2: 0.8784093350136011
Adj. R^2: 0.8603350469750823
MSE: 0.0002646352592454583
MAE: 0.012728553152923592
RSE: 0.01626761381535283
```

In [85]:

```python
## Obtain the most significant coefficients for the final analysis model in the full set

coefs = lin_reg.params
p_values = lin_reg.pvalues
coefs_p_values = pd.DataFrame({'coef': coefs, 'p_value': p_values}).drop('const')
significant_coefs = coefs_p_values[coefs_p_values['p_value'] < 0.05]
top_pos_coefs = significant_coefs[significant_coefs['coef'] > 0].sort_values(by='coef', ascending=False).head(5
top_neg_coefs = significant_coefs[significant_coefs['coef'] < 0].sort_values(by='coef', ascending=True).head(5)

print("Top 5 Positive Significant Coefficients:")
print(top_pos_coefs)
print("\nTop 5 Negative Significant Coefficients:")
print(top_neg_coefs)
```

```
Top 5 Positive Significant Coefficients:
                                                        coef       p_value
P_26                                                0.220811  8.318581e-07
P_32                                                0.204944  1.514107e-06
Value:GenderIncomeInequality_Person_15OrMoreYea...  0.068383  8.709187e-07
P_47                                                0.057621  2.841712e-03
P_46                                                0.037917  1.724048e-03

Top 5 Negative Significant Coefficients:
                                                        coef       p_value
Value:Rate_Person_BachelorOfScienceAndEngineeri... -0.435819  1.461405e-06
P_45                                               -0.222863  1.247271e-09
P_49                                               -0.018195  4.047459e-03
P_56                                               -0.015676  2.403265e-05
P_10                                               -0.006459  3.391754e-02
```

In [86]:
```
## Obtain predictions of kfr_pooled_p25 for the final analysis model in the full set

print(y_pred[:5])
```

```
0    0.371851
1    0.413179
2    0.396186
3    0.360396
4    0.368280
dtype: float64
```

In [87]:
```
### Experiment with linear regression (Non-PCA outlier-included model)
## Run the linear regression with outliers
# After experimenting with the two models, one with outliers and one without, model performance was enhanced af
# Therefore, I will conduct further analysis without outliers

value_columns_outliers = [col for col in cb_selected_cleaned.columns if "Value:" in col]
p_columns_outliers = [col for col in cb_selected_cleaned.columns if "P_" in col]
selected_columns_outliers = value_columns_outliers + p_columns_outliers

X2 = cb_selected_cleaned[selected_columns]
y2 = cb_selected_cleaned['kfr_pooled_p25']
X2 = sm.add_constant(X2)
lin_reg2 = sm.OLS(y2, X2).fit()

print(lin_reg2.summary())

y2_pred = lin_reg2.predict(X2)

r2_with = r2_score(y2, y2_pred)
adj_r2_with = 1 - (1-r2_with)*(len(y2)-1)/(len(y2)-X2.shape[1]-1)
mse_with = mean_squared_error(y2, y2_pred)
mae_with = mean_absolute_error(y2, y2_pred)
rse_with = np.sqrt(mse_with)

print(f'R^2: {r2_with}')
print(f'Adj. R^2: {adj_r2_with}')
print(f'MSE: {mse_with}')
print(f'MAE: {mae_with}')
print(f'RSE: {rse_with}')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:         kfr_pooled_p25   R-squared:                       0.878
Model:                            OLS   Adj. R-squared:                  0.862
Method:                 Least Squares   F-statistic:                     54.77
Date:                Tue, 05 Dec 2023   Prob (F-statistic):               0.00
Time:                        11:13:07   Log-Likelihood:                 2913.0
No. Observations:                1126   AIC:                            -5562.
Df Residuals:                     994   BIC:                            -4899.
Df Model:                         131
Covariance Type:            nonrobust
==============================================================================================================
==================
                                                                   coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------------------------------------
------------------
const                                                            1.8903     10.939      0.173      0.863     -
19.575      23.356
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome      0.0534      0.014      3.828      0.000
0.026       0.081
Value:Median_Age_Person                                           0.0012      0.000      2.597      0.010
0.000       0.002
Value:Median_Income_Person                                     2.026e-06   5.13e-07      3.950      0.000    1.
02e-06    3.03e-06
Value:StandardizedPrecipitationIndex_Atmosphere                  -0.0003      0.001     -0.383      0.701
-0.002       0.001
Value:Rate_Person_BachelorOfArtsHumanitiesAndOtherMajor          -0.2683      0.119     -2.254      0.024
-0.502      -0.035
Value:Rate_Person_BachelorOfBusinessMajor                        -0.1545      0.113     -1.363      0.173
-0.377       0.068
Value:Rate_Person_BachelorOfEducationMajor                        0.0063      0.131      0.048      0.962
```

```
-0.250       0.262
Value:Rate_Person_BachelorOfScienceAndEngineeringMajor          -0.3393      0.092     -3.685      0.000
-0.520      -0.159
Value:Rate_Person_BachelorOfScienceAndEngineeringRelatedMajor    0.0694      0.191      0.363      0.717
-0.306       0.444
Value:Rate_HousingUnit_WithCashRent                              0.0004      0.019      0.023      0.982
-0.037       0.038
P_1                                                             -0.0002   8.63e-05     -2.567      0.010
-0.000   -5.21e-05
P_2                                                           6.218e-05   9.64e-05      0.645      0.519
-0.000       0.000
P_3                                                            2.21e-05   9.95e-05      0.222      0.824
-0.000       0.000
P_4                                                           7.774e-05      0.000      0.702      0.483
-0.000       0.000
P_5                                                             -0.0002      0.000     -1.764      0.078
-0.000    2.36e-05
P_6                                                           7.417e-05      0.000      0.591      0.555
-0.000       0.000
P_7                                                           9.423e-06   7.69e-05      0.123      0.902
-0.000       0.000
P_8                                                          -4.481e-05      0.000     -0.327      0.743
-0.000       0.000
P_9                                                              0.0001      0.000      0.851      0.395
-0.000       0.000
P_10                                                            -0.0037      0.003     -1.215      0.225
-0.010       0.002
P_11                                                             0.0013      0.004      0.330      0.741
-0.006       0.009
P_12                                                            -0.0008      0.001     -1.337      0.181
-0.002       0.000
P_13                                                             0.0009      0.001      1.462      0.144
-0.000       0.002
P_14                                                             0.0008      0.001      1.297      0.195
-0.000       0.002
P_15                                                             0.0008      0.001      1.347      0.178
-0.000       0.002
P_16                                                             0.0008      0.001      1.341      0.180
-0.000       0.002
P_17                                                             0.0007      0.001      0.762      0.446
-0.001       0.003
P_18                                                          5.353e-05      0.001      0.083      0.934
-0.001       0.001
P_19                                                          5.464e-05      0.001      0.085      0.933
-0.001       0.001
P_20                                                          9.531e-05      0.001      0.148      0.883
-0.001       0.001
P_21                                                             0.0002      0.000      0.319      0.750
-0.001       0.001
P_22                                                            -0.0001      0.000     -0.280      0.780
-0.001       0.001
P_23                                                            -0.0002      0.000     -0.327      0.744
-0.001       0.001
P_24                                                         -1.314e-06   2.53e-07     -5.201      0.000     -1.
81e-06   -8.18e-07
P_25                                                            -0.0001      0.000     -0.380      0.704
-0.001       0.000
P_26                                                             0.1801      0.045      4.044      0.000
0.093       0.268
P_27                                                            -0.0364      0.044     -0.832      0.405
-0.122       0.049
P_28                                                             0.0348      0.032      1.079      0.281
-0.028       0.098
P_29                                                          8.458e-07   2.62e-07      3.222      0.001      3.
31e-07    1.36e-06
P_30                                                         -1.317e-06      5e-07     -2.635      0.009     -2
.3e-06   -3.36e-07
P_31                                                            -0.0344      0.030     -1.148      0.251
-0.093       0.024
P_32                                                             0.1738      0.043      4.023      0.000
0.089       0.259
P_33                                                            -0.0192      0.037     -0.522      0.601
-0.091       0.053
P_34                                                            -0.0957      0.174     -0.550      0.583
-0.437       0.246
P_35                                                            -0.2253      0.167     -1.353      0.176
-0.552       0.101
P_36                                                            -0.0967      0.261     -0.370      0.711
-0.609       0.416
P_37                                                             0.1634      0.186      0.879      0.380
-0.202       0.528
P_38                                                             0.2620      0.183      1.429      0.153
-0.098       0.622
P_39                                                             0.2379      0.182      1.310      0.190
-0.118       0.594
P_40                                                             0.3156      0.288      1.096      0.273
-0.249       0.881
P_41                                                            -0.0001      0.001     -0.095      0.925
-0.003       0.002
```

P_42	6.736e-06	9.16e-06	0.735	0.462	-1.12e-05	2.47e-05
P_43	-0.0325	0.016	-2.027	0.043	-0.064	-0.001
P_44	0.0060	0.032	0.189	0.850	-0.056	0.068
P_45	-0.2238	0.038	-5.887	0.000	-0.298	-0.149
P_46	0.0624	0.012	5.164	0.000	0.039	0.086
P_47	0.0509	0.020	2.581	0.010	0.012	0.090
P_48	0.0001	0.000	0.690	0.490	-0.000	0.000
P_49	-0.0166	0.006	-2.580	0.010	-0.029	-0.004
P_50	0.1909	0.171	1.113	0.266	-0.146	0.527
P_51	-8.132e-06	6.28e-06	-1.295	0.196	-2.05e-05	4.19e-06
P_52	-1.292e-05	3.04e-06	-4.250	0.000	-1.89e-05	-6.95e-06
P_53	-0.0527	0.065	-0.809	0.419	-0.180	0.075
P_54	5.699e-05	1.74e-05	3.282	0.001	2.29e-05	9.11e-05
P_55	-0.0064	0.005	-1.298	0.195	-0.016	0.003
P_56	-0.0160	0.004	-4.221	0.000	-0.023	-0.009
P_57	0.0028	0.001	3.300	0.001	0.001	0.004
P_58	-0.0010	0.001	-1.310	0.191	-0.002	0.000
P_59	3.898e-05	3.3e-05	1.181	0.238	-2.58e-05	0.000
P_60	8.076e-06	6.52e-06	1.239	0.215	-4.71e-06	2.09e-05
P_61	8.043e-06	3.82e-05	0.211	0.833	-6.69e-05	8.3e-05
P_62	-6.577e-08	7.35e-07	-0.089	0.929	-1.51e-06	1.38e-06
P_63	0.0042	0.001	4.093	0.000	0.002	0.006
P_64	0.0059	0.002	2.671	0.008	0.002	0.010
P_65	-0.0087	0.004	-2.073	0.038	-0.017	-0.000
P_66	-6.06e-06	1.77e-05	-0.343	0.732	-4.08e-05	2.86e-05
P_67	-8.793e-05	0.000	-0.775	0.439	-0.000	0.000
P_68	0.0019	0.001	1.484	0.138	-0.001	0.005
P_69	-0.0002	0.000	-1.102	0.271	-0.000	0.000
P_70	-4.847e-09	3.14e-07	-0.015	0.988	-6.21e-07	6.11e-07
P_71	-0.0004	8.81e-05	-4.662	0.000	-0.001	-0.000
P_72	4.343e-06	6.06e-06	0.717	0.474	-7.55e-06	1.62e-05
P_73	3.598e-06	4.11e-06	0.876	0.381	-4.46e-06	1.17e-05
P_74	-0.0001	0.001	-0.256	0.798	-0.001	0.001
P_75	0.0009	0.000	2.283	0.023	0.000	0.002
P_76	0.0008	0.000	2.821	0.005	0.000	0.001
P_77	-9.16e-05	4.7e-05	-1.951	0.051	-0.000	5.57e-07
P_78	-0.0001	0.000	-0.386	0.699	-0.001	0.000
P_79	3.272e-05	0.000	0.124	0.901	-0.000	0.001
P_80	-6.149e-05	0.000	-0.584	0.560	-0.000	0.000
P_81	0.0048	0.002	2.218	0.027	0.001	0.009
P_82	0.0767	0.070	1.097	0.273	-0.061	0.214
P_83	0.0769	0.070	1.100	0.272	-0.060	0.214
P_84	0.0755	0.070	1.079	0.281	-0.062	0.213
P_85	0.0780	0.070	1.115	0.265	-0.059	0.215
P_86	0.0796	0.070	1.138	0.255		

```
-0.058       0.217
P_87                                                              0.0770      0.070      1.100      0.271
-0.060       0.214
P_88                                                              0.0728      0.070      1.040      0.298
-0.065       0.210
P_89                                                              0.0699      0.070      0.998      0.319
-0.068       0.208
P_90                                                              0.0809      0.070      1.154      0.249
-0.057       0.218
P_91                                                              0.0788      0.070      1.125      0.261
-0.059       0.216
P_92                                                              0.0768      0.070      1.098      0.273
-0.060       0.214
P_93                                                              0.0767      0.070      1.096      0.273
-0.061       0.214
P_94                                                             -0.0942      0.084     -1.119      0.264
-0.259       0.071
P_95                                                             -0.0942      0.084     -1.119      0.263
-0.259       0.071
P_96                                                             -0.0937      0.084     -1.113      0.266
-0.259       0.072
P_97                                                             -0.0942      0.084     -1.119      0.263
-0.259       0.071
P_98                                                             -0.0959      0.084     -1.138      0.255
-0.261       0.069
P_99                                                             -0.0939      0.084     -1.116      0.265
-0.259       0.071
P_100                                                            -0.0911      0.084     -1.080      0.281
-0.257       0.075
P_101                                                            -0.0906      0.084     -1.077      0.282
-0.256       0.075
P_102                                                            -0.1004      0.084     -1.193      0.233
-0.265       0.065
P_103                                                            -0.0954      0.084     -1.134      0.257
-0.261       0.070
P_104                                                            -0.0944      0.084     -1.121      0.263
-0.260       0.071
P_105                                                            -0.0940      0.084     -1.117      0.264
-0.259       0.071
P_106                                                            -0.0010      0.001     -1.228      0.220
-0.003       0.001
P_107                                                            -0.0011      0.001     -1.158      0.247
-0.003       0.001
P_108                                                             0.0007      0.001      0.871      0.384
-0.001       0.002
P_109                                                          8.256e-05      0.000      0.332      0.740
-0.000       0.001
P_110                                                             0.0032      0.003      1.192      0.234
-0.002       0.009
P_111                                                         -5.984e-05      0.000     -0.134      0.893
-0.001       0.001
P_112                                                             0.0005      0.000      1.612      0.107
-0.000       0.001
P_113                                                            -0.0002   7.77e-05     -2.396      0.017
-0.000   -3.37e-05
P_114                                                             0.0002      0.000      0.422      0.673
-0.001       0.001
P_115                                                            -0.0003      0.000     -1.324      0.186
-0.001       0.000
P_116                                                            -0.0016      0.001     -1.786      0.074
-0.003       0.000
P_117                                                            -0.0004      0.000     -3.033      0.002
-0.001      -0.000
P_118                                                             0.0012      0.000      3.199      0.001
0.000       0.002
P_119                                                            -0.0003      0.000     -1.165      0.244
-0.001       0.000
P_120                                                             0.0004      0.000      1.648      0.100     -7
.9e-05       0.001
P_121                                                            -0.0010      0.001     -1.091      0.276
-0.003       0.001
==============================================================================
Omnibus:                       72.795   Durbin-Watson:                   1.787
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              275.904
Skew:                           0.165   Prob(JB):                     1.22e-60
Kurtosis:                       5.402   Cond. No.                     1.77e+09
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.77e+09. This might indicate that there are
strong multicollinearity or other numerical problems.
R^2: 0.8783195565982084
Adj. R^2: 0.8621445127623206
MSE: 0.00033146919040689607
MAE: 0.01381283427335565
RSE: 0.018206295350973963
```

In [88]:
```
### Get a sense of the underlying structure of the full set and preprocess the data
```

```
## Inspect multicollinearity using VIF method
# Given that all the VIFs listed are well above 10, it's likely that there are significant multicollinearity is
# In practical terms, this means: High multicollinearity renders variable coefficients unreliable, inflates sta

X_const = sm.add_constant(cb_selected_cleaned)

vif_data = pd.DataFrame()
vif_data["feature"] = X_const.columns
vif_data["VIF"] = [variance_inflation_factor(X_const.values, i) for i in range(len(X_const.columns))]

vif_data_sorted = vif_data.sort_values(by="VIF", ascending=False)
vif_data_sorted.head(50)
```

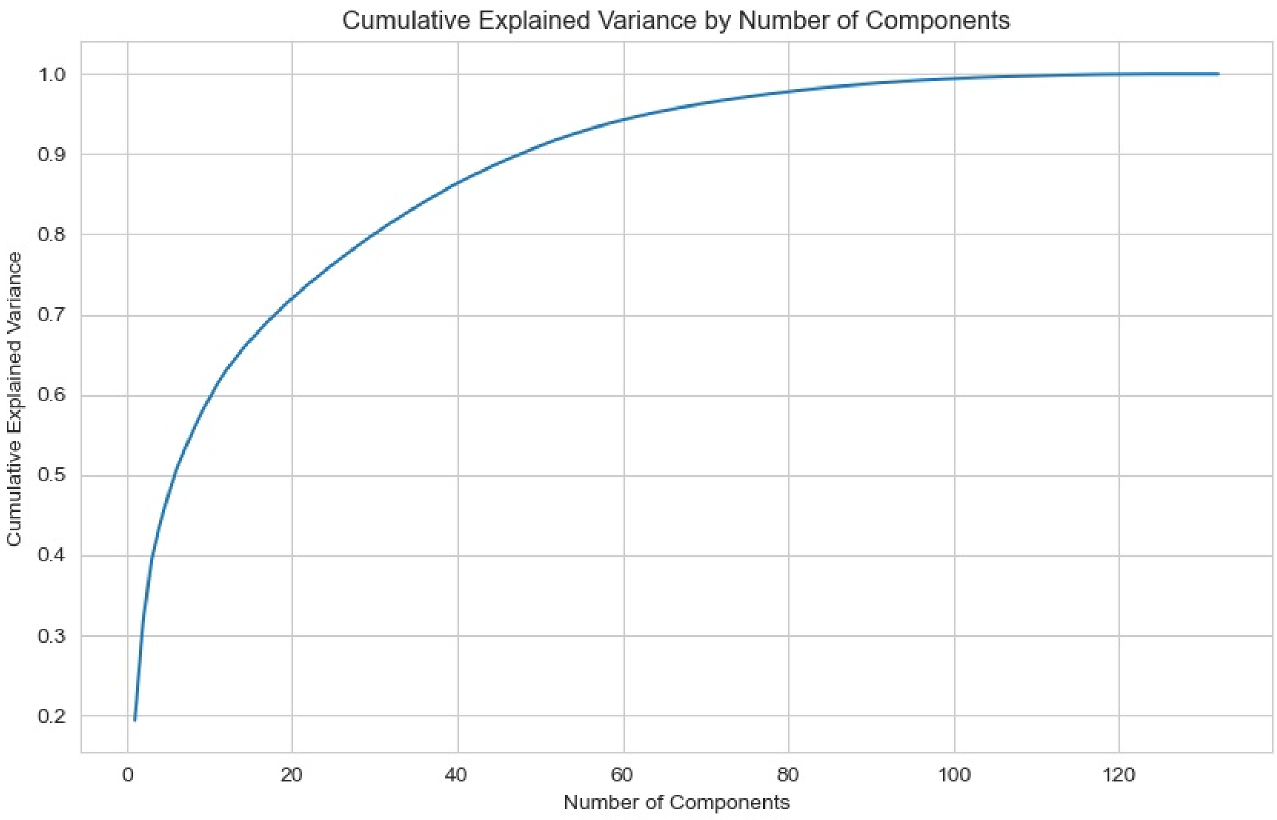
Out[88]:

	feature	VIF
0	const	3.588181e+08
105	P_94	9.753392e+06
108	P_97	6.614680e+06
106	P_95	3.619658e+06
28	P_17	3.361257e+06
93	P_82	3.068480e+06
110	P_99	1.899065e+06
94	P_83	1.730061e+06
23	P_12	1.606438e+06
98	P_87	9.819767e+05
30	P_19	8.011462e+05
116	P_105	7.204926e+05
104	P_93	4.624857e+05
96	P_85	4.606027e+05
107	P_96	2.630441e+05
95	P_84	2.075952e+05
115	P_104	1.624425e+05
29	P_18	1.177445e+05
32	P_21	6.744612e+04
34	P_23	4.828860e+04
103	P_92	3.358905e+04
97	P_86	2.238120e+04
27	P_16	2.044973e+04
109	P_98	1.539387e+04
31	P_20	1.394343e+04
102	P_91	8.262574e+03
114	P_103	5.368239e+03
61	P_50	4.426133e+03
33	P_22	4.107499e+03
112	P_101	3.503127e+03
49	P_38	3.393479e+03
113	P_102	3.125961e+03
26	P_15	2.829284e+03
46	P_35	2.756660e+03
101	P_90	2.221347e+03
48	P_37	2.125381e+03
45	P_34	1.962219e+03
76	P_65	1.457926e+03
50	P_39	1.274840e+03
99	P_88	6.628632e+02
25	P_14	6.530971e+02
75	P_64	6.264350e+02
111	P_100	5.094502e+02
65	P_54	2.981915e+02
21	P_10	2.450794e+02
62	P_51	2.367102e+02
22	P_11	2.356188e+02
100	P_89	2.290460e+02
51	P_40	1.014840e+02
47	P_36	1.003961e+02

In [89]:

```
### Get a sense of the underlying structure of the full set and preprocess the data
## Standardize the variables and Apply PCA
# I didn't choose to remove variables (such as by using feature engineering techniques) because the questions d
# Another way to enhance predictions and address multicollinearity is through Principal Component Analysis (PCA
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_no_outliers)

pca = PCA()
X_pca = pca.fit_transform(X_scaled)

cumulative_variance = np.cumsum(pca.explained_variance_ratio_)
n_components_95 = np.where(cumulative_variance >= 0.95)[0][0] + 1

plt.figure(figsize=(10, 6))
plt.plot(range(1, len(cumulative_variance) + 1), cumulative_variance)
plt.title('Cumulative Explained Variance by Number of Components')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.show()

n_components_95, cumulative_variance[:n_components_95]
```



Out[89]:
```
(64,
 array([0.19369802, 0.31584997, 0.39300288, 0.43675133, 0.47321546,
        0.50601042, 0.5307823 , 0.55430168, 0.57618486, 0.59506609,
        0.6137794 , 0.6295169 , 0.64318455, 0.65625154, 0.66808619,
        0.67934678, 0.69034096, 0.70065527, 0.71043552, 0.71998186,
        0.72897275, 0.73789886, 0.74641052, 0.75466876, 0.76273315,
        0.77071734, 0.77847131, 0.78604166, 0.79344271, 0.80079295,
        0.80769734, 0.81444415, 0.82113343, 0.82772414, 0.83414989,
        0.84033957, 0.84633041, 0.85224249, 0.85812424, 0.86374299,
        0.86902001, 0.87411901, 0.87902876, 0.88386192, 0.88863216,
        0.89307658, 0.89748278, 0.90183355, 0.90604818, 0.91018352,
        0.91414675, 0.91800741, 0.92154304, 0.92495503, 0.92828219,
        0.93146134, 0.93439997, 0.93729191, 0.93999341, 0.9426041 ,
        0.94516176, 0.94759972, 0.94992404, 0.95215232]))
```

In [90]:
```
### Experiment with linear regression (PCA model with outliers included)
## Make predictions using the first 64 components that explain at least 95% of the variance
# The model using PCA outperforms the one without PCA, indicating a higher explained variance and lower predict
# However, for practical interpretation and considering PCA's ineffectiveness for some medthods like tree-based
# I will conduct further analysis of the full set, excluding outliers and not using PCA

X_pca_reduced = X_pca[:, :n_components_95]
y3 = df_no_outliers['kfr_pooled_p25'].values
X_pca_reduced = sm.add_constant(X_pca_reduced)
lin_reg_pca = sm.OLS(y3, X_pca_reduced).fit()

print(lin_reg_pca.summary())

y3_pred = lin_reg_pca.predict(X_pca_reduced)

r2_pca = r2_score(y3, y3_pred)
adj_r2_pca = 1 - (1-r2_pca)*(len(y3)-1)/(len(y3)-X_pca_reduced.shape[1]-1)
mse_pca = mean_squared_error(y3, y3_pred)
mae_pca = mean_absolute_error(y3, y3_pred)
rse_pca = np.sqrt(mse_pca)
```

```python
print(f'R^2: {r2_pca}')
print(f'Adj. R^2: {adj_r2_pca}')
print(f'MSE: {mse_pca}')
print(f'MAE: {mae_pca}')
print(f'RSE: {rse_pca}')
```

```
                            OLS Regression Results                            
==============================================================================
Dep. Variable:                      y   R-squared:                       0.907
Model:                            OLS   Adj. R-squared:                  0.901
Method:                 Least Squares   F-statistic:                     145.6
Date:                Tue, 05 Dec 2023   Prob (F-statistic):               0.00
Time:                        11:13:10   Log-Likelihood:                 2893.0
No. Observations:                1021   AIC:                            -5656.
Df Residuals:                     956   BIC:                            -5336.
Df Model:                          64                                         
Covariance Type:            nonrobust                                         
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4163      0.000    904.433      0.000       0.415       0.417
x1            -0.0057    9.1e-05    -62.581      0.000      -0.006      -0.006
x2            -0.0047      0.000    -40.700      0.000      -0.005      -0.004
x3             0.0041      0.000     28.623      0.000       0.004       0.004
x4             0.0009      0.000      4.634      0.000       0.001       0.001
x5            -0.0035      0.000    -16.882      0.000      -0.004      -0.003
x6             0.0011      0.000      5.014      0.000       0.001       0.002
x7            -0.0048      0.000    -18.905      0.000      -0.005      -0.004
x8             0.0021      0.000      8.030      0.000       0.002       0.003
x9             0.0010      0.000      3.697      0.000       0.000       0.002
x10            0.0066      0.000     22.619      0.000       0.006       0.007
x11           -0.0014      0.000     -4.669      0.000      -0.002      -0.001
x12           -0.0001      0.000     -0.333      0.739      -0.001       0.001
x13           -0.0050      0.000    -14.694      0.000      -0.006      -0.004
x14            0.0024      0.000      6.732      0.000       0.002       0.003
x15           -0.0011      0.000     -3.058      0.002      -0.002      -0.000
x16            0.0028      0.000      7.513      0.000       0.002       0.004
x17           -0.0024      0.000     -6.344      0.000      -0.003      -0.002
x18           -0.0049      0.000    -12.338      0.000      -0.006      -0.004
x19           -0.0007      0.000     -1.844      0.066      -0.002    4.81e-05
x20           -0.0018      0.000     -4.297      0.000      -0.003      -0.001
x21           -0.0003      0.000     -0.676      0.499      -0.001       0.001
x22           -0.0009      0.000     -2.121      0.034      -0.002   -6.71e-05
x23            0.0004      0.000      0.903      0.367      -0.000       0.001
x24           -0.0009      0.000     -1.932      0.054      -0.002    1.36e-05
x25            0.0015      0.000      3.390      0.001       0.001       0.002
x26            0.0024      0.000      5.415      0.000       0.002       0.003
x27            0.0007      0.000      1.518      0.129      -0.000       0.002
x28            0.0005      0.000      1.044      0.297      -0.000       0.001
x29           -0.0052      0.000    -11.082      0.000      -0.006      -0.004
x30           -0.0036      0.000     -7.619      0.000      -0.004      -0.003
x31            0.0018      0.000      3.807      0.000       0.001       0.003
x32            0.0011      0.000      2.279      0.023       0.000       0.002
x33           -0.0018      0.000     -3.634      0.000      -0.003      -0.001
x34           -0.0027      0.000     -5.521      0.000      -0.004      -0.002
x35           -0.0046      0.000     -9.267      0.000      -0.006      -0.004
x36            0.0051      0.001      9.921      0.000       0.004       0.006
x37           -0.0041      0.001     -7.906      0.000      -0.005      -0.003
x38            0.0039      0.001      7.413      0.000       0.003       0.005
x39            0.0031      0.001      5.945      0.000       0.002       0.004
x40           -0.0016      0.001     -3.067      0.002      -0.003      -0.001
x41            0.0024      0.001      4.289      0.000       0.001       0.003
x42            0.0038      0.001      6.855      0.000       0.003       0.005
x43            0.0003      0.001      0.562      0.574      -0.001       0.001
x44           -0.0029      0.001     -5.047      0.000      -0.004      -0.002
x45            0.0012      0.001      2.098      0.036    7.86e-05       0.002
x46           -0.0058      0.001     -9.712      0.000      -0.007      -0.005
x47           -0.0015      0.001     -2.464      0.014      -0.003      -0.000
x48           -0.0024      0.001     -3.953      0.000      -0.004      -0.001
x49           -0.0047      0.001     -7.609      0.000      -0.006      -0.003
x50            0.0007      0.001      1.181      0.238      -0.000       0.002
x51            0.0026      0.001      4.029      0.000       0.001       0.004
x52            0.0021      0.001      3.306      0.001       0.001       0.003
x53            0.0006      0.001      0.858      0.391      -0.001       0.002
x54           -0.0018      0.001     -2.622      0.009      -0.003      -0.000
x55           -0.0022      0.001     -3.176      0.002      -0.004      -0.001
x56            0.0015      0.001      2.069      0.039    7.55e-05       0.003
x57            0.0003      0.001      0.397      0.691      -0.001       0.002
x58           -0.0048      0.001     -6.502      0.000      -0.006      -0.003
x59           -0.0011      0.001     -1.400      0.162      -0.003       0.000
x60            0.0022      0.001      2.742      0.006       0.001       0.004
x61           -0.0008      0.001     -0.999      0.318      -0.002       0.001
x62           -0.0058      0.001     -7.190      0.000      -0.007      -0.004
x63           -0.0026      0.001     -3.113      0.002      -0.004      -0.001
x64            0.0021      0.001      2.484      0.013       0.000       0.004
==============================================================================
Omnibus:                        9.887   Durbin-Watson:                   1.886
Prob(Omnibus):                  0.007   Jarque-Bera (JB):               14.594
Skew:                           0.019   Prob(JB):                     0.000678
```

```
Kurtosis:                       3.585   Cond. No.                         9.32
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
R^2: 0.906959439161388
Adj. R^2: 0.9006268355441003
MSE: 0.00020249755966564623
MAE: 0.010995583954843955
RSE: 0.014230163725890374
```

9. Implement a decision tree on the full predictor set using 10 fold cross-validation to select the optimal tree size. What is the first split? Discuss why the first split is often an important predictor or correlate of the outcome.

The optimal tree size for the decision tree is 5.

The first split in the optimal decision tree is made based on the predictor `P_57` (Percent of Adults That Report Fair or Poor Health (Persons 18 Years and Over) . Specifically, the split occurs at a threshold of approximately 14.75.

The first split of a decision tree is often considered important for several reasons: The first split in a decision tree is the one that reduces outcome variability the most, indicating a strong association with the target variable. It affects the largest subset of data and sets the stage for subsequent splits, reflecting its importance in the predictive model. The top split often offers clear and immediate insight into the primary factor that influences the outcome variable.

In [93]:
```python
### Implement a decision tree on the full set
## use 10 fold cross-validation to select the optimal tree size

kf = KFold(n_splits=10, shuffle=True, random_state=42)

avg_rmse = []

depths = list(range(1, 31))

for depth in depths:
    tree = DecisionTreeRegressor(max_depth=depth, random_state=42)
    mse_scores = -cross_val_score(tree, X, y, cv=kf, scoring='neg_mean_squared_error')
    rmse_scores = np.sqrt(mse_scores)
    avg_rmse.append(rmse_scores.mean())

optimal_depth = depths[np.argmin(avg_rmse)]
min_rmse = min(avg_rmse)

best_tree = DecisionTreeRegressor(max_depth=optimal_depth, random_state=42)
best_tree.fit(X, y)

y_pred_best_tree = best_tree.predict(X)

mse_best_tree = mean_squared_error(y, y_pred_best_tree)
r2_best_tree = r2_score(y, y_pred_best_tree)

print(f'Optimal tree depth: {optimal_depth}')
print(f'Minimum average RMSE: {min_rmse}')
print(f'MSE for optimal Decision Tree: {mse_best_tree}')
print(f'R2 for optimal Decision Tree: {r2_best_tree}')
```

```
Optimal tree depth: 5
Minimum average RMSE: 0.029281416252176277
MSE for optimal Decision Tree: 0.00046974729973050627
R2 for optimal Decision Tree: 0.7841675114924137
```

In [94]:
```python
### Implement a decision tree on the full set
## Find the index and threshold used for the first split

optimal_tree = DecisionTreeRegressor(max_depth=optimal_depth, random_state=42)
optimal_tree.fit(X, y)

feature_index = optimal_tree.tree_.feature[0]
feature_name = X.columns[feature_index]
threshold = optimal_tree.tree_.threshold[0]

feature_name, threshold
```

Out[94]:
```
('P_57', 14.75)
```

10. You could have created a larger tree that would have had lower prediction error in this training data. Why do we use cross-validation to select a smaller tree instead of just using as many splits as possible?

Using cross-validation to choose a smaller decision tree, instead of the largest possible one, helps us build a model that performs well not just on our current data but also on new, unseen data. Firstly, a very large tree can fit the training data too closely, capturing noise as if it were a real pattern. This can lead to mistakes when predicting new data; secondly, cross-validation tests the tree on different subsets of the data to ensure it works well in general, not just on the data it was trained on; and thirdly, smaller trees are easier to understand and manage, and often they're all you need to make good predictions. Therefore, cross-validation helps find a good middle ground — a model that's simple yet effective at making predictions.

11. Implement a random forest with at least 1000 bootstrap samples and obtain predictions.

Based on the output metrics: It seems the Random Forest regressor has been trained effectively and is providing highly accurate predictions for the full dataset. If this performance is consistent across different test sets and in a real-world scenario, it suggests that the model is well-fitted and could be a reliable tool for making predictions based on the features provided.

First five predictions: [0.36373053 0.41150519 0.39935276 0.36919808 0.35601651]

In [95]:
```
### Implement a random forest on the full set
# Use 1000 bootstrap samples and obtain predictions using the trained random forest

rf_regressor = RandomForestRegressor(n_estimators=1000, random_state=42, n_jobs=-1)
rf_regressor.fit(X, y)
rf_predictions = rf_regressor.predict(X)
mse_rf = mean_squared_error(y, rf_predictions)
r2_rf = r2_score(y, rf_predictions)

print(f'MSE for Random Forest: {mse_rf}')
print(f'R2 for Random Forest: {r2_rf}')
print(f'First five predictions: {rf_predictions[:5]}')
```

```
MSE for Random Forest: 5.343772195222946e-05
R2 for Random Forest: 0.9754472319143868
First five predictions: [0.36373053 0.41150519 0.39935276 0.36919808 0.35601651]
```

12. Calculate and compare the mean squared error for your results on 8, 9, 11 in -sample.

Linear Regression: 0.00026463525924545615, Decision Tree: 0.00046974729973050627, Random Forest: 5.343772195222946e-05

In sample, the Random Forest has the lowest MSE, which suggests that it has the best performance among the three in terms of error minimization. The Decision Tree has the highest MSE, indicating the poorest fit among the three. The Linear Regression's performance is in the middle of the other two.

In [96]:
```
### Compare MSE for all the 3 models' results in-sample

linear_reg_predictions = lin_reg.predict(X)
optimal_tree = DecisionTreeRegressor(max_depth=5)
optimal_tree.fit(X, y)
decision_tree_predictions = optimal_tree.predict(X)

mse_linear_reg = ((y - linear_reg_predictions) ** 2).mean()
mse_decision_tree = ((y - decision_tree_predictions) ** 2).mean()
mse_random_forest = ((y - rf_predictions) ** 2).mean()

mse_values = {
    'Linear Regression': mse_linear_reg,
    'Decision Tree': mse_decision_tree,
    'Random Forest': mse_random_forest
}

mse_values
```

Out[96]:
```
{'Linear Regression': 0.0002646352592454583,
 'Decision Tree': 0.00046974729973050627,
 'Random Forest': 5.343772195222946e-05}
```

13. Briefly comment on whether or not you think your regression from question 8, question 9 or from question 11 will predict krf_pooled_p25 better out-of-sample.

Here's a brief evaluation:

- OLS Regression Model: It has an R-squared of 0.878 and a potential issue with multicollinearity, given the large condition number. High multicollinearity can affect the stability of the coefficient estimates, which may lead to poorer out-of-sample performance. If the model is overfit to the in-sample data, it may not generalize well to new data.
- Decision Tree Model: The optimal tree depth was found to be 5, which suggests the model is not overly complex. It has an R-squared of 0.784, which is lower than the OLS model, indicating it may not capture as much of the variance in the training data. However, because it's less complex, it might generalize better and could be more robust to out-of-sample data than a highly parameterized OLS model.
- Random Forest Model: This model showed an R-squared of 0.975, which is very high, and a very low MSE. While Random Forest models are less likely to overfit compared to individual decision trees due to their ensemble nature, there's still a risk of overfitting if the model is too complex or if the hyperparameters are not tuned appropriately.

Part 3: Out-of-sample validation

14. Now turn to the test data set. Calculate the mean squared error for your results from 8, 9, and 11 out-of-sample.

Linear Regression: 0.0002765328298568708, Decision Tree: 0.0007590081564404664, Random Forest: 0.00029013203028211256

In [120…

```
### Prepare the data for the test set

# Load the test dataset
test_data = pd.read_stata("atlas_test.dta")
```

In [121…

```
cb_selected
```

Out[121]:

	kfr_pooled_p25	Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	Value:Median_Age_Person	Value:Median_Income_Per
0	0.354766	0.254326	39.6	2445
1	0.413865	0.227504	57.3	2299
2	0.394591	0.218621	55.4	2096
3	0.356809	0.114001	40.1	1974
4	0.349491	0.160414	36.0	2672
...	...	...	...	
2238	NaN	0.120096	39.1	1022
2239	NaN	0.060818	40.2	1086
2240	NaN	0.118058	36.5	1089
2241	NaN	0.067963	37.6	1414
2242	NaN	0.151401	39.7	1047

2243 rows × 133 columns

In [122…

```
### Prepare the data for the test set
## Select rows where 'kfr_pooled_p25' is missing

# Extract the 'identifier' columns from the 'cb' DataFrame
id_columns = [col for col in cb.columns if "identifier" in col]

# Merge on the 'identifier' column, ensure it exists in both DataFrames
if 'identifier' in cb.columns and 'identifier' in cb_selected.columns:
    cb_selected = cb_selected.merge(cb[id_columns], on='identifier')
else:
    print("'identifier' column not found in one of the DataFrames")

# Filter rows where 'kfr_pooled_p25' is NaN
cb_selected_cleaned_reversed = cb_selected[pd.isna(cb_selected['kfr_pooled_p25'])]

# Final DataFrame with selected and cleaned rows
cb_selected_cleaned_reversed
```

Out[122]:

	kfr_pooled_p25	Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	Value:Median_Age_Person	Value:Median_Income_Per
1126	NaN	0.222662	37.0	2698
1127	NaN	0.192593	37.5	3233
1128	NaN	0.205765	47.0	2689
1129	NaN	0.108767	33.7	4088
1130	NaN	0.119638	30.8	2341
...	...	...	...	
2238	NaN	0.120096	39.1	1022
2239	NaN	0.060818	40.2	1086
2240	NaN	0.118058	36.5	1089
2241	NaN	0.067963	37.6	1414
2242	NaN	0.151401	39.7	1047

1117 rows × 133 columns

In [123…

```
### Prepare the data for the test set
# Merge datasets

test_data = test_data.rename(columns={"geoid": "identifier"})
test_data['identifier'] = test_data['identifier'].astype(str).str.replace(".0", "")
test_data_merged = cb_selected_cleaned_reversed.merge(test_data, on=["identifier"], how="inner")
test_data_merged_dropped = test_data_merged.drop(['kfr_pooled_p25', 'identifier'], axis=1)
test_data_merged_dropped
```

Out[123]:

	Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome	Value:Median_Age_Person	Value:Median_Income_Person	Value:Stand
0	0.222662	37.0	26984.0	
1	0.192593	37.5	32339.0	
2	0.205765	47.0	26895.0	
3	0.108767	33.7	40884.0	
4	0.119638	30.8	23418.0	
...	...	...	...	
1112	0.120096	39.1	10227.0	
1113	0.060818	40.2	10865.0	
1114	0.118058	36.5	10893.0	
1115	0.067963	37.6	14146.0	
1116	0.151401	39.7	10479.0	

1117 rows × 132 columns

In [124…

```
### Prepare the data for the test set
## Inspect the missing values

missing_values_count = test_data_merged_dropped.isna().sum()
columns_with_missing_values = missing_values_count[missing_values_count > 0]
total_rows = len(test_data_merged_dropped)
missing_percentage = (columns_with_missing_values / total_rows) * 100

missing_data_df = pd.DataFrame({
    'Missing Values': columns_with_missing_values,
    'Percentage': missing_percentage
})

missing_data_df
```

Out[124]:

	Missing Values	Percentage
Value:StandardizedPrecipitationIndex_Atmosphere	36	3.222919

In [125…

```
### Prepare the data for the test set
## Handle the missing values

test_data_filled = test_data_merged_dropped.fillna(test_data_merged_dropped.median())
test_missing_values_final = test_data_filled.isna().sum()
test_missing_values_final = test_missing_values_final[test_missing_values_final > 0]
test_missing_values_final
```

Out[125]:

```
Series([], dtype: int64)
```

In [126…

```
### Prepare the data for the test set
## Identify potential outliers using Z-score method
df_zscore_train = test_data_filled.apply(zscore)
outliers = (df_zscore_train.abs() > 3).sum()
outliers
```

Out[126]:

```
Value:GenderIncomeInequality_Person_15OrMoreYears_WithIncome    11
Value:Median_Age_Person                                         12
Value:Median_Income_Person                                      10
Value:StandardizedPrecipitationIndex_Atmosphere                  7
Value:Rate_Person_BachelorOfArtsHumanitiesAndOtherMajor         19
                                                                ..
P_118                                                           30
P_119                                                           22
P_120                                                           21
P_121                                                           15
kfr_actual                                                       9
Length: 132, dtype: int64
```

In [127…

```
### Prepare the data for the test set
## Handle the outliers and prepare the datasets for testing

df_no_outliers_test = test_data_filled[(df_zscore_train.abs() <= 3).all(axis=1)]
X_train = X
X_test = df_no_outliers_test.drop(columns=['kfr_actual'])
X_test.insert(0, 'const', 1)
y_test = df_no_outliers_test['kfr_actual'].values
test_data_filled_renamed = df_no_outliers_test.rename(columns={"kfr_actual": "kfr_pooled_p25"})
y_test = test_data_filled_renamed['kfr_pooled_p25'].values
```

In [128…

```
### Calculate MSE for all the 3 models' results out-of-sample

linear_reg_test_predictions_aligned = lin_reg.predict(X_test)
decision_tree_test_predictions_aligned = optimal_tree.predict(X_test)
```

```
rf_test_predictions_aligned = rf_regressor.predict(X_test)

mse_linear_reg_test_aligned = ((y_test - linear_reg_test_predictions_aligned) ** 2).mean()
mse_decision_tree_test_aligned = ((y_test - decision_tree_test_predictions_aligned) ** 2).mean()
mse_rf_test_aligned = ((y_test - rf_test_predictions_aligned) ** 2).mean()

mse_out_of_sample_aligned = {
    'Linear Regression': mse_linear_reg_test_aligned,
    'Decision Tree': mse_decision_tree_test_aligned,
    'Random Forest': mse_rf_test_aligned
}

print(mse_out_of_sample_aligned)
```

```
{'Linear Regression': 0.0002765328298568708, 'Decision Tree': 0.0007590081564404664, 'Random Forest': 0.0002901
3203028211256}
```

15. Which model did the best? Write a one page summary of your analysis with a nicely formatted table showing the in-sample and out-of-sample mean squared error for your models estimated in questions 8, 9, and 11.

In the empirical project aimed at forecasting the future economic status of children whose parents are in the lower quarter of national earnings, three models were tested using data from the Opportunity Atlas on counties with over 10,000 residents. The Linear Regression model scored an R-squared of 0.878 but had multicollinearity issues, casting doubt on its stability despite its interpretability and decent out-of-sample Mean Squared Error (MSE) performance. The Decision Tree was less accurate, with a lower in-sample R-squared and a higher out-of-sample MSE, though it offered simplicity and was easy to understand.

The standout was the Random Forest model, scoring an impressive in-sample R-squared of 0.975 and the lowest MSE, indicating strong fit and predictive power. It also maintained a low out-of-sample MSE, suggesting it is reliable for general use despite being complex and less interpretable. The study suggested that while Random Forest provided the best balance between accuracy and complexity, the final model choice would also need to consider the importance of interpretability to stakeholders.

The following table provides a succinct overview of the MSE outcomes for each model:

Model	In-Sample MSE	Out-of-Sample MSE
Linear Regression	0.00026463525924545615	0.0002765328298568708
Decision Tree	0.00046974729973050627	0.0007590081564404664
Random Forest	5.343772195222946e-05	0.00029013203028211256

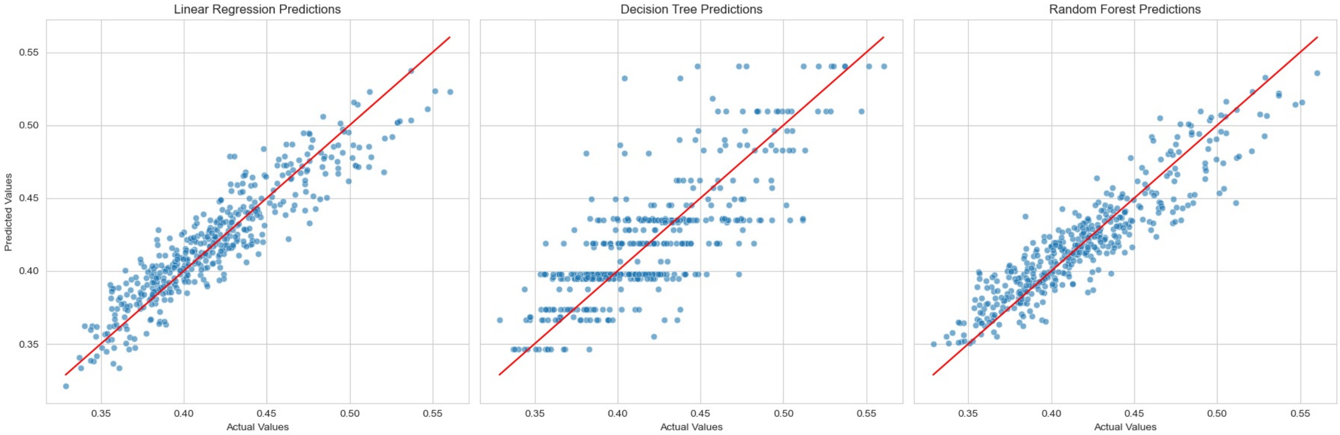
The study reaffirms the significance of employing advanced modeling techniques to create accurate and actionable forecasts in the field of socio-economic research, highlighting their ability to decipher complex patterns, enhance predictive accuracy, and inform effective policy-making. However, it also acknowledges limitations such as potential biases, the exclusion of smaller counties, and the absence of less tangible mobility factors like social capital. Furthermore, the Random Forest model's complexity could limit its generalizability across different datasets or over time, especially in the context of changing socio-economic conditions.

In summary, the research advances our understanding of predicting intergenerational mobility but underscores the need for a clear theoretical framework to select variables, robust data quality, and the ethical use of predictive modeling in socio-economic policy. It highlights the importance of not only technical precision but also the broader implications of such forecasts on policy and equity.

16. Draw some graphs or maps to visualize your predictions.

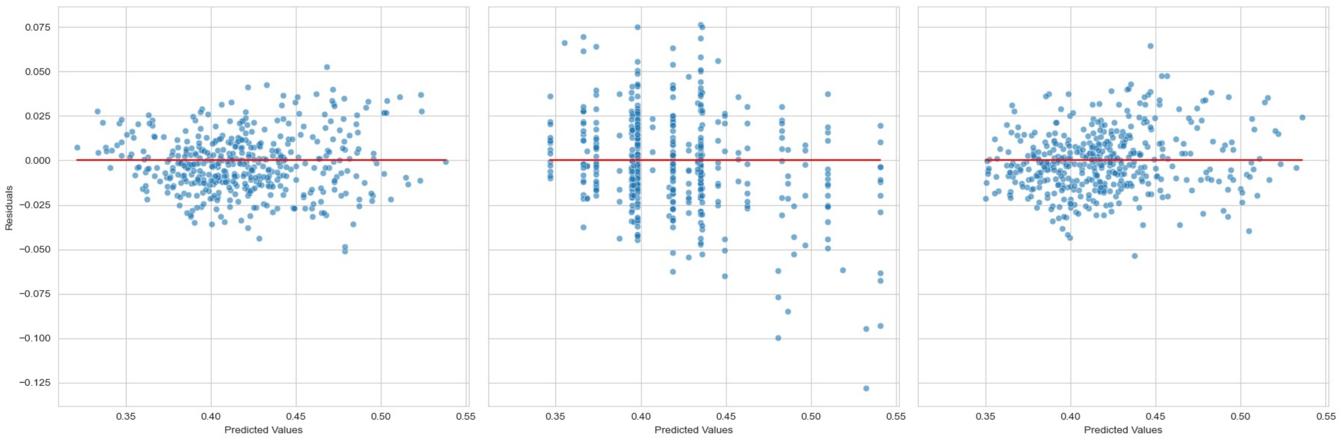
In [129…

```
### Prediction visualization
## Scatter plots of the actual vs. predicted values with a fitted regression line
# Points closer to the regression line indicate better predictions

sns.set_style("whitegrid")

def plot_predictions(ax, y_true, y_pred, title):
    sns.scatterplot(ax=ax, x=y_true, y=y_pred, alpha=0.6)
    sns.lineplot(ax=ax, x=y_true, y=y_true, color='red')
    ax.set_title(title)
    ax.set_xlabel('Actual Values')
    ax.set_ylabel('Predicted Values')

fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True, sharey=True)

plot_predictions(axes[0], y_test, linear_reg_test_predictions_aligned, 'Linear Regression Predictions')
plot_predictions(axes[1], y_test, decision_tree_test_predictions_aligned, 'Decision Tree Predictions')
plot_predictions(axes[2], y_test, rf_test_predictions_aligned, 'Random Forest Predictions')

plt.tight_layout()
plt.show()
```



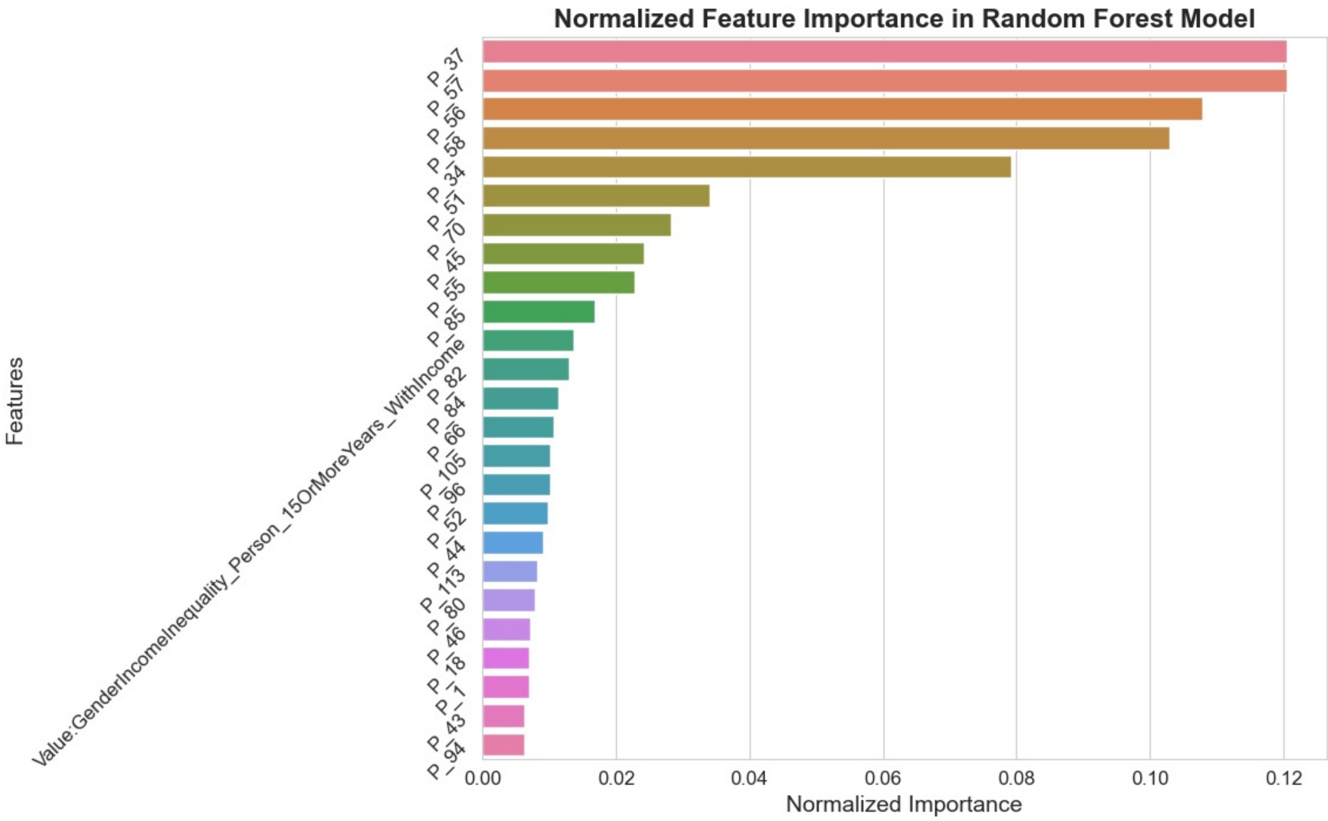
In [130…

```
### Prediction visualization
## Plot the residuals of the differences between the observed and predicted values
# Ideally, the residuals should be randomly dispersed around the horizontal axis, indicating that the model's p

sns.set_style("whitegrid")

def plot_residuals(ax, y_true, y_pred, title):
    residuals = y_true - y_pred
    sns.scatterplot(ax=ax, x=y_pred, y=residuals, alpha=0.6)
    sns.lineplot(ax=ax, x=y_pred, y=[0]*len(y_pred), color='red')
    ax.set_xlabel('Predicted Values')
    ax.set_ylabel('Residuals')

fig, axes = plt.subplots(1, 3, figsize=(18, 6), sharex=True, sharey=True)

plot_residuals(axes[0], y_test, linear_reg_test_predictions_aligned, 'Linear Regression Residuals')
plot_residuals(axes[1], y_test, decision_tree_test_predictions_aligned, 'Decision Tree Residuals')
plot_residuals(axes[2], y_test, rf_test_predictions_aligned, 'Random Forest Residuals')

plt.tight_layout()
plt.show()
```



In [131…

```
### Prediction visualization
## Feature Importance Plots for Random Forest
# Since the Random Forest model performed best, understanding which features most influence the predictions is
# This visualization would shed light on the underlying factors that drive intergenerational mobility

feature_importances = rf_regressor.feature_importances_
feature_importances_normalized = feature_importances / np.sum(feature_importances)
features_series = pd.Series(feature_importances_normalized, index=X_test.columns)
features_series_sorted = features_series.sort_values(ascending=False)
cumulative_importance = np.cumsum(features_series_sorted)
features_to_keep = cumulative_importance[cumulative_importance <= 0.8]
features_series_filtered = features_series_sorted.loc[features_to_keep.index]

plt.figure(figsize=(12, 8))
color_palette = sns.color_palette("husl", len(features_series_filtered))

sns.barplot(x=features_series_filtered, y=features_series_filtered.index, palette=color_palette)

plt.xticks(fontsize=12)
plt.yticks(fontsize=12, rotation=45)  # Rotate y-axis labels for better readability
plt.title('Normalized Feature Importance in Random Forest Model', fontsize=16, fontweight='bold')
plt.xlabel('Normalized Importance', fontsize=14)
plt.ylabel('Features', fontsize=14)
plt.grid(True, axis='x')
plt.subplots_adjust(left=0.3)
plt.show()
```

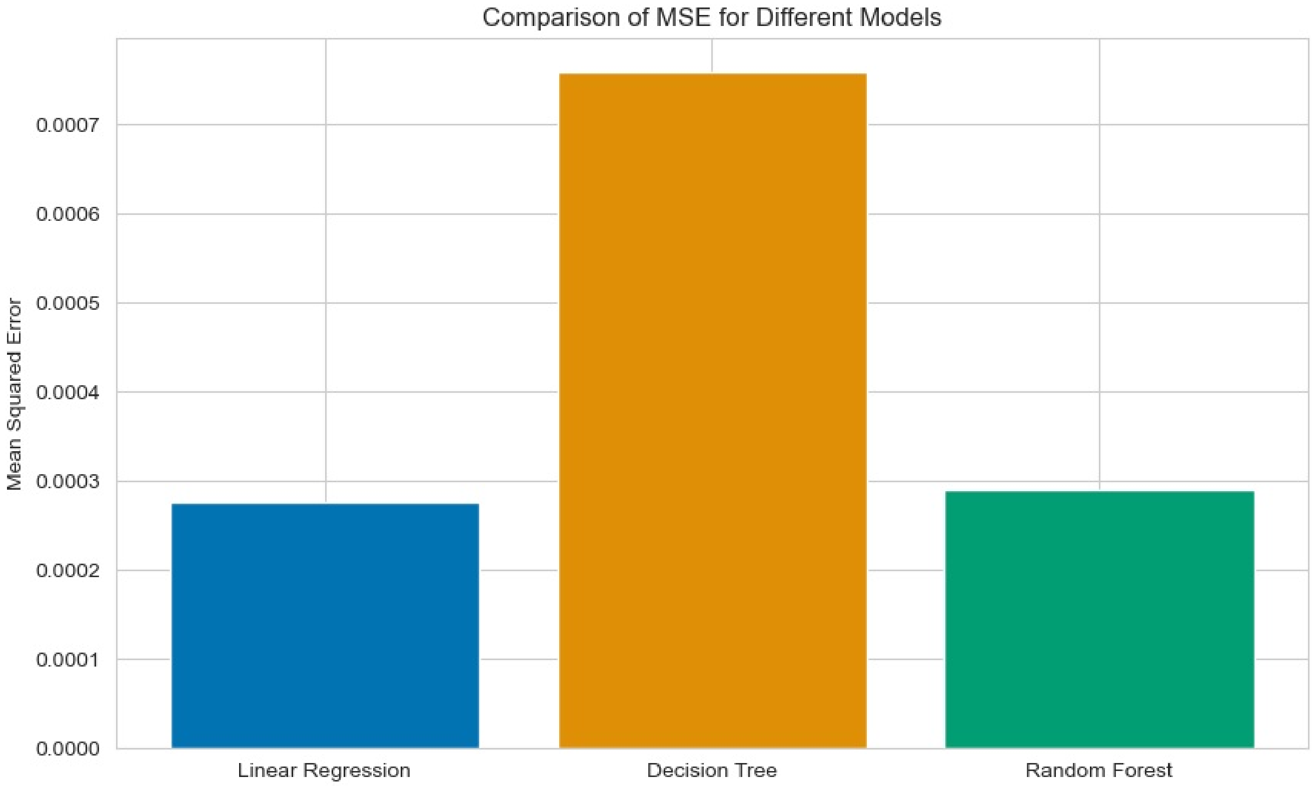


Top 5 Features:

- `P_37` Share black 2000
- `P_57` Percent of Adults That Report Fair or Poor Health (Persons 18 Years and Over)
- `P_56` Mentally Unhealthy Days per Month (Persons 18 Years and Over)
- `P_58` Percent of Low Birthweight Births (<2.5kg)
- `P_34` Share black 2010

In [132…

```
### Prediction visualization
## Bar Charts or Box Plots for Error Metrics
# A direct comparison of the error metrics across the models

mse_values = [mse_linear_reg_test_aligned, mse_decision_tree_test_aligned, mse_rf_test_aligned]
model_names = ['Linear Regression', 'Decision Tree', 'Random Forest']

palette = sns.color_palette("colorblind", len(model_names))

plt.figure(figsize=(10, 6))
plt.bar(model_names, mse_values, color=palette)
plt.title('Comparison of MSE for Different Models')
plt.ylabel('Mean Squared Error')
plt.show()
```


Comparison of MSE for Different Models
Mean Squared Error
0.0007
0.0006
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Linear Regression
Decision Tree
Random Forest