Diabetes, obesity, and inactivity: A multi-linear regression of three peas in a pod

A Punchline Report

Authors:

Suraj Basavaraj Rajalod - 02131154

Deeksha Mallampet - 02120800

Pavan Kumar Gummuluru - 02121263

Vinayak Gururaj Sonter - 02136799

ISSUES:

The data provided contains the information about people diagnosed with diabetics, obesity and inactivity across various counties in the United States in the year 2018.

The data-base includes factors like FIPS, county, state, year and percentage of people diagnosed.

We address the following the questions based on the collected data-base:

- Can we predict the diabetic data based on the inactivity data with the help of Linear Regression model using mathematical statistic plots?
- Can we analyze diabetic value by inactivity and obesity values using Multi-Linear Regression?
- We will check whether there is any heteroscedasticity in the residuals and how that might affect the accuracy of linear model of prediction?

FINDINGS:

- While we were working on the data through statistical plots, we found that diabetic data and inactivity data are normally distributed and are better correlated than the rest of the data. So, we performed linear regression model using diabetic and inactivity data points and predicted the diabetics data.
- We found correlation between all the three variables and plotted a 3D scatter plot to check whether the diabetic variable is dependent on inactivity and obesity variables.
- We performed Breusch-Pagan test to find the heteroscedasticity in the residuals.

DISCUSSION:

Firstly, we found that there are 1370 inactivity data points, 3142 diabetic data points and 363 obesity data points. This means that the percentage of people diagnosed by obesity are less compared to percentage of people diagnosed by inactivity which is also lesser than the people diagnosed by diabetics.

Secondly, we checked the distribution of data points for each variable and the correlation between the variables and found that inactivity data points are better correlated to the diabetic data points as compared to the obesity data points.

We then implemented the linear regression model to predict the diabetic data points and then performed statistical analysis on the model that is been trained.

Appendix A Method:

The data was downloaded in excel format and imported to Visual Studio code to perform coding. We loaded the inactivity, obesity and diabetic into respective data frames. We described each variable (finding mean, median, mode, standard deviation, min, max, 25%, 50% and 75% of the data). We merged the three data sets to get the intersection data points of the three variables.

Then we plotted the histogram to check the distribution of each variable (not considering FIPS as it is unique) where we observed that the obesity data is right-skewed. We plotted heat map to find the correlation between three variables and then found that the diabetic data and inactivity data are better correlated than the obesity data with respect to diabetic data. We confirmed the correlation between diabetic and inactivity data by calculating Pearson's Correlation (p-value).

We built a Linear Regression model by splitting 75% of the data as train data and the remaining 25% as the test data. We were able to predict the diabetic data with the train data set and test it with 25% of the actual data. We plotted the scatter plot to show the output.

With the merged data, we found the correlation between all the three variables so, we performed Multi Linear Regression to predict the diabetic data set with two independent variables. We then performed Breusch-Pagan Test on this model and found that p-value is not more than 0.05 which tells that there is no heteroscedasticity and thereby we can reject the null hypothesis. We plotted 3D scatter plot to show the output of multi linear regression.

Appendix B Results:

A pair plot was created to visualize the relationships between %Diabetic, %Obese and %Inactivity.

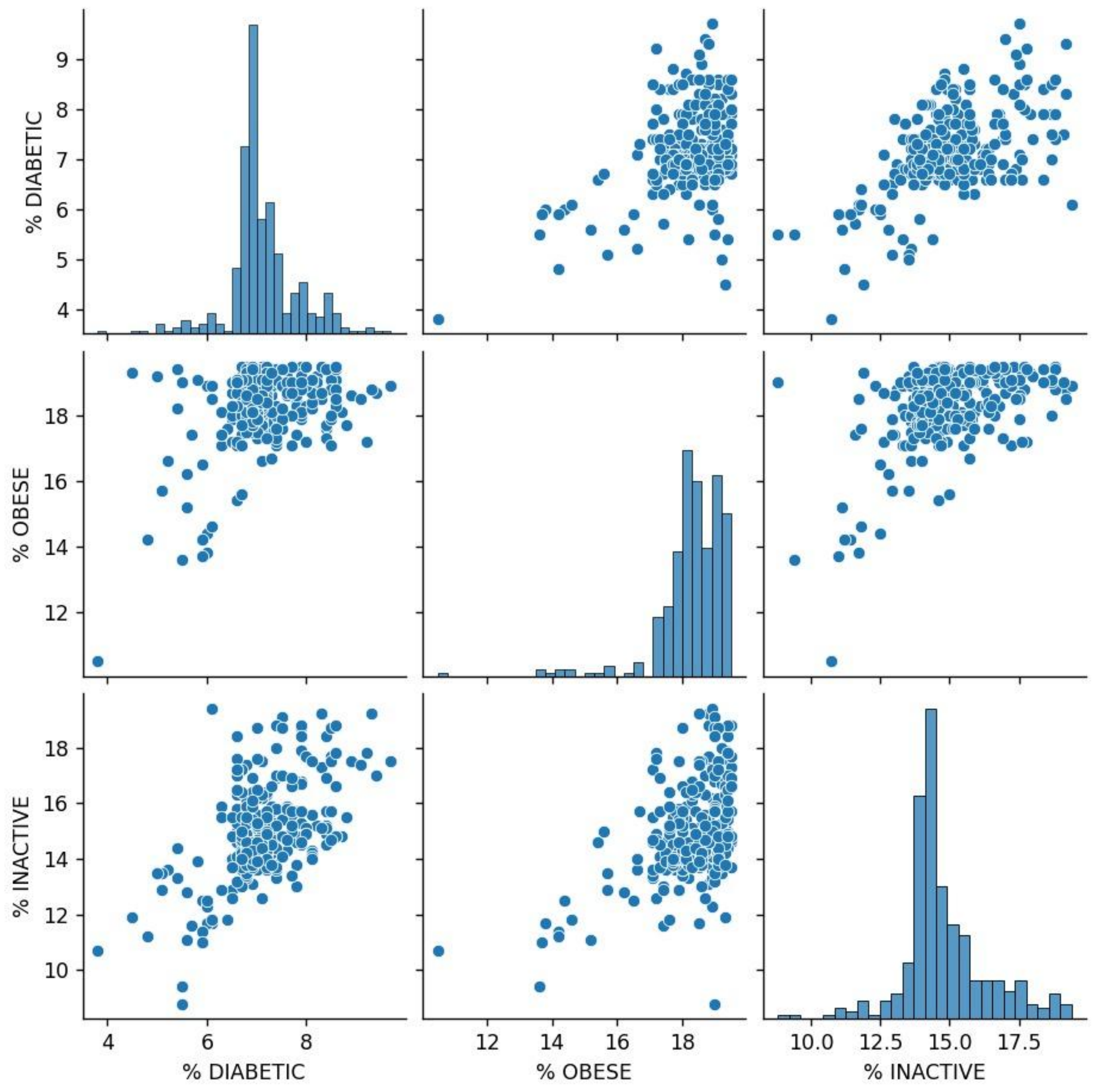


figure1. Pair plot

We then plotted the histogram where we found the diabetic data and inactivity data are centre skewed whereas, the obesity data is left skewed.

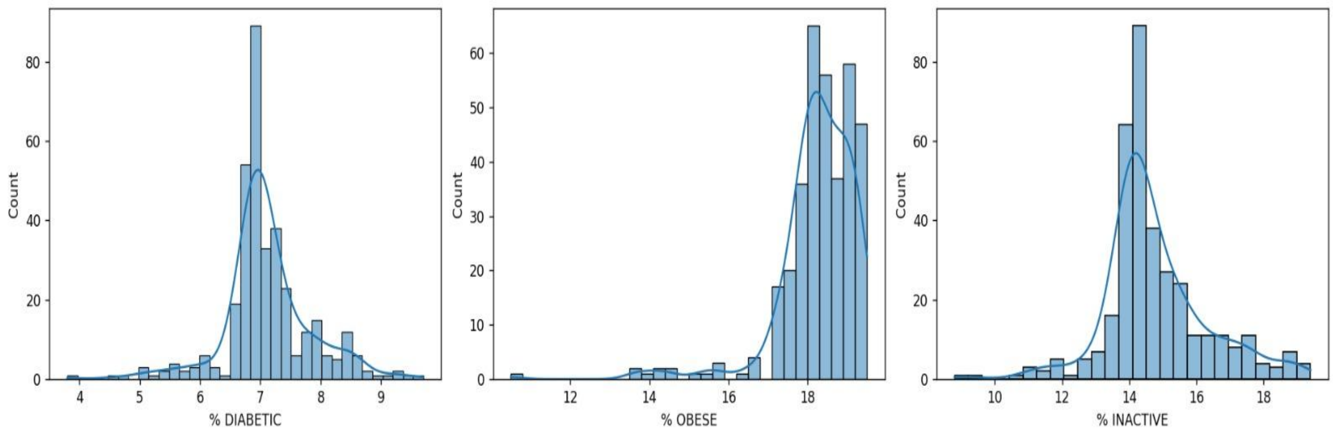
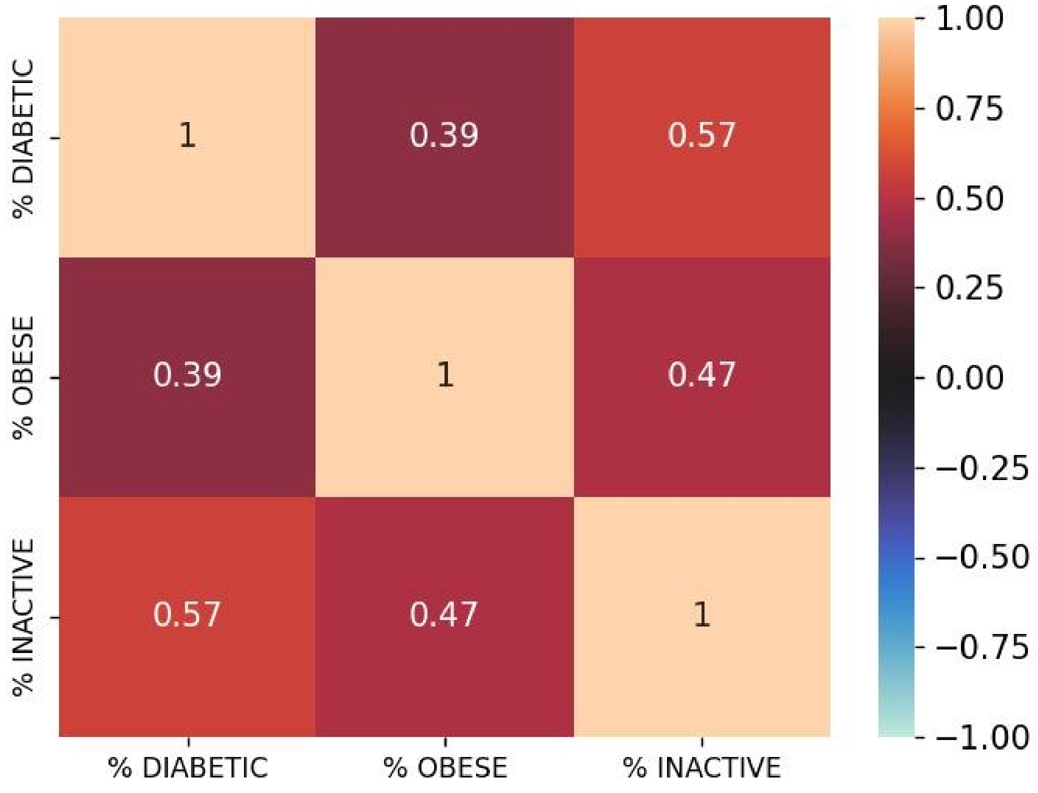


figure2. Histogram

Then, a correlation heat map was generated to visualize the correlation of coefficients between the variables.



	% DIABETIC	% OBESE	% INACTIVE
% DIABETIC	1	0.39	0.57
% OBESE	0.39	1	0.47
% INACTIVE	0.57	0.47	1

figure 3. Correlation Heat map

Scatter plot for diabetic and inactivity.

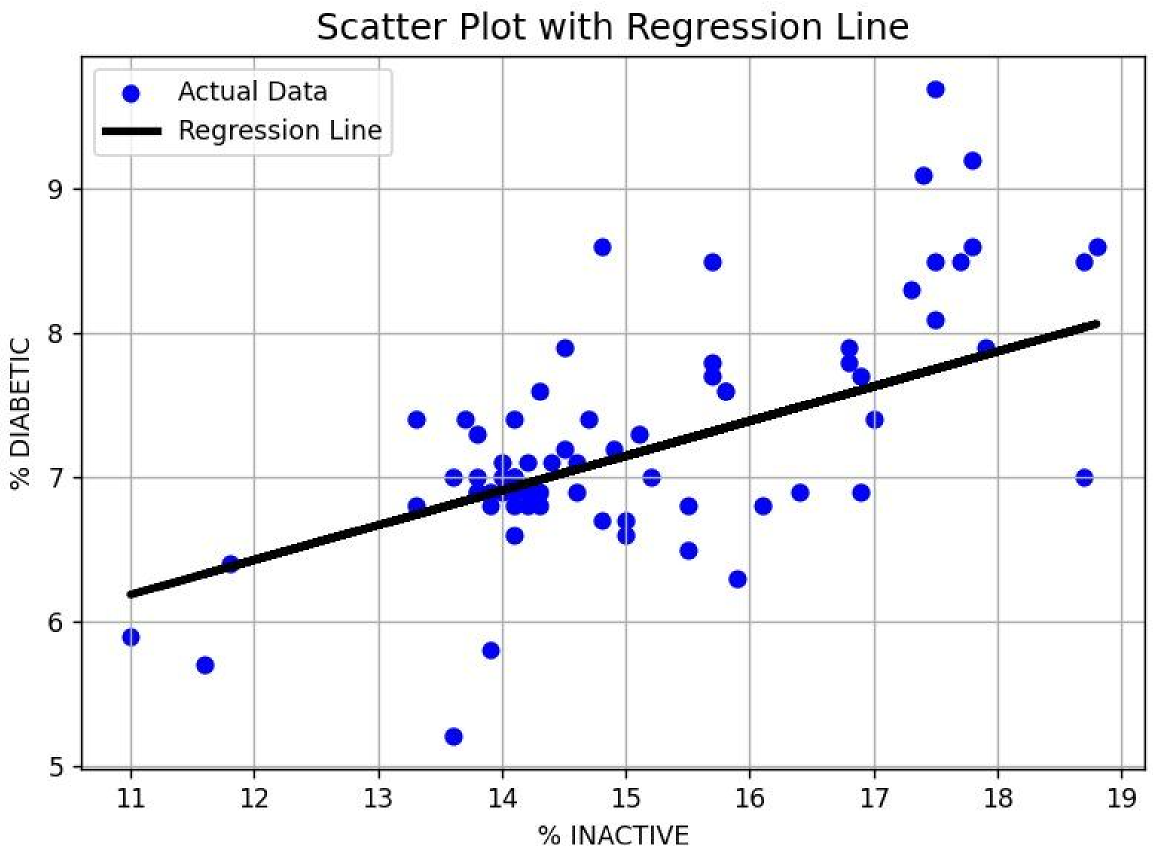


figure 4. Scatter Plot with Regression Line

Scatter plot for diabetic, obesity and diabetic, inactivity with respect to Multi-linear Regression.

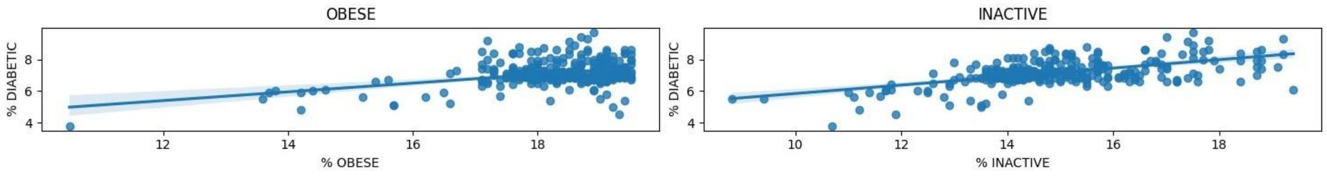
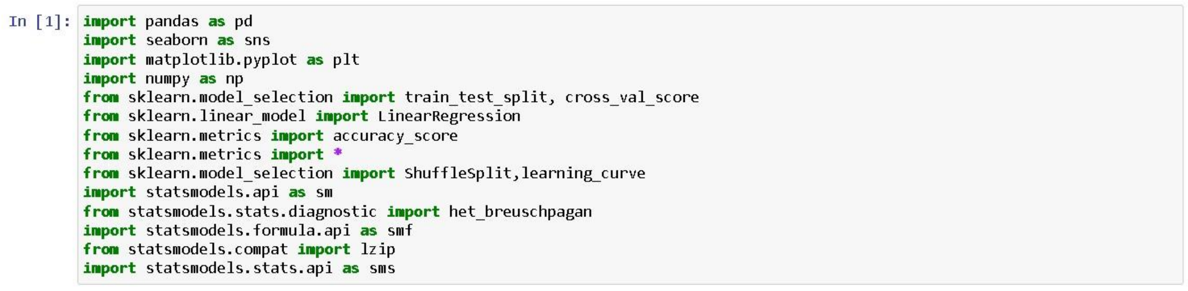


figure5:scatter plot

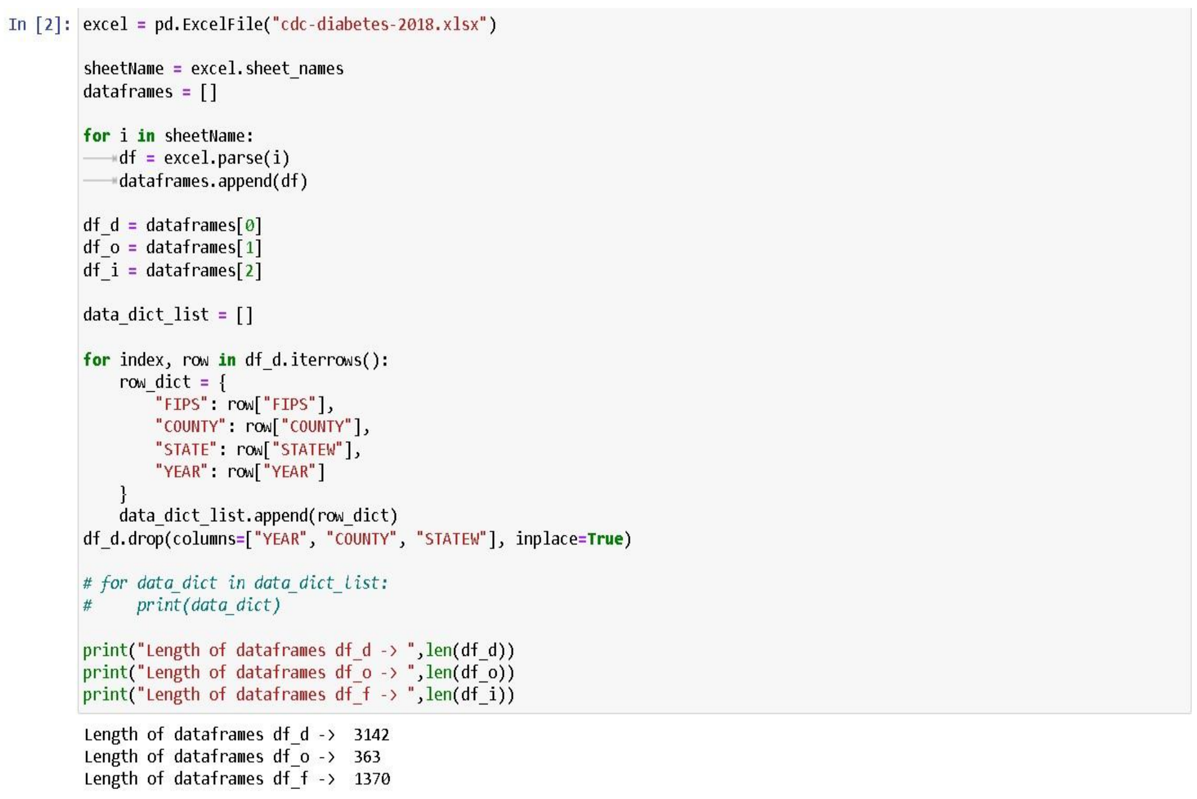
Appendix C Code:

The code is implemented in Jupyter notebook and the images of each cell is shown below



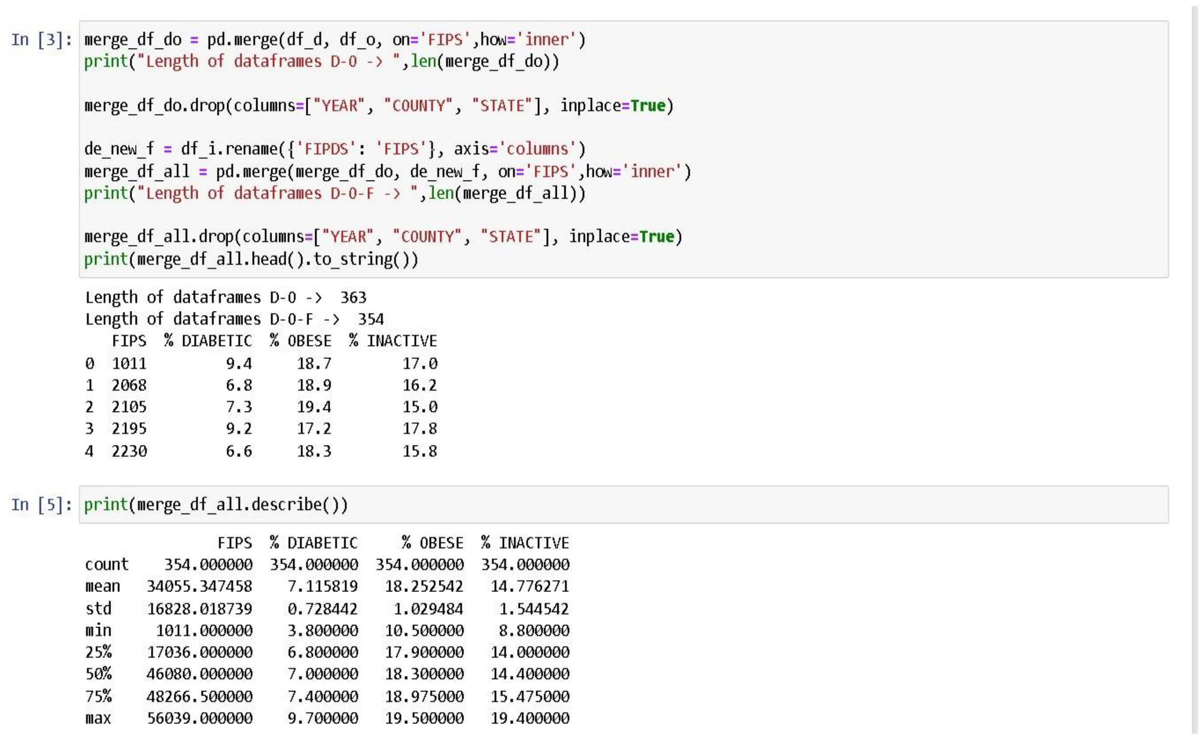
```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        import numpy as np
        from sklearn.model_selection import train_test_split, cross_val_score
        from sklearn.linear_model import LinearRegression
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import *
        from sklearn.model_selection import ShuffleSplit,learning_curve
        import statsmodels.api as sm
        from statsmodels.stats.diagnostic import het_breuschpagan
        import statsmodels.formula.api as smf
        from statsmodels.compat import lzip
        import statsmodels.stats.api as sms
```

These are the packages that are required in the project



```
In [2]: excel = pd.ExcelFile("cdc-diabetes-2018.xlsx")

        sheetName = excel.sheet_names
        dataframes = []

        for i in sheetName:
            df = excel.parse(i)
            dataframes.append(df)

        df_d = dataframes[0]
        df_o = dataframes[1]
        df_i = dataframes[2]

        data_dict_list = []

        for index, row in df_d.iterrows():
            row_dict = {
                "FIPS": row["FIPS"],
                "COUNTY": row["COUNTY"],
                "STATE": row["STATEW"],
                "YEAR": row["YEAR"]
            }
            data_dict_list.append(row_dict)
        df_d.drop(columns=["YEAR", "COUNTY", "STATEW"], inplace=True)

        # for data_dict in data_dict_list:
        #     print(data_dict)

        print("Length of dataframes df_d -> ",len(df_d))
        print("Length of dataframes df_o -> ",len(df_o))
        print("Length of dataframes df_f -> ",len(df_i))

        Length of dataframes df_d ->  3142
        Length of dataframes df_o ->  363
        Length of dataframes df_f ->  1370
```

The data frames are created from the excel sheet that is been imported. After loading the data into data frames we print the length of each dataframes.



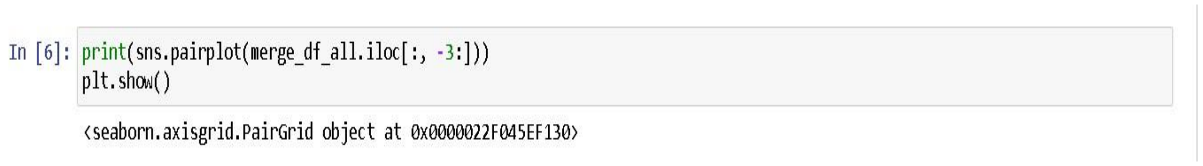
```
In [3]: merge_df_do = pd.merge(df_d, df_o, on='FIPS',how='inner')
        print("Length of dataframes D-O -> ",len(merge_df_do))

        merge_df_do.drop(columns=["YEAR", "COUNTY", "STATE"], inplace=True)

        de_new_f = df_i.rename({'FIPDS': 'FIPS'}, axis='columns')
        merge_df_all = pd.merge(merge_df_do, de_new_f, on='FIPS',how='inner')
        print("Length of dataframes D-O-F -> ",len(merge_df_all))

        merge_df_all.drop(columns=["YEAR", "COUNTY", "STATE"], inplace=True)
        print(merge_df_all.head().to_string())
```

```
Length of dataframes D-O ->  363
Length of dataframes D-O-F ->  354
   FIPS  % DIABETIC  % OBESE  % INACTIVE
0  1011         9.4     18.7        17.0
1  2068         6.8     18.9        16.2
2  2105         7.3     19.4        15.0
3  2195         9.2     17.2        17.8
4  2230         6.6     18.3        15.8
```

```
In [5]: print(merge_df_all.describe())
```

```
               FIPS  % DIABETIC     % OBESE  % INACTIVE
count    354.000000  354.000000  354.000000  354.000000
mean   34055.347458    7.115819   18.252542   14.776271
std    16828.018739    0.728442    1.029484    1.544542
min     1011.000000    3.800000   10.500000    8.800000
25%    17036.000000    6.800000   17.900000   14.000000
50%    46080.000000    7.000000   18.300000   14.400000
75%    48266.500000    7.400000   18.975000   15.475000
max    56039.000000    9.700000   19.500000   19.400000
```

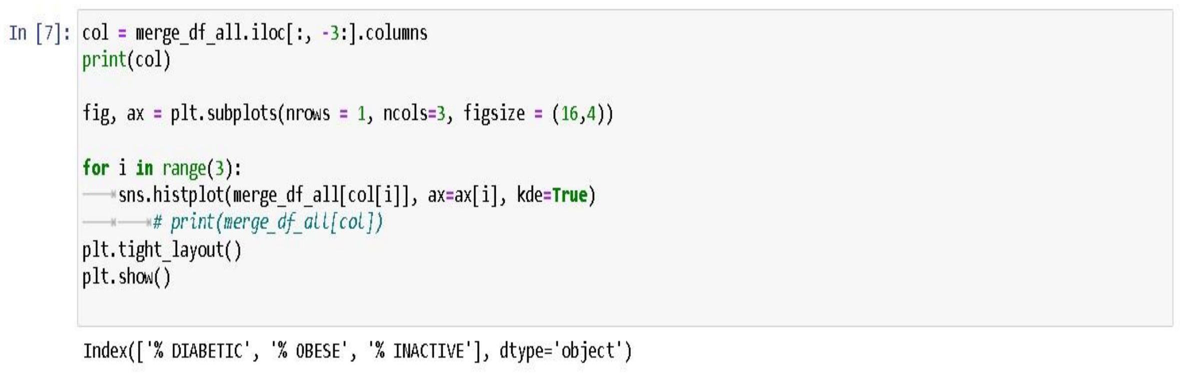
The data frames are merged and later describe function is used to describe the count, mean, standard deviation and other statistical values.



```
In [6]: print(sns.pairplot(merge_df_all.iloc[:, -3:]))
        plt.show()
```

```
<seaborn.axisgrid.PairGrid object at 0x0000022F045EF130>
```

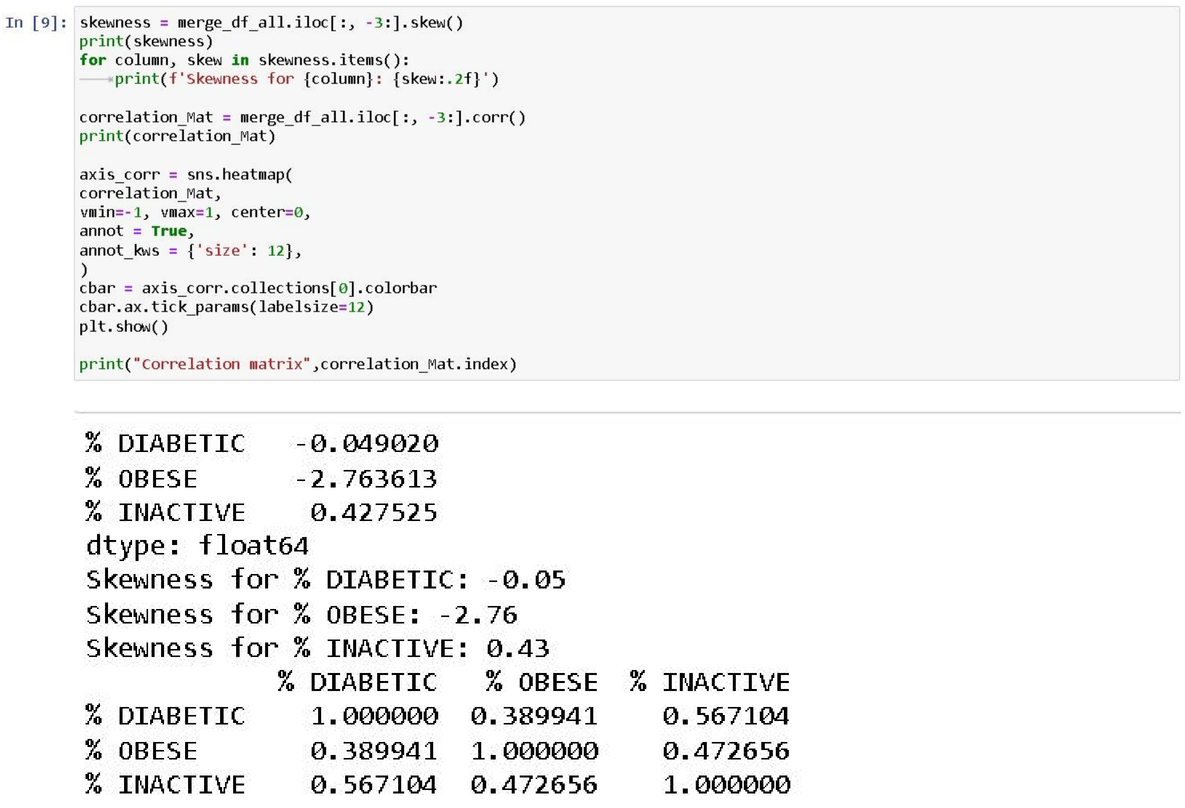
The pair plot is already been showed in Appendix B figure 1.



```
In [7]: col = merge_df_all.iloc[:, -3:].columns
        print(col)

        fig, ax = plt.subplots(nrows = 1, ncols=3, figsize = (16,4))

        for i in range(3):
            sns.histplot(merge_df_all[col[i]], ax=ax[i], kde=True)
                # print(merge_df_all[col])
        plt.tight_layout()
        plt.show()
```

```
Index(['% DIABETIC', '% OBESE', '% INACTIVE'], dtype='object')
```

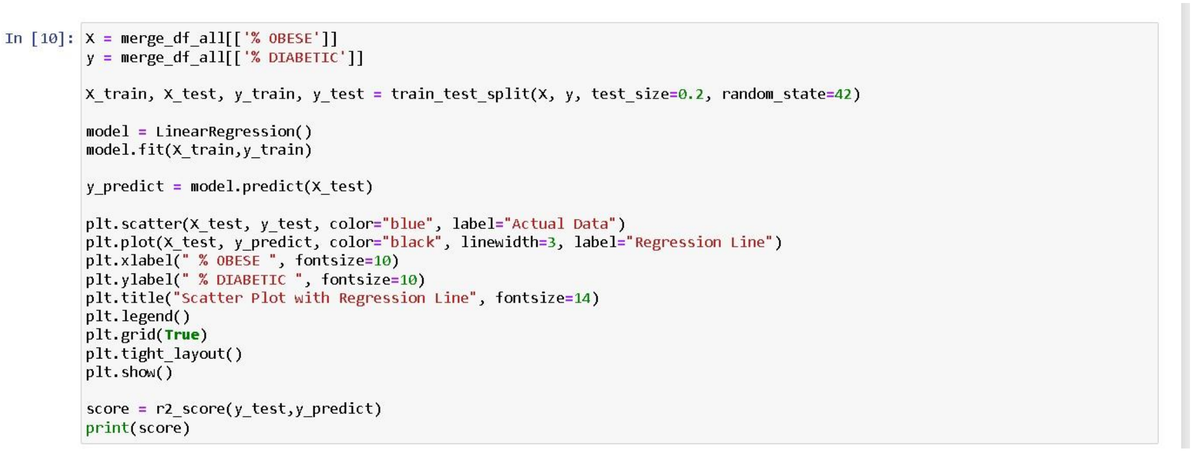
The hist plot is already shown in Appendix B figure 2.



```
In [9]: skewness = merge_df_all.iloc[:, -3:].skew()
        print(skewness)
        for column, skew in skewness.items():
            print(f'Skewness for {column}: {skew:.2f}')

        correlation_Mat = merge_df_all.iloc[:, -3:].corr()
        print(correlation_Mat)

        axis_corr = sns.heatmap(
        correlation_Mat,
        vmin=-1, vmax=1, center=0,
        annot = True,
        annot_kws = {'size': 12},
        )
        cbar = axis_corr.collections[0].colorbar
        cbar.ax.tick_params(labelsize=12)
        plt.show()

        print("Correlation matrix",correlation_Mat.index)
```

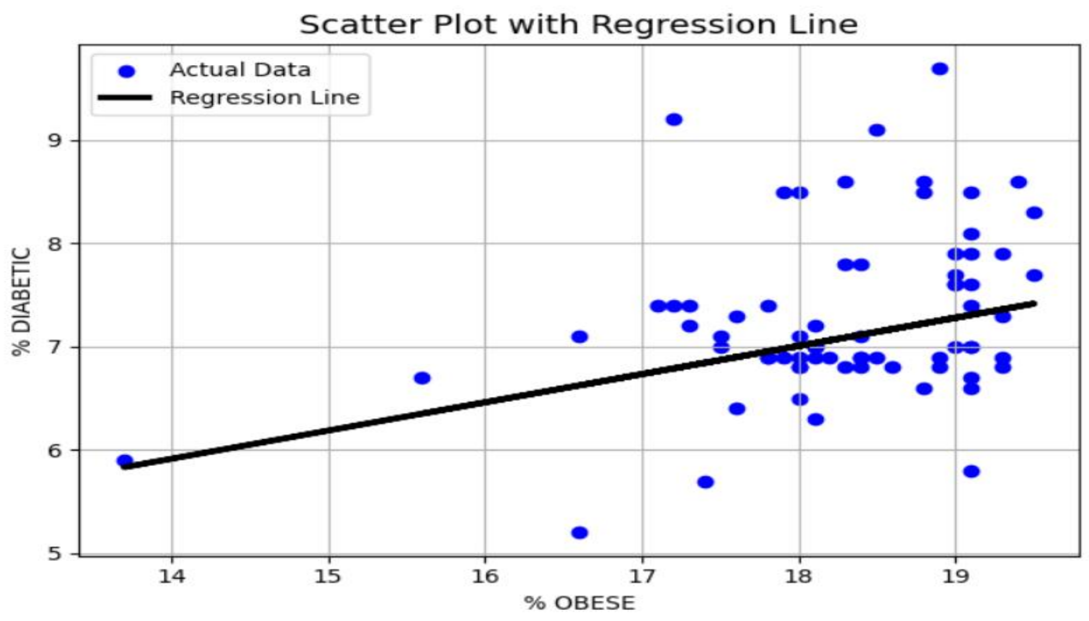
```
% DIABETIC   -0.049020
% OBESE      -2.763613
% INACTIVE    0.427525
dtype: float64
Skewness for % DIABETIC: -0.05
Skewness for % OBESE: -2.76
Skewness for % INACTIVE: 0.43
            % DIABETIC   % OBESE  % INACTIVE
% DIABETIC    1.000000  0.389941    0.567104
% OBESE       0.389941  1.000000    0.472656
% INACTIVE    0.567104  0.472656    1.000000
```

This is used to find the skewness of the variables in the merged data frames. The heat map is already shown in Appendix B figure 3



```
In [10]: X = merge_df_all[['% OBESE']]
         y = merge_df_all[['% DIABETIC']]

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         model = LinearRegression()
         model.fit(X_train,y_train)

         y_predict = model.predict(X_test)

         plt.scatter(X_test, y_test, color="blue", label="Actual Data")
         plt.plot(X_test, y_predict, color="black", linewidth=3, label="Regression Line")
         plt.xlabel(" % OBESE ", fontsize=10)
         plt.ylabel(" % DIABETIC ", fontsize=10)
         plt.title("Scatter Plot with Regression Line", fontsize=14)
         plt.legend()
         plt.grid(True)
         plt.tight_layout()
         plt.show()

         score = r2_score(y_test,y_predict)
         print(score)
```

In here we prepared a linear regression model between Obesity and Diabetic data variables



```
0.049706086551072226
```

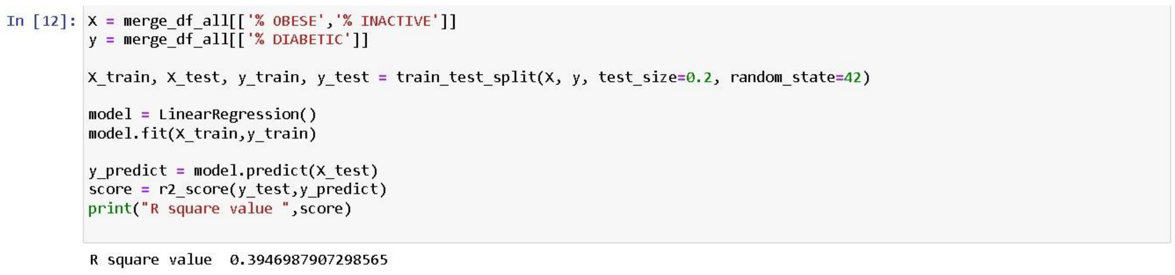
figure 6: scatter plot with regression line

The above figure is the scatter plot with the regression between obesity and diabetics.

```
In [11]: X = merge_df_all[['% INACTIVE']]
         y = merge_df_all[['% DIABETIC']]

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         model = LinearRegression()
         model.fit(X_train,y_train)

         y_predict = model.predict(X_test)

         plt.scatter(X_test, y_test, color="blue", label="Actual Data")
         plt.plot(X_test, y_predict, color="black", linewidth=3, label="Regression Line")
         plt.xlabel(" % INACTIVE ", fontsize=10)
         plt.ylabel(" % DIABETIC ", fontsize=10)
         plt.title("Scatter Plot with Regression Line", fontsize=14)
         plt.legend()
         plt.grid(True)
         plt.tight_layout()
         plt.show()

         score = r2_score(y_test,y_predict)
         print("R square value ",score)

         mean_y = np.mean(y_test)
         tss = np.sum((y_test - mean_y) ** 2)
         rss = np.sum((y_test - y_predict) ** 2)
         r_squared = 1 - (rss / tss)
         print("R-squared:", r_squared)
```

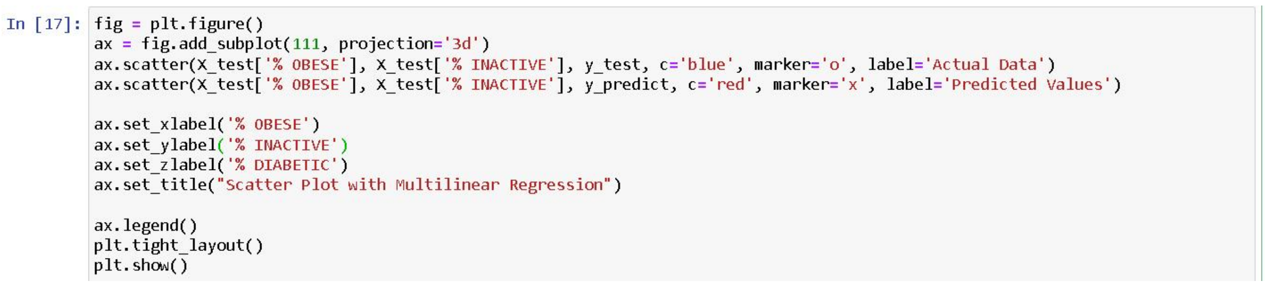
```
R square value  0.4329549953709547
R-squared: % DIABETIC    0.432955
dtype: float64
```

This is the code for linear regression model for Inactive data points and diabetic data points. The scatter plot with regression line is mention in Appendix B figure 4.



```
In [12]: X = merge_df_all[['% OBESE','% INACTIVE']]
         y = merge_df_all[['% DIABETIC']]

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

         model = LinearRegression()
         model.fit(X_train,y_train)

         y_predict = model.predict(X_test)
         score = r2_score(y_test,y_predict)
         print("R square value ",score)

         R square value  0.3946987907298565
```

This part of code is the implementation of multiple linear regression of the three variables.



```
In [17]: fig = plt.figure()
         ax = fig.add_subplot(111, projection='3d')
         ax.scatter(X_test['% OBESE'], X_test['% INACTIVE'], y_test, c='blue', marker='o', label='Actual Data')
         ax.scatter(X_test['% OBESE'], X_test['% INACTIVE'], y_predict, c='red', marker='x', label='Predicted Values')

         ax.set_xlabel('% OBESE')
         ax.set_ylabel('% INACTIVE')
         ax.set_zlabel('% DIABETIC')
         ax.set_title("Scatter Plot with Multilinear Regression")

         ax.legend()
         plt.tight_layout()
         plt.show()
```

This part of code is used to plot the scatter plot with predicted values for multiple linear regression in 3D

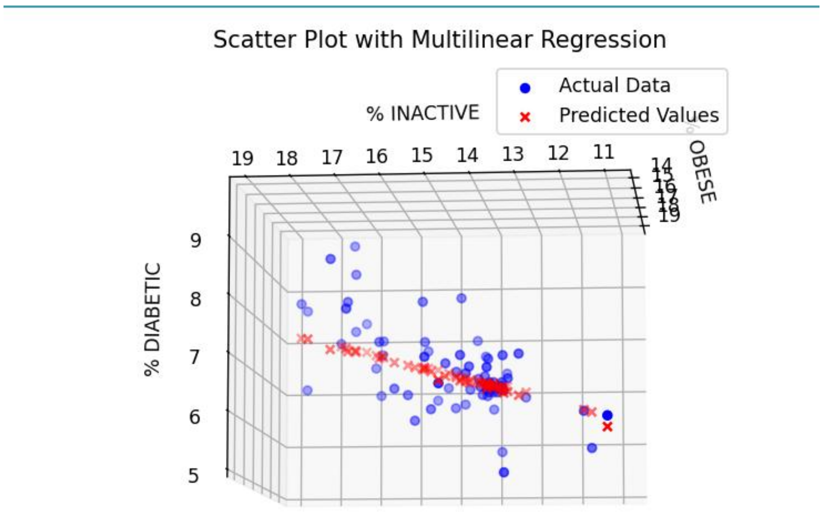
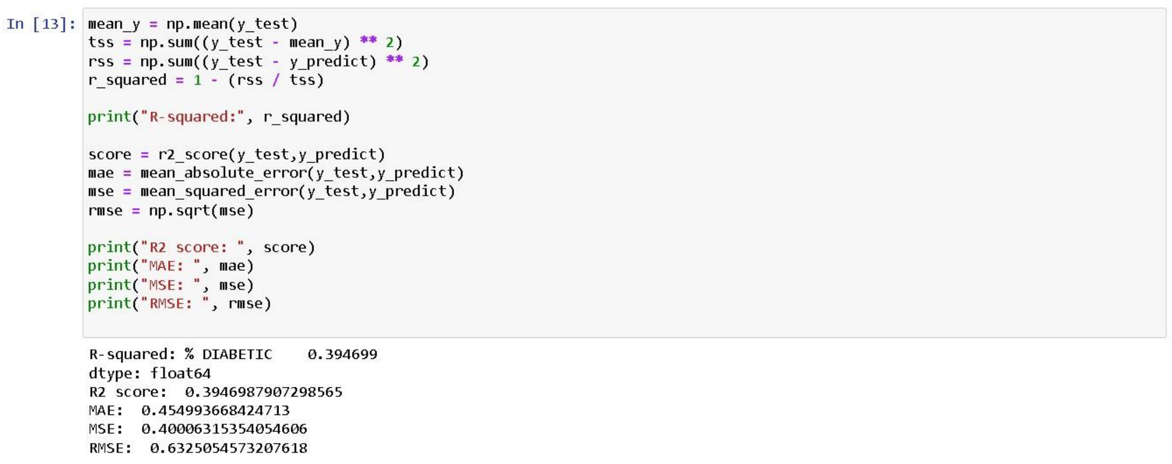


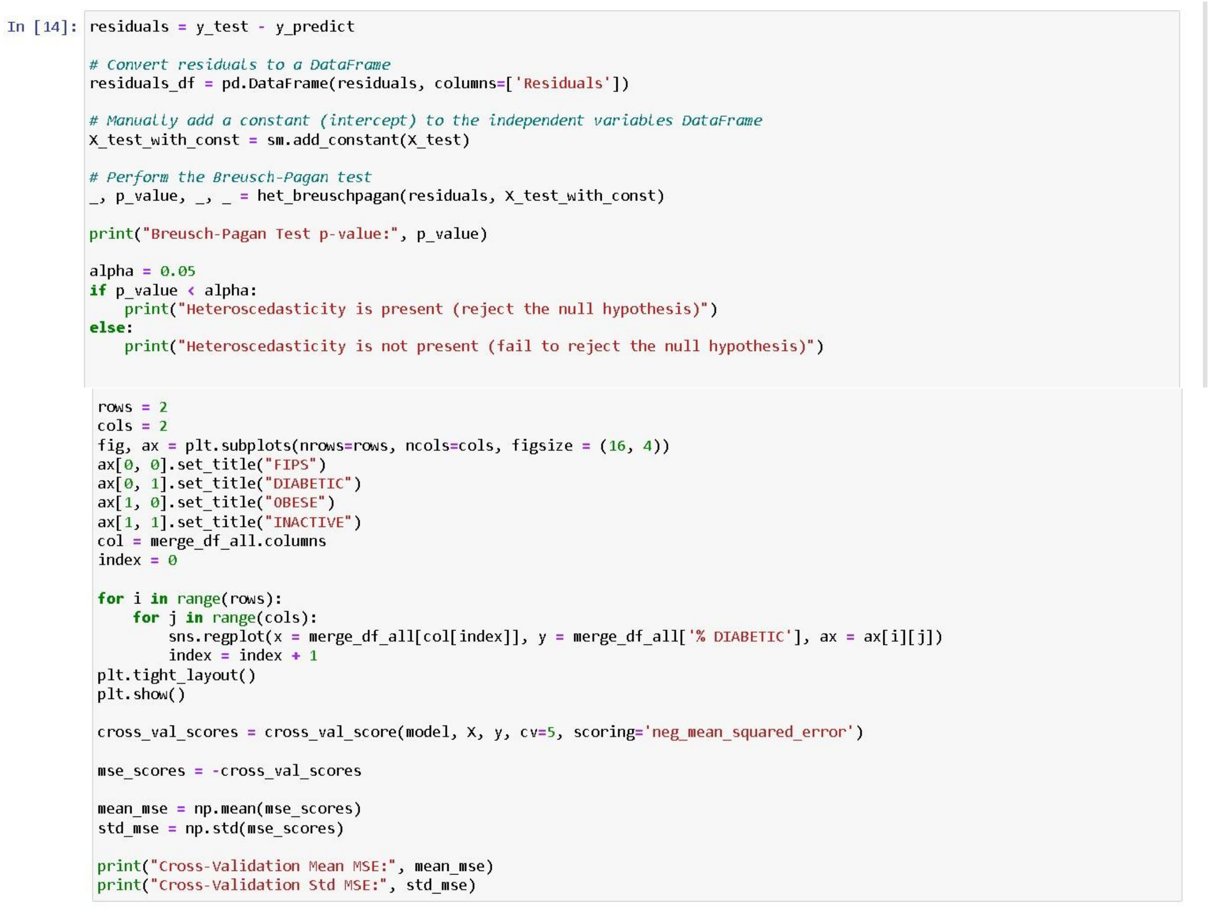
figure 7: 3D scatter plot with predicted values



```
In [13]: mean_y = np.mean(y_test)
         tss = np.sum((y_test - mean_y) ** 2)
         rss = np.sum((y_test - y_predict) ** 2)
         r_squared = 1 - (rss / tss)

         print("R-squared:", r_squared)

         score = r2_score(y_test,y_predict)
         mae = mean_absolute_error(y_test,y_predict)
         mse = mean_squared_error(y_test,y_predict)
         rmse = np.sqrt(mse)

         print("R2 score: ", score)
         print("MAE: ", mae)
         print("MSE: ", mse)
         print("RMSE: ", rmse)
```

```
R-squared: % DIABETIC    0.394699
dtype: float64
R2 score:  0.3946987907298565
MAE:  0.454993668424713
MSE:  0.40006315354054606
RMSE:  0.6325054573207618
```

We also print the R-squared value of multiple linear regression, Mean absolute error, Mean square Error and Root mean square error.

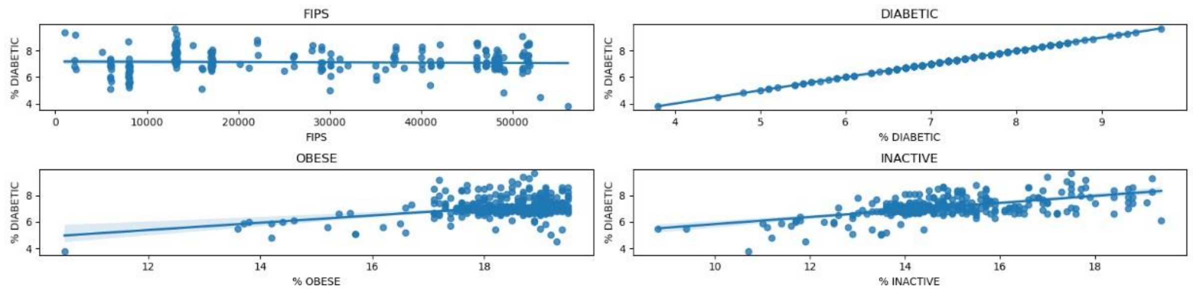


```
In [14]: residuals = y_test - y_predict

         # Convert residuals to a DataFrame
         residuals_df = pd.DataFrame(residuals, columns=['Residuals'])

         # Manually add a constant (intercept) to the independent variables DataFrame
         X_test_with_const = sm.add_constant(X_test)

         # Perform the Breusch-Pagan test
         _, p_value, _, _ = het_breuschpagan(residuals, X_test_with_const)

         print("Breusch-Pagan Test p-value:", p_value)

         alpha = 0.05
         if p_value < alpha:
             print("Heteroscedasticity is present (reject the null hypothesis)")
         else:
             print("Heteroscedasticity is not present (fail to reject the null hypothesis)")
```

```
rows = 2
cols = 2
fig, ax = plt.subplots(nrows=rows, ncols=cols, figsize = (16, 4))
ax[0, 0].set_title("FIPS")
ax[0, 1].set_title("DIABETIC")
ax[1, 0].set_title("OBESE")
ax[1, 1].set_title("INACTIVE")
col = merge_df_all.columns
index = 0

for i in range(rows):
    for j in range(cols):
        sns.regplot(x = merge_df_all[col[index]], y = merge_df_all['% DIABETIC'], ax = ax[i][j])
        index = index + 1
plt.tight_layout()
plt.show()

cross_val_scores = cross_val_score(model, X, y, cv=5, scoring='neg_mean_squared_error')

mse_scores = -cross_val_scores

mean_mse = np.mean(mse_scores)
std_mse = np.std(mse_scores)

print("Cross-Validation Mean MSE:", mean_mse)
print("Cross-Validation Std MSE:", std_mse)
```

Here we are implementing the Breusch-Pagan test to calculate the p-value. The p-value determines whether there is any heteroscedasticity in the model. If the p-value is less than 0.05 then we can reject null hypothesis

```
Breusch-Pagan Test p-value: 0.008175101856988642
Heteroscedasticity is present (reject the null hypothesis)
```



```
Cross-Validation Mean MSE: 0.37469506533023245
Cross-Validation Std MSE: 0.183068151772051
```

Here we print the p-value, cross-validation mean MSE, cross-validation standard MSE as well.

Contributions:

All the group members equally contributed to the project.