**Assignment 2**

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

import statsmodels.api as sm

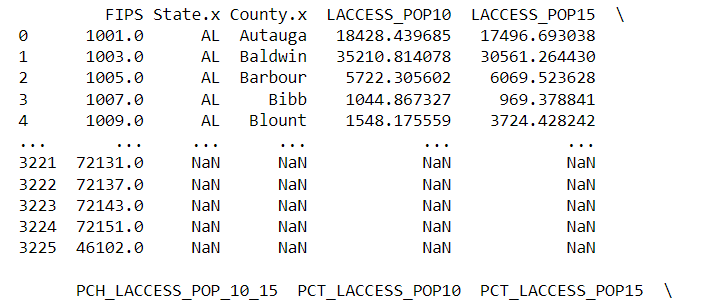
from scipy import stats

**Question 1**

##Q1 importing Local dataset

df = pd.read\_excel(r'C:\Users\shaun.rolph\Desktop\Assignmenttwo\FEA2020.xlsx', sheet\_name='FEA')

print(df)



**## Q1 subset creation**

df = pd.DataFrame(df, columns=['PCT\_LACCESS\_POP15', 'GROCPTH16', 'FFRPTH16', 'METRO13', 'MEDHHINC15', 'RECFACPTH16', 'PCT\_DIABETES\_ADULTS13', 'Pop2020', 'FIPS', 'State.x', 'County.x', 'State.y', 'County.y'])

A screenshot of a computer

Description automatically generateddf.head()

**## Q1A Making DF for just variables to be histogramed**

df\_Histograms = pd.DataFrame(df, columns=['PCT\_LACCESS\_POP15', 'GROCPTH16', 'FFRPTH16', 'METRO13', 'MEDHHINC15', 'RECFACPTH16', 'PCT\_DIABETES\_ADULTS13', 'Pop2020', 'FIPS'])

df.head()

**## Q1A Making DF for just variables to be histogramed**

# Function to create histograms for each column

def create\_histograms(df\_Histograms, log=False):

# Determine the number of columns in the DataFrame

num\_columns = df\_Histograms.shape[1]

# Define the number of rows and columns for subplots

num\_rows = int(np.ceil(num\_columns\*\*0.5))

num\_cols = int(np.ceil(num\_columns / num\_rows))

# Create subplots

fig, axes = plt.subplots(num\_rows, num\_cols, figsize=(15, 10))

fig.subplots\_adjust(wspace=0.3, hspace=0.3)

# Flatten the axes if num\_rows and num\_cols are not the same

axes = axes.flatten()

# Create histograms for each column

for i in range(num\_columns):

ax = axes[i]

column\_data = df.iloc[:, i]

ax.hist(column\_data, bins=20, color='blue', alpha=0.7)

ax.set\_title(df.columns[i])

if log:

ax.set\_yscale('log')

# Hide any extra empty subplots

for i in range(num\_columns, len(axes)):

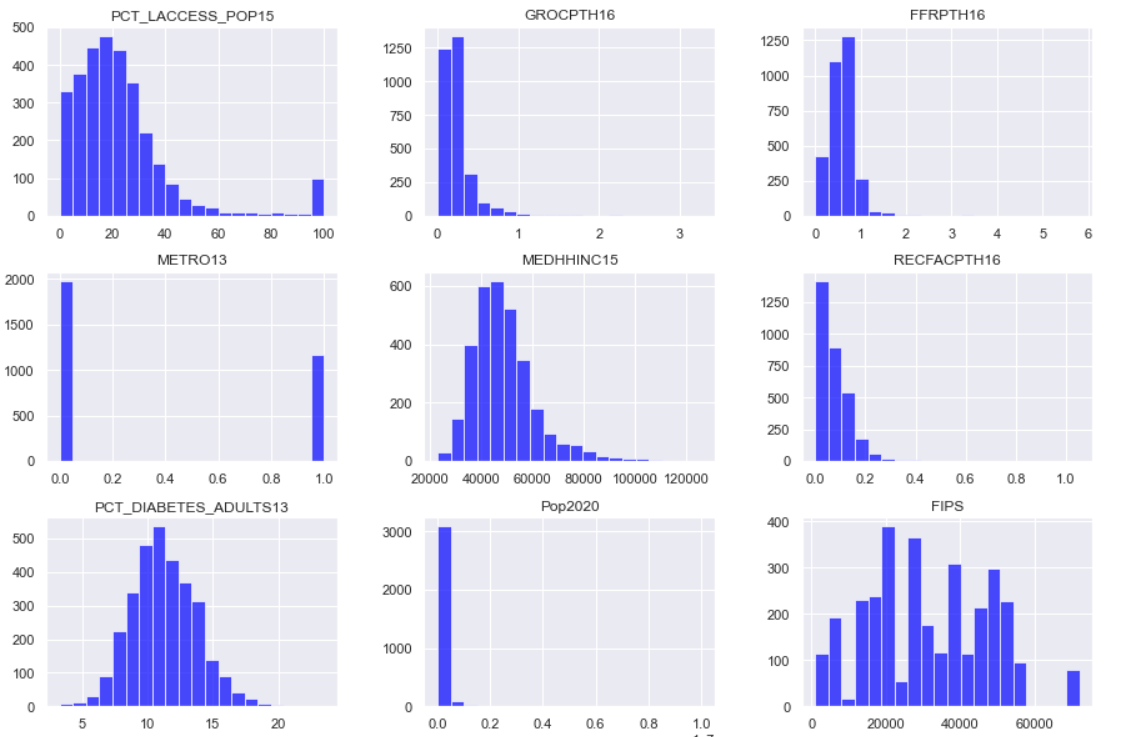
fig.delaxes(axes[i])

plt.show()

**## Q1A**

# create regular histograms

create\_histograms(df\_Histograms)



**## Q1A**

# create log histograms

create\_histograms(df\_Histograms, log=True)



**## Q1A: Do any have skewed distributions?**

PCT\_LACCESS\_POP15 looks heavily skewed towards the left side of the distribution with some clear critical observations on the far-right side of the tail. RECFACPTH16 has a central base within the distribution and then outliers in the higher quartile of the distribution. But PCT\_LACCESS\_POP15 just from looking at the histogram looks to be the most skewed results. When taking the log of PCT\_LACCESS\_POP15 there is still a skew one could always normalize the distribution through z-scores to assist in understanding the skew of distribution better and help with those outliers to on the right side of the distribution.

**## Q1b**

# Calculate the correlation matrix

correlation\_matrix = df\_Histograms.corr()

# Define the color map for positive and negative correlations

cmap = sns.diverging\_palette(220, 10, as\_cmap=True)

# Create a heatmap of the correlation matrix

plt.figure(figsize=(10, 10))

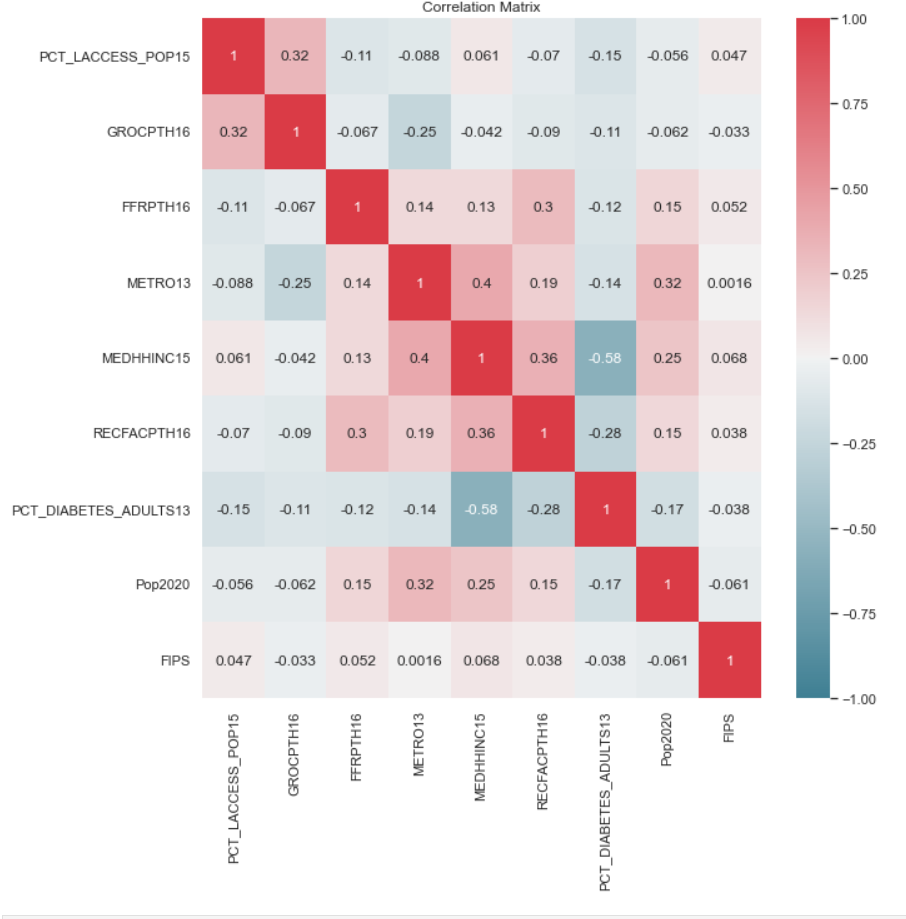
sns.heatmap(correlation\_matrix, annot=True, cmap=cmap, vmin=-1, vmax=1)

# Customize the plot

plt.title('Correlation Matrix')

# Show the plot

plt.show()



**## Q1b**

# Find the strongest correlating variables

max\_corr = 0 # Initialize the maximum correlation value

strongest\_pairs = [] # Initialize a list to store the strongest pairs

for i in range(len(correlation\_matrix.columns)):

for j in range(i+1, len(correlation\_matrix.columns)):

if abs(correlation\_matrix.iloc[i, j]) > max\_corr:

max\_corr = abs(correlation\_matrix.iloc[i, j])

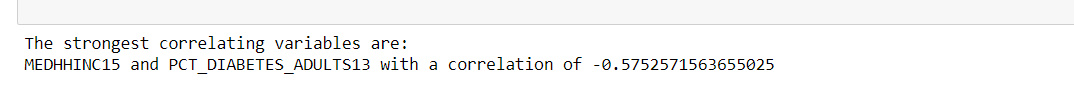
strongest\_pairs = [(correlation\_matrix.columns[i], correlation\_matrix.columns[j])]

elif abs(correlation\_matrix.iloc[i, j]) == max\_corr:

strongest\_pairs.append((correlation\_matrix.columns[i], correlation\_matrix.columns[j]))

print("The strongest correlating variables are:")

for pair in strongest\_pairs:

 print(f"{pair[0]} and {pair[1]} with a correlation of {correlation\_matrix.loc[pair[0], pair[1]]}")

**## Q1b. Interpret results.**

The correlation matrix shows the strength both positive and negative of the relationship between each variables. Measuring if two variables are linearly related helps identify what could be used for a potential linear regression to further experiment with the relationship between variables. Above shows the strong correlation which happens to be negative meaning the relationship has a negative slope, as one variable increase the other decreases potentially.

**## Q2A. create Scatter Plot**

# Specify the names of the columns you want to use for the scatter plot

x\_column = 'MEDHHINC15'

y\_column = 'PCT\_DIABETES\_ADULTS13'

# Calculate the correlation coefficient

corr\_coefficient = df\_Histograms[x\_column].corr(df\_Histograms[y\_column])

# Create a scatter plot with a trendline

plt.figure(figsize=(10, 10))

sns.regplot(x=x\_column, y=y\_column, data=df\_Histograms, ci=None, line\_kws={'color': 'red'})

# Display the correlation coefficient as text

plt.text(np.min(df\_Histograms[x\_column]), np.max(df\_Histograms[y\_column]), f'Correlation: {corr\_coefficient:.4f}', fontsize=14)

# Customize the plot

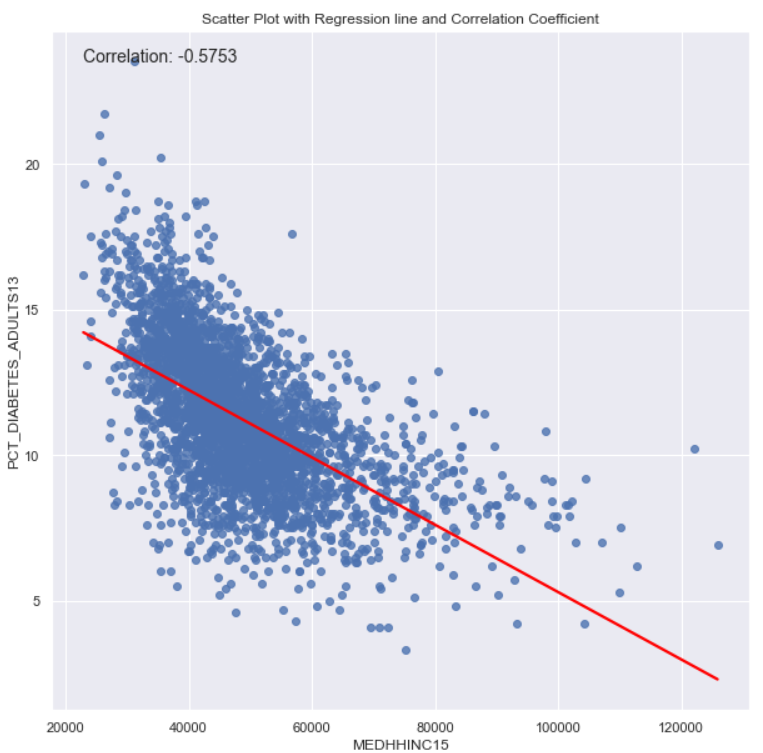
plt.title('Scatter Plot with Regression line and Correlation Coefficient')

plt.xlabel(x\_column)

plt.ylabel(y\_column)

# Show the plot

plt.show()



**## Q2b. Dropping results for regression**

dfresult = df.dropna(subset=['MEDHHINC15', 'PCT\_DIABETES\_ADULTS13'])

print(dfresult)

**## Q2b.Runing linear regression between 'MEDHHINC15' (X) & 'PCT\_DIABETES\_ADULTS13' (Y)**

# Specify the names of the columns you want to use for the regression

x\_column\_name = 'MEDHHINC15'

y\_column\_name = 'PCT\_DIABETES\_ADULTS13'

# Extract the columns from the DataFrame

x\_column = dfresult[x\_column\_name]

y\_column = dfresult[y\_column\_name]

# Add a constant term to the independent variable (X)

X = sm.add\_constant(x\_column)

# Fit the linear regression model

model = sm.OLS(y\_column, X).fit()

# Get the regression summary

summary = model.summary()

# Print the regression summary

print(summary)

A screenshot of a computer screen

Description automatically generated

**## Q2b. Interpret Results**

This linear regression is statically significant with a p value <.05, we can reject the null hypotheses. The coefficient of .01% means that with every unit increase of 'MEDHHINC15' (median household income) PCT\_DIABETES\_ADULTS13 (adult diabetes rate) decreases by .01%.

**## Q2c. Creating scatterplot with slope info**

# Specify the names of the columns you want to use for the scatter plot

x\_column = 'MEDHHINC15'

y\_column = 'PCT\_DIABETES\_ADULTS13'

# Calculate the correlation coefficient

corr\_coefficient = dfresult[x\_column].corr(dfresult[y\_column])

# Calculate the regression equation

slope, intercept, r\_value, p\_value, std\_err = stats.linregress(dfresult[x\_column], dfresult[y\_column])

regression\_equation = f'Regression Line: y = {slope:.6f}x + {intercept:.6f}'

# Create a scatter plot with a trendline

plt.figure(figsize=(8, 8))

sns.regplot(x=x\_column, y=y\_column, data=df, ci=None, line\_kws={'color': 'red'})

# Display the correlation coefficient and regression equation further up in the top right corner

plt.text(np.max(dfresult[x\_column]) - 0.1 \* (np.max(dfresult[x\_column]) - np.min(dfresult[x\_column])),

np.max(dfresult[y\_column]) - 0.05 \* (np.max(dfresult[y\_column]) - np.min(dfresult[y\_column])),

f'Correlation: {corr\_coefficient:.2f}', fontsize=12, horizontalalignment='right')

plt.text(np.max(dfresult[x\_column]) - 0.1 \* (np.max(dfresult[x\_column]) - np.min(dfresult[x\_column])),

np.max(dfresult[y\_column]) - 0.1 \* (np.max(dfresult[y\_column]) - np.min(dfresult[y\_column])),

regression\_equation, fontsize=12, horizontalalignment='right')

# Customize the plot

plt.title('Scatter Plot with Trendline and Regression Equation')

plt.xlabel(x\_column)

plt.ylabel(y\_column)

# Show the plot

plt.show()



**## Q2d. How does this regression satisfy or not satisfy the 5 assumptions of linear regression?**

Linear relationship: The scatter plot shows the negative trend between the relationship of these variables as well as the line of bets fit. There is clean linearity.

Multivariate normality: Normality is measured by Skew within the ols linear regression model and is positively skewed with a value of .024. I would consider this fairly symmetrical when normalized via z-scores. But running the regression on the log of these variables would help the data be better multivariablely normal.

Multicollinearity: The correlation coef is less than 1, so no multicollinearity.

Auto-correlation: The Durbin Watson test within the old linear regression above has a value of 1.086 which could mean there is some positive autocorrelation as the accessible range of no auto-correlation is between 1.5 to 2.5.

Homoscedasticity: The scatter plot above looks to show the analysis is homoscedastic, as the residuals look equal across the line of best fit.

**## Q2e. Add Metro**

# Specify the names of the columns you want to use for the scatter plot

x\_column = 'MEDHHINC15'

y\_column = 'PCT\_DIABETES\_ADULTS13'

# Calculate and display the correlation coefficients as text

corr\_coefficient\_0 = dfresult[dfresult['METRO13'] == 0][x\_column].corr(dfresult[dfresult['METRO13'] == 0][y\_column])

corr\_coefficient\_1 = dfresult[dfresult['METRO13'] == 1][x\_column].corr(dfresult[dfresult['METRO13'] == 1][y\_column])

# Calculate the regression equation for each category

slope\_0, intercept\_0, \_, \_, \_ = stats.linregress(dfresult[dfresult['METRO13'] == 0][x\_column], dfresult[dfresult['METRO13'] == 0][y\_column])

slope\_1, intercept\_1, \_, \_, \_ = stats.linregress(dfresult[dfresult['METRO13'] == 1][x\_column], dfresult[dfresult['METRO13'] == 1][y\_column])

# Create a scatter plot with separate trendlines and colors for each category

plt.figure(figsize=(10, 6))

sns.scatterplot(x=x\_column, y=y\_column, hue='METRO13', data=dfresult, palette={0: 'blue', 1: 'red'})

sns.lineplot(x=dfresult[dfresult['METRO13'] == 0][x\_column], y=slope\_0 \* dfresult[dfresult['METRO13'] == 0][x\_column] + intercept\_0, color='blue', label='METRO13 0')

sns.lineplot(x=dfresult[dfresult['METRO13'] == 1][x\_column], y=slope\_1 \* dfresult[dfresult['METRO13'] == 1][x\_column] + intercept\_1, color='red', label='METRO13 1')

##Customize the plot

plt.title('Scatter Plot with Trendlines and Correlation Coefficients')

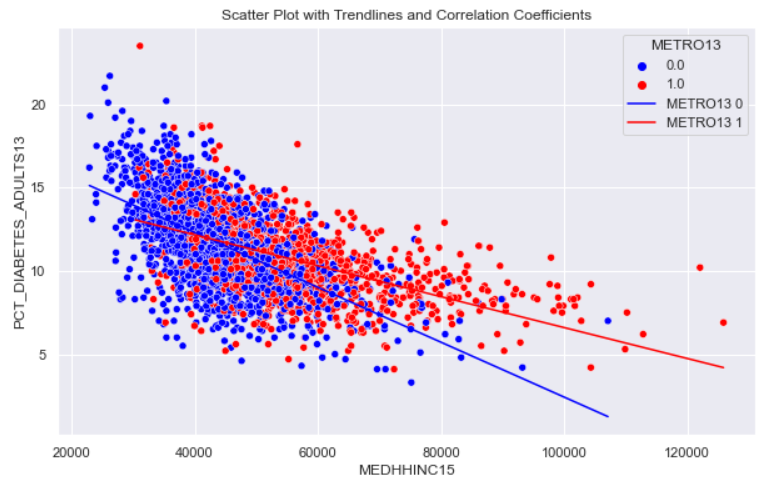
plt.xlabel(x\_column)

plt.ylabel(y\_column)

plt.legend(title='METRO13')

# Show the plot

plt.show()



**## Q2e. Add metro to regression**

# Specify the names of the columns you want to use for the scatter plot and regression analysis

x\_column = 'MEDHHINC15'

y\_column = 'PCT\_DIABETES\_ADULTS13'

categorical\_variable = 'METRO13'

# Calculate the regression equation for each category

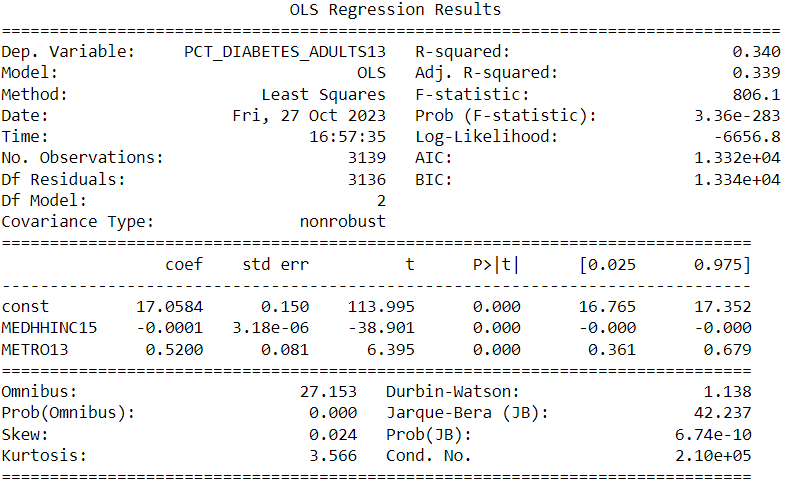
X = sm.add\_constant(dfresult[[x\_column, categorical\_variable]]) # Add a constant for the intercept

model = sm.OLS(dfresult[y\_column], X)

results = model.fit()

# Display regression results

print(results.summary())



**## Q2e. Interpret your results.**

This linear regression is still statically significant with a p value <.05, we can reject the null hypotheses. The coefficient of .01% means that with every unit increase of 'MEDHHINC15' (median household income) PCT\_DIABETES\_ADULTS13 (adult diabetes rate) decreases by .01%.

The difference between PCT\_DIABETES\_ADULTS13 (adult diabetes rate) is positively statistically significant with a p value <.05 of those living in a metro areas vs those who are not. The adult diabetes rate in those living in a metropolitan area have a chance of 52% higher rates than those not in a metropolitan area. But there could be an omitted variable bias such as population density that could be considered a confounding variable.

**## Q2e Add more variables into the regression**

# Specify the names of the columns you want to use for the scatter plot and regression analysis

x\_column = 'MEDHHINC15'

y\_column = 'PCT\_DIABETES\_ADULTS13'

categorical\_variable = 'METRO13'

additional\_vars = ['FFRPTH16', 'RECFACPTH16'] # Add the names of the additional independent variables

# Calculate the regression equation for each category

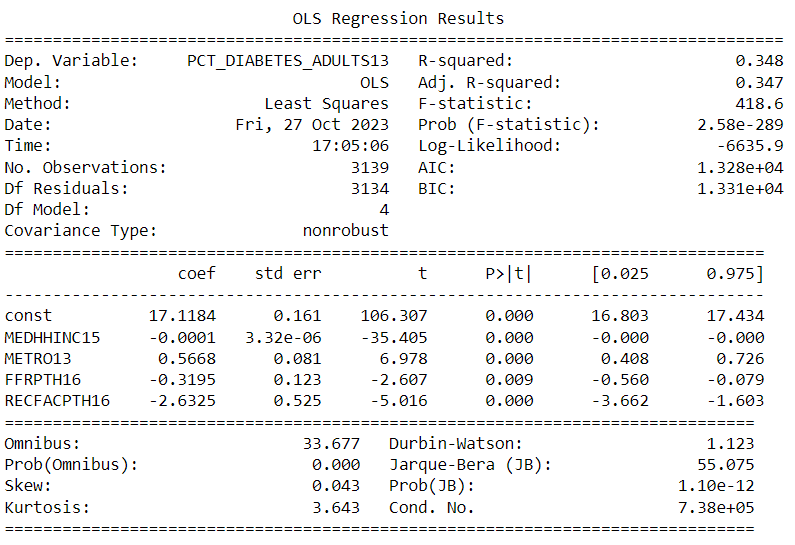
X = sm.add\_constant(dfresult[[x\_column, categorical\_variable] + additional\_vars]) # Include the additional independent variables

model = sm.OLS(dfresult[y\_column], X)

results = model.fit()

# Display regression results

print(results.summary())



**## Q2e . Interpret your results. What are potential confounders? Are there other**

**variables that may influence the outcome independently?**

This muti-linear regression is still statically significant with a p value <.05, we can reject the null hypotheses. The coefficient of .01% means that with every unit increase of 'MEDHHINC15' (median household income) PCT\_DIABETES\_ADULTS13 (adult diabetes rate) decreases by .01%.

The adult diabetes rate in those living in a metropolitan area have a chance of 56% higher rates than those not in a metropolitan area now with the additional variables included within the model. All variables are stat. sig. but FFRPTH is the least significant. RECFAC (rec facilities/1,000 pop) seems to have the highest negative coefficient relationship with every unit increase leading to a potentially reducing adult diabetes rate by 262%. Again, there could be many omitted variables biases such as access to grocery stores, or even region in the US. But more research would need to be gone on how wealth and access to preventative health care for diabetes impacts the adult rate of diabetes.