Exploring the Interplay of Diabetes, Obesity, and Inactivity in US Counties

Kruthika Reddy Murthy - 02109003
Neeraj Reddy Neelapu – 02077978

1 The Issues

In this report, we explore health trends in 2018, focusing on diabetes, obesity, and inactivity rates across US counties. Our objective is to present a lucid and non-technical analysis of these health issues impacting the diabetic population.
The data for this analysis is sourced from the Center for Disease Control and Prevention (CDC), a repository of up-to-date information on disease analysis in countries, with a focus on devising preventive methods. Our analysis addresses the following key questions:

- Correlation Exploration: What is the correlation between diabetes, obesity, and inactivity in U.S. counties?

- Predictive Modeling: Can linear regression unveil predictive insights about diabetes based on obesity and inactivity rates?

- Data Quality Concerns: How do outliers, missing values, and duplicates impact the robustness of our analyses?

Our code systematically handles these challenges, ensuring the reliability of our findings and contributing valuable insights into public health concerns.

2 Findings

We analyzed the data to understand how the rates of diabetes correlate with rates of obesity and inactivity across different counties in the United States. Our correlation analysis results were derived from the examination of the relationships between the variables (% DIABETIC, % OBESE, and % INACTIVE) in the dataset. we used statistical methods to calculate the correlation coefficients between these variables. The correlation coefficient provides a measure of the strength and direction of the linear relationship between two variables. In our case, positive correlation coefficients would indicate that higher values in one variable are associated with higher values in another. This insight is crucial for understanding the broader health trends and potential risk factors across different regions.
Linear regression models predict diabetes rates based on obesity and inactivity, emphasizing the significance of these factors. We also conducted simple linear regression analyses focusing on obesity vs. diabetes and inactivity vs. diabetes. Outliers, missing values, and duplicates were addressed meticulously, enhancing the credibility of our results.

3 Discussions

The discussion of our findings is presented in a straightforward manner, aiming to make it easily understandable. The positive correlations indicate that counties with higher obesity and inactivity rates tend to have higher diabetes rates. This observation emphasizes the need for a comprehensive approach to public health interventions, considering the interconnected nature of these factors.
In terms of implications, our results highlight the importance of addressing obesity and inactivity as preventive measures for diabetes. Public health initiatives targeting these lifestyle factors may contribute to reducing the prevalence of diabetes in different regions. The linear regression models used not only provide predictive insights but also emphasize the significance of obesity and inactivity in understanding diabetes rates.

4 Author Contributions

In this project, both authors made significant contributions. Being responsible for data collection, preprocessing, and conducting the linear regression analysis. Contributing to data analysis, interpretation of results, and visualizations, including the creation of graphs and charts. The two authors collaborated closely in merging datasets, performing statistical tests, and discussing the implications of the findings. Their combined efforts ensured a comprehensive and insightful analysis of the relationships between diabetes, obesity, and physical inactivity across U.S. counties in 2018.

5 Appendix A: Method

Data Collection:
The data was obtained from the Centers for Disease Control and Prevention (CDC). It provides county-level information on diabetes, obesity, and inactivity rates for the year 2018.

Variable Creation:
Diabetes rate (% DIABETIC)
Obesity rate (% OBESE)
Inactivity rate (% INACTIVE)

Analytic Methods
We used descriptive statistics to explore the data by using different methods as following:

- Data Preprocessing: Before analysis, the data undergoes preprocessing. This involves handling issues like duplicate records, missing values, and outliers. In your code, you handle outliers using the Interquartile Range (IQR) method and remove duplicates.

- Data Integration: The data from different sources (diabetes, obesity, and inactivity) are integrated by merging them based on common identifiers like "YEAR" and "FIPS" or "FIPDS," which represent the county codes.

- Exploratory Data Analysis (EDA): EDA is used to understand the data better. While not explicitly mentioned, this step may include visualizations, summary statistics, and checking for relationships between variables.

- Linear Regression: Linear regression is applied to analyze the relationship between two continuous variables. In this project, you perform linear regression for different combinations of variables, such as:
  % OBESE vs. % DIABETIC
  % INACTIVE vs. % DIABETIC
  % OBESE and % INACTIVE vs. % DIABETIC (Multiple Linear Regression)

- Train-Test Split: The data is split into training and testing sets to evaluate the performance of the regression models.

- Model Fitting: Linear regression models are created using the training data, and the coefficients (slopes and intercept) are estimated.

- Model Evaluation: The performance of the models is assessed using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and hypothesis tests for heteroscedasticity. You also check the coefficients for significance.

- Kurtosis and Skewness: Kurtosis and skewness of variables are calculated. These statistics describe the shape and symmetry of the data distribution.

- Residual Analysis: The skewness, kurtosis, median, and standard deviation of residuals are examined to understand the model's predictive performance.

- Visualization: You create scatterplots of actual vs. predicted values and regression lines. For multiple linear regression, you visualize the regression plane in 3D space.

- Statistical Tests: You conduct statistical tests such as the Breusch-Pagan test for heteroscedasticity.

6 Appendix B: Results

Our dataset originally contained 3,142 data points. After merging the datasets, it reduced to 354 data points, which included information about %DIABETIC, %OBESE, and %INACTIVE for all U.S. counties in the year 2018.

Before we performed data transformation and preprocessing, the data points numbered 354. Once we completed these steps, the data points were reduced to 339.

We calculated kurtosis, skewness, Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) values for three factors (%DIABETIC, %OBESE, and %INACTIVE) both before and after applying linear regression.

Before Linear Regression:
Kurtosis before linear regression - Diabetic: -0.15452043503385493
Kurtosis before linear regression - Obesity: -0.4870496991420761
Kurtosis before linear regression - Inactive: -0.8543300319009877
Skewness before linear regression - Diabetic: -0.016571543970232954
Skewness before linear regression - Obesity: 0.0739131829093501
Skewness before linear regression - Inactivity: -0.016571543970232954
MSE before linear regression - Diabetes: 0.3616292550030423
RMSE before linear regression - Diabetes: 0.6013561798161239
MSE before linear regression - Obesity: 0.3616292550030423
RMSE before linear regression - Obesity: 0.6013561798161239
MSE before linear regression - Inactivity: 0.3616292550030423
RMSE before linear regression - Inactivity: 0.6013561798161239

We performed Multiple Linear Regression to build a model that predicts diabetes based on obesity and inactivity. The model will take the form of an equation, such as:
diabetes = a + b * obesity + c * inactivity + ϵ
where a, b, and c are coefficients that will be estimated from the data.

To evaluate the performance of a regression model we used different statistical metrics, they are:

Mean Square Error (DIABETIC vs OBESE and INACTIVE): 0.3616292550030423
Mean Absolute Error (DIABETIC vs OBESE and INACTIVE): 0.4093128490591154
Root Mean Squared Error (DIABETIC vs OBESE and INACTIVE): 0.6013561798161239
Kurtosis (DIABETIC vs OBESE and INACTIVE): 1.9566336099904058
Skewness (DIABETIC vs OBESE and INACTIVE): -0.26259039480325724
Median (DIABETIC vs OBESE and INACTIVE): 0.0010092517888140584
Standard Deviation (DIABETIC vs OBESE and INACTIVE): 0.5945245486575608

We also used OLS (Ordinary Least Squares) method to obtain the results of multiple regression model

OLS Regression Results

Dep. Variable:	% DIABETIC	R-squared:	0.164
Model:	OLS	Adj. R-squared:	0.157
Method:	Least Squares	F-statistic:	26.21
Date:	Mon, 09 Oct 2023	Prob (F-statistic):	4.00e-11
Time:	12:20:17	Log-Likelihood:	-231.57
No. Observations:	271	AIC:	469.1
Df Residuals:	268	BIC:	480.0
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.2446	0.990	4.286	0.000	2.295	6.195
% OBESE	0.0063	0.058	0.110	0.913	-0.107	0.120
% INACTIVE	0.1881	0.028	6.693	0.000	0.133	0.243

...

Figure 1. Result of OLS for DIABETIC vs OBESE and INACTIVE

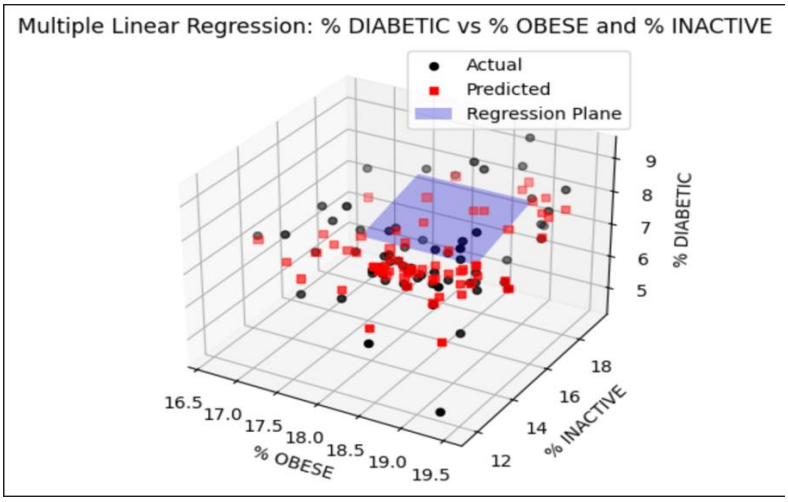


Figure 1.1 DIABETIC vs OBESE and INACTIVE

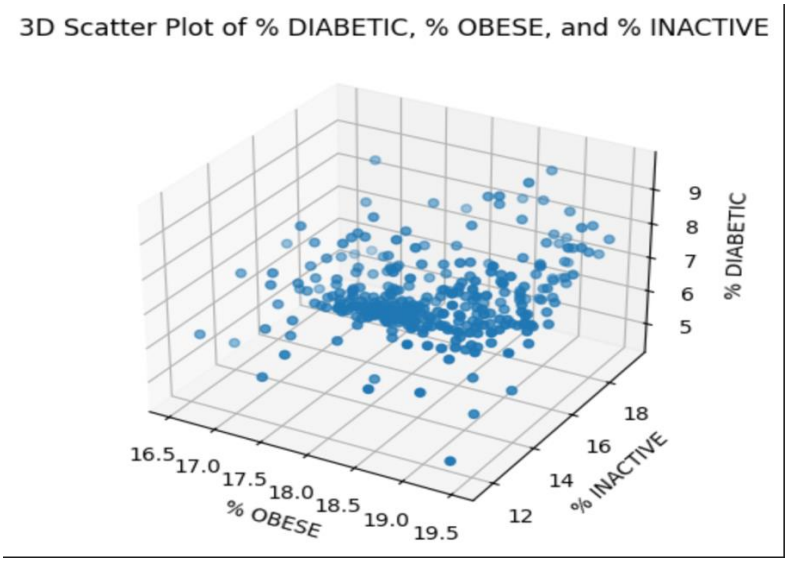


Figure 1.2 Correlation analysis of DIABETIC vs OBESE and INACTIVE

In our analysis of the relationship between diabetes and obesity, and how they relate to physical inactivity, we observed that the kurtosis value for this specific context is 1.9566. This value provides us with insights into the shape of the data distribution.

A kurtosis value greater than 3, as is the case with our kurtosis of 1.9566, indicates positive kurtosis. In simpler terms, this means that the distribution of the data has 'heavier tails' and is more 'leptokurtic' compared to a normal distribution.

A skewness value indicates the asymmetry of the data distribution. A value of 0 suggests a perfectly symmetrical distribution, where the data is evenly distributed on both sides of the mean. A negative skewness, as we have here, implies that the data distribution is skewed to the left, or negatively skewed. In our context, this means that the tail of the distribution is longer on the left side, and there may be some outliers or lower values that are dragging the distribution in that direction.

Then we performed simple Linear Regression on Obese vs Diabetics and Inactivity vs Diabetics and calculated Kurtosis, Skewness, RMSE and MSE and obtained these values:

Kurtosis after linear regression - Diabetic: 1.9439901869077971
Kurtosis after linear regression - Obesity: 1.9566336099904058
Kurtosis after linear regression - Inactive: 1.9566336099904058
Skewness after linear regression - Diabetic: -0.2578514160894123
Skewness after linear regression - Obesity: -0.192973836736394
Skewness after linear regression - Inactivity: -0.26259039480325724
MSE after linear regression - Diabetes: 0.3616292550030423
RMSE after linear regression - Diabetes: 0.6013561798161239
MSE after linear regression - Obesity: 0.29360723610109246
RMSE after linear regression - Obesity: 0.5418553645587468
MSE after linear regression - Inactivity: 0.3616292550030423
RMSE after linear regression - Inactivity: 0.6013561798161239

OLS Regression Results

Dep. Variable:	% DIABETIC	R-squared:	0.024
Model:	OLS	Adj. R-squared:	0.020
Method:	Least Squares	F-statistic:	6.561
Date:	Mon, 09 Oct 2023	Prob (F-statistic):	0.0110
Time:	12:20:23	Log-Likelihood:	-252.52
No. Observations:	271	AIC:	509.0
Df Residuals:	269	BIC:	516.2
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.4232	1.068	4.143	0.000	2.321	6.525
% OBESE	0.1485	0.058	2.561	0.011	0.034	0.263

Omnibus:	45.872	Durbin-Watson:	2.149
Prob(Omnibus):	0.000	Jarque-Bera (JB):	118.701
Skew:	0.771	Prob(JB):	1.68e-26
Kurtosis:	5.852	Cond. No.	526.

Figure 2 Result of OLS for %OBESE VS %DIABETIC

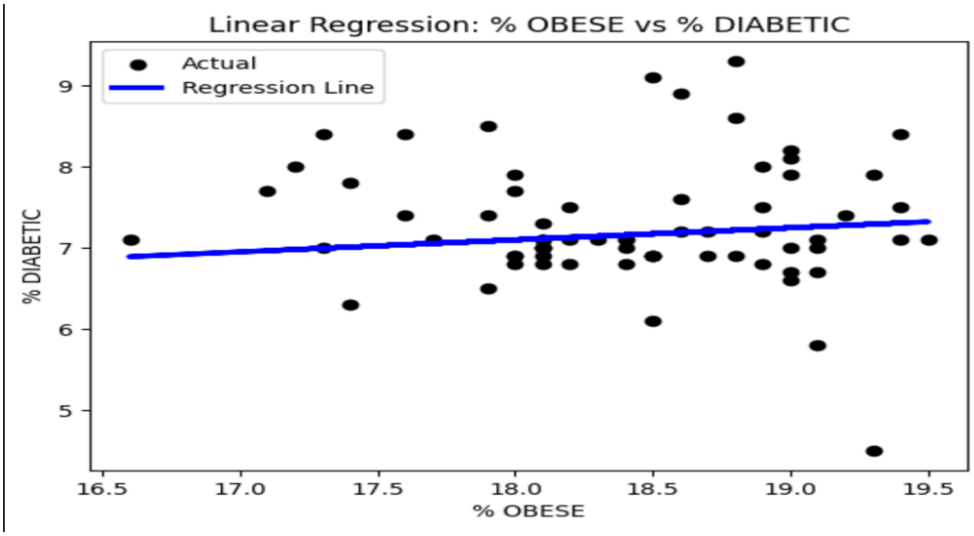


Figure 2.1 Linear regression plot of %OBESE VS %DIABETIC

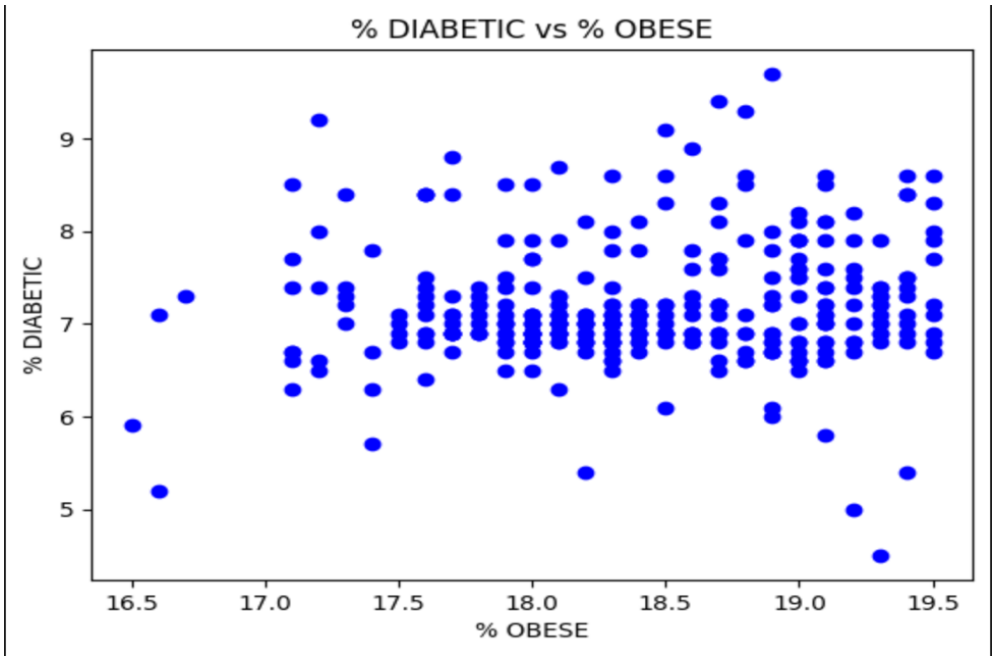


Figure 2.2 Correlation analysis of %OBESE VS %DIABETIC

OLS Regression Results

Dep. Variable:	% DIABETIC	R-squared:	0.164
Model:	OLS	Adj. R-squared:	0.160
Method:	Least Squares	F-statistic:	52.61
Date:	Mon, 09 Oct 2023	Prob (F-statistic):	4.34e-12
Time:	12:20:32	Log-Likelihood:	-231.58
No. Observations:	271	AIC:	467.2
Df Residuals:	269	BIC:	474.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.3445	0.389	11.164	0.000	3.578	5.111
% INACTIVE	0.1892	0.026	7.253	0.000	0.138	0.241

Omnibus:	21.526	Durbin-Watson:	2.058
Prob(Omnibus):	0.000	Jarque-Bera (JB):	54.653
Skew:	0.317	Prob(JB):	1.36e-12
Kurtosis:	5.106	Cond. No.	168.

Figure 3 Result of OLS for %INACTIVE VS %DIABETIC

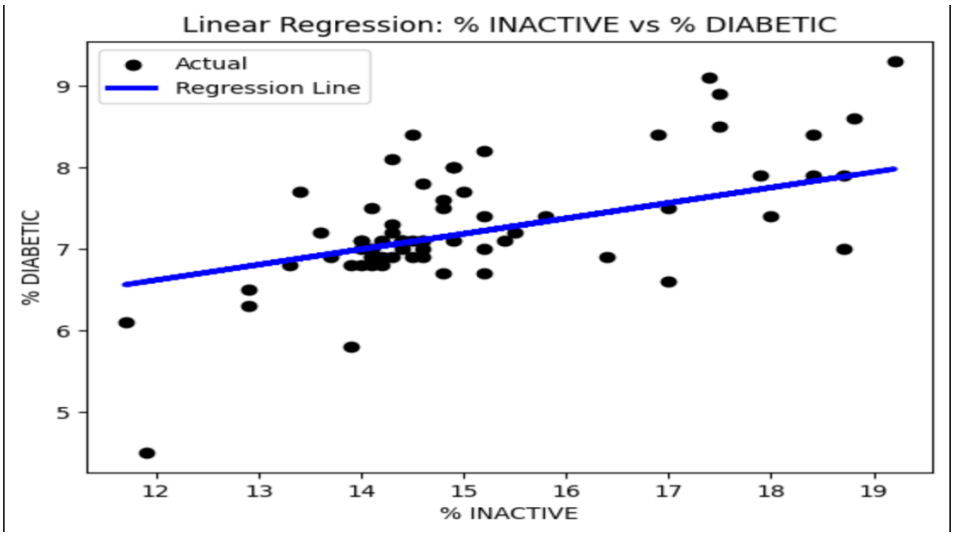


Figure 3.1 Linear regression plot of %INACTIVE VS DIABETIC

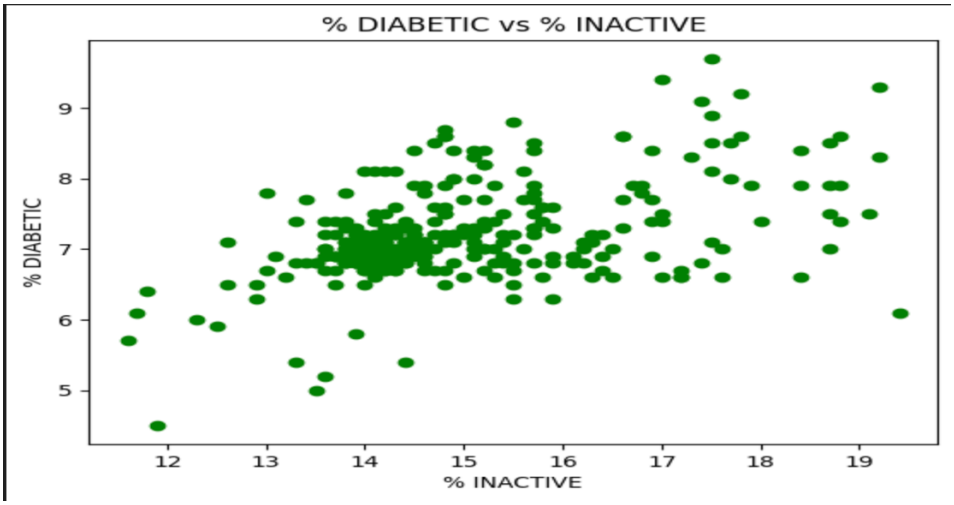


Figure 3.2 %INACTIVE VS DIABETIC

Kurtosis after linear regression:

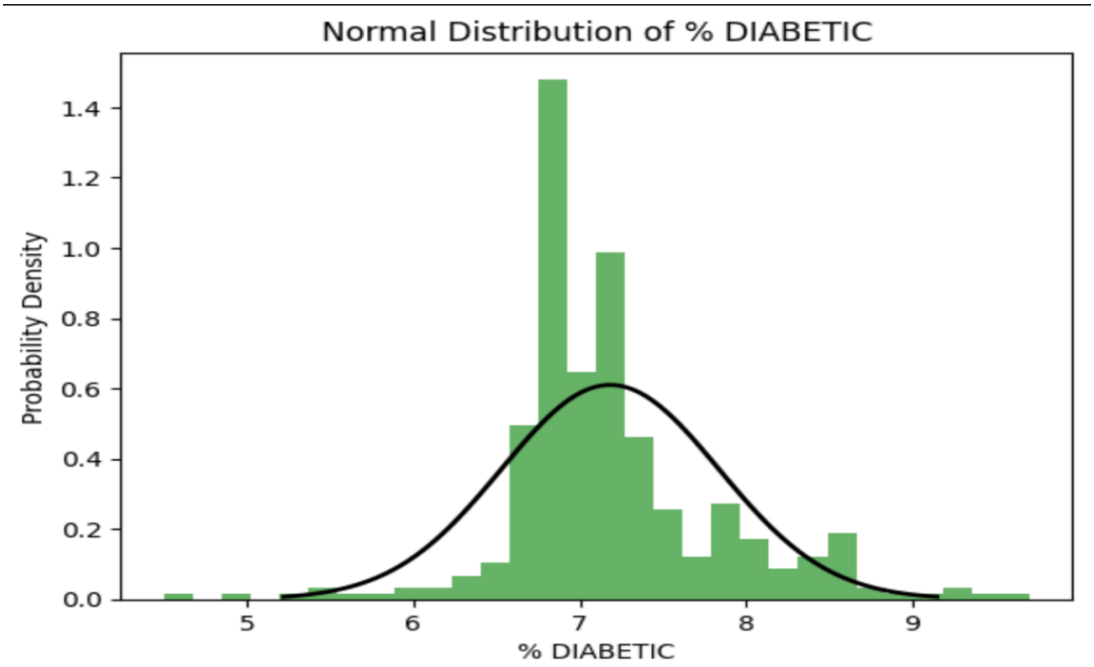
- The kurtosis values provided after linear regression represent the distribution of the residuals (differences between actual and predicted values) for each variable (Diabetic, Obesity, Inactivity).
- For Diabetic, the kurtosis is 1.944, indicating a moderately peaked distribution with tails that are lighter than a normal distribution.
- For Obesity and Inactivity, the kurtosis values are 1.957, suggesting similar characteristics.

Skewness after linear regression:

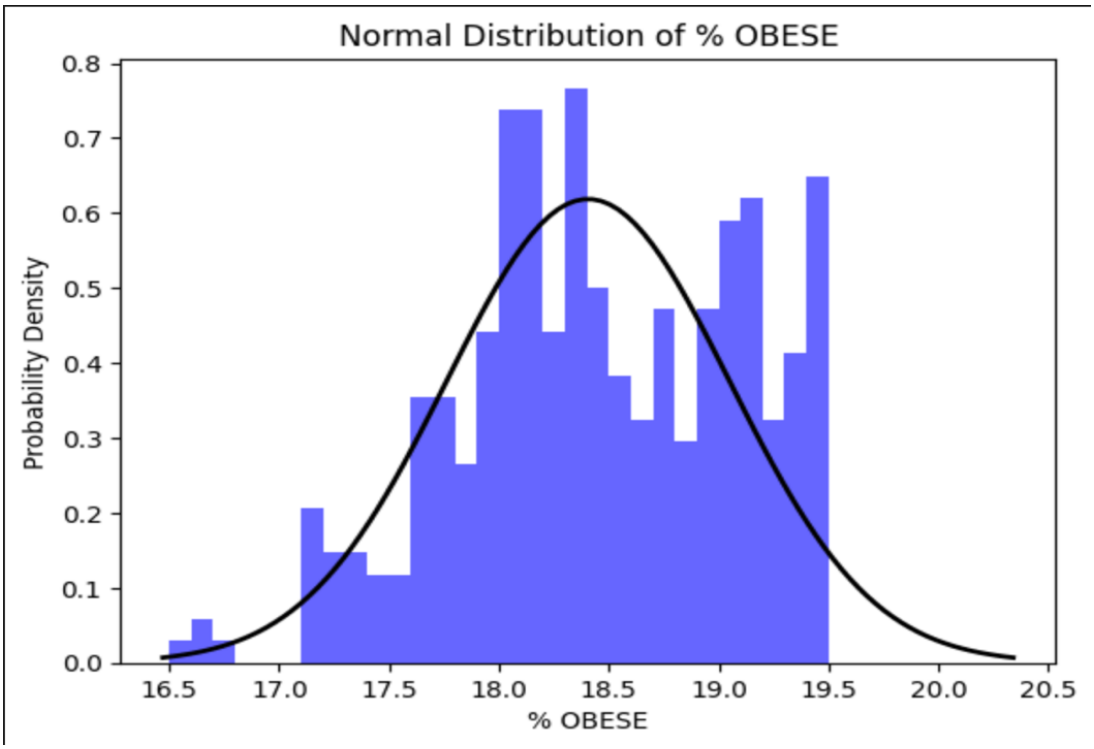
- Skewness measures the asymmetry in the distribution of data.
- For Diabetic, the skewness is -0.258, implying that the distribution is slightly skewed to the left.
- For Obesity, the skewness is -0.193, suggesting a similar but less pronounced left skew.
- For Inactivity, the skewness is -0.263, indicating a slight left skew.

Mean Squared Error (MSE) after linear regression:

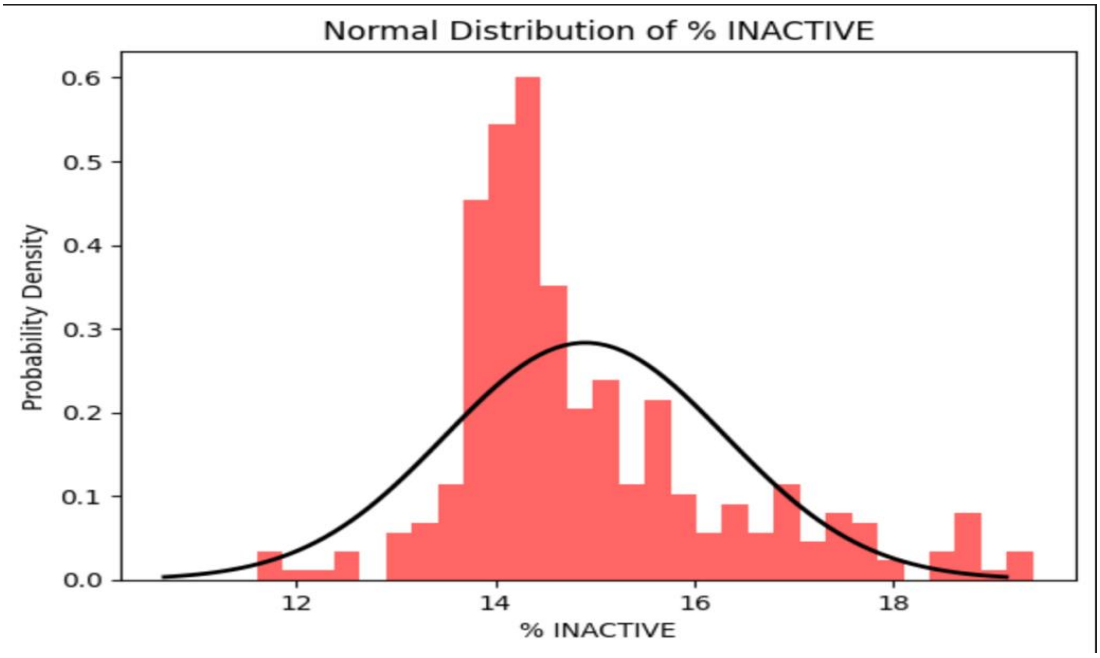
- MSE quantifies the average squared difference between actual and predicted values. Lower MSE indicates a better fit.
- For Diabetes and Inactivity, the MSE is 0.362, while for Obesity, it's 0.294.



Normal Distribution of %DIABETIC



Normal Distribution of %OBESE



Normal Distribution of %INACTIVE

7 Appendix C: Data and Code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import kurtosis, skew
from statsmodels.stats.diagnostic import het_breuschpagan
import statsmodels.api as sm
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import scipy.stats as stats
# Specify the path to your Excel file
excel_file_path = r'C:\\Users\\Kruthika reddy\\Desktop\\mth\\cdc-diabetes-2018 (1).xlsx'
# Read data from Excel sheets
df_obesity = pd.read_excel(excel_file_path, sheet_name='Obesity')
df_diabetic = pd.read_excel(excel_file_path, sheet_name='Diabetes')
df_inactive = pd.read_excel(excel_file_path, sheet_name='Inactivity')

# Before  preprocessing and transformation of data
# Merge the datasets on FIPS code and YEAR
print('df_diabetic:', df_diabetic.shape)
print('df_obesity:', df_obesity.shape)
df_merged = pd.merge(df_diabetic, df_obesity, on=['YEAR', 'FIPS'])
print('After First Merge:', df_merged.shape)
```

```
df_merged = pd.merge(df_merged, df_inactive, left_on=['YEAR', 'FIPS'], right_on=['YEAR', 'FIPDS'])
print('After Second Merge:', df_merged.shape)
print('Number of Rows and Columns in df_merged:', df_merged.shape)

# After preprocessing and transformation of data
# Function to handle outliers using IQR
def handle_outliers(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1

    # Filtering values between Q1-1.5*IQR and Q3+1.5*IQR
    df_no_outliers = df[(df[column_name] >= Q1 - 1.5 * IQR) & (df[column_name] <= Q3 + 1.5 * IQR)]

    return df_no_outliers

# Function to preprocess and transform the data
def preprocess_and_transform(df):
    # Check for and handle duplicates
    df = df.drop_duplicates()

    # Check for missing values and handle them (if needed)
    df = df.dropna()  # Uncomment this line if you want to remove rows with missing values

    # Handle outliers
    df = handle_outliers(df, df.columns[4])  # Assuming the numerical column is at index 4

    # Add any additional preprocessing or transformation steps here

    return df

# Preprocess and transform each dataset
df_obesity = preprocess_and_transform(df_obesity)
df_diabetic = preprocess_and_transform(df_diabetic)
df_inactive = preprocess_and_transform(df_inactive)

# After preprocessing and transformation of data
# Merge the datasets on FIPS code and YEAR
print('df_diabetic:', df_diabetic.shape)
print('df_obesity:', df_obesity.shape)
print('df_inactive:', df_inactive.shape)
# Merging diabetic and obesity datasets
df_merged = pd.merge(df_diabetic, df_obesity, on=['YEAR', 'FIPS'])
print('After First Merge:', df_merged.shape)
```

```
# Merging the above df merged and inactive dataset
df_merged = pd.merge(df_merged, df_inactive, left_on=['YEAR', 'FIPS'], right_on=['YEAR', 'FIPDS'])
print('After Second Merge:', df_merged.shape)
print('Number of Rows and Columns in df_merged:', df_merged.shape)

# Kurtosis, skewness, mean squared error, root mean squared error before linear regression

# Assuming y_test_diabetic_vs_obese_inactive and y_pred_diabetic_vs_obese_inactive are your variables
y_test_diabetic = y_test_diabetic_vs_obese_inactive
y_test_obesity = y_test_diabetic_vs_obese_inactive
y_test_inactive = y_test_diabetic_vs_obese_inactive

y_pred_diabetic = y_pred_diabetic_vs_obese_inactive
y_pred_obesity = y_pred_diabetic_vs_obese_inactive
y_pred_inactive = y_pred_diabetic_vs_obese_inactive

# Calculate kurtosis before linear regression
kurtosis_diabetic_before = kurtosis(df_diabetic['% DIABETIC'])
kurtosis_obesity_before = kurtosis(df_obesity['% OBESE'])
kurtosis_inactive_before = kurtosis(df_inactive['% INACTIVE'])

print('Kurtosis before linear regression - Diabetic:', kurtosis_diabetic_before)
print('Kurtosis before linear regression - Obesity:', kurtosis_obesity_before)
print('Kurtosis before linear regression - Inactive:', kurtosis_inactive_before)

# Calculate skewness before linear regression
skew_diabetic_before = skew(y_test_diabetic)
print('Skewness before linear regression - Diabetic:', skew_diabetic_before)
skew_obese_before = skew(y_test_obese)
print('Skewness before linear regression - Obesity:', skew_obese_before)
skew_inactive_before = skew(y_test_inactive)
print('Skewness before linear regression - Inactivity:', skew_inactive_before)

# Diabetes
mse_diabetic_before = mean_squared_error(y_test_diabetic , y_pred_diabetic)
rmse_diabetic_before = np.sqrt(mse_diabetic_before)
print('MSE before linear regression - Diabetes:', mse_diabetic_before)
print('RMSE before linear regression - Diabetes:', rmse_diabetic_before)
# Obesity
mse_obese_before = mean_squared_error(y_test_obesity, y_pred_obesity)
rmse_obese_before = np.sqrt(mse_obese_before)
print('MSE before linear regression - Obesity:', mse_obese_before)
print('RMSE before linear regression - Obesity:', rmse_obese_before)
```

```
# Inactivity
mse_inactive_before = mean_squared_error(y_test_inactive, y_pred_inactive)
rmse_inactive_before = np.sqrt(mse_inactive_before)
print('MSE before linear regression - Inactivity:', mse_inactive_before)
print('RMSE before linear regression - Inactivity:', rmse_inactive_before)

# Select the independent variables (X) and dependent variable (y)
X_diabetic_vs_obese_inactive = df_merged[['% OBESE', '% INACTIVE']]
y_diabetic_vs_obese_inactive = df_merged['% DIABETIC']

# Split the data into training and testing sets
X_train_diabetic_vs_obese_inactive, X_test_diabetic_vs_obese_inactive,
y_train_diabetic_vs_obese_inactive, y_test_diabetic_vs_obese_inactive = train_test_split(
    X_diabetic_vs_obese_inactive, y_diabetic_vs_obese_inactive, test_size=0.2, random_state=0)

X_test_with_constant_diabetic_vs_obese_inactive =
sm.add_constant(X_test_diabetic_vs_obese_inactive)

# Create a linear regression model
model_diabetic_vs_obese_inactive = LinearRegression()

# Train the model
model_diabetic_vs_obese_inactive.fit(X_train_diabetic_vs_obese_inactive,
y_train_diabetic_vs_obese_inactive)

# Make predictions on the test set
y_pred_diabetic_vs_obese_inactive =
model_diabetic_vs_obese_inactive.predict(X_test_diabetic_vs_obese_inactive)
# Plot the regression plane for % DIABETIC vs % OBESE and % INACTIVE
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Plot the actual data points
ax.scatter(X_test_diabetic_vs_obese_inactive['% OBESE'],
        X_test_diabetic_vs_obese_inactive['% INACTIVE'],
        y_test_diabetic_vs_obese_inactive, c='black', marker='o', label='Actual')

# Plot the predicted values
ax.scatter(X_test_diabetic_vs_obese_inactive['% OBESE'],
        X_test_diabetic_vs_obese_inactive['% INACTIVE'],
        y_pred_diabetic_vs_obese_inactive, c='red', marker='s', label='Predicted')

# Create a meshgrid for the plane
grid_x, grid_y = np.meshgrid(X_test_diabetic_vs_obese_inactive['% OBESE'],
                        X_test_diabetic_vs_obese_inactive['% INACTIVE'])
grid_z = (model_diabetic_vs_obese_inactive.coef_[0] * grid_x +
```

```
        model_diabetic_vs_obese_inactive.coef_[1] * grid_y +
        model_diabetic_vs_obese_inactive.intercept_)

# Plot the regression plane
ax.plot_surface(grid_x, grid_y, grid_z, alpha=0.3, rstride=100, cstride=100, color='blue',
label='Regression Plane')
ax.set_xlabel('% OBESE')
ax.set_ylabel('% INACTIVE')
ax.set_zlabel('% DIABETIC')
ax.set_title('Multiple Linear Regression: % DIABETIC vs % OBESE and % INACTIVE')

plt.legend()
plt.show()

# Evaluate the model
mse_diabetic_vs_obese_inactive =
metrics.mean_squared_error(y_test_diabetic_vs_obese_inactive,
y_pred_diabetic_vs_obese_inactive)
mae_diabetic_vs_obese_inactive =
metrics.mean_absolute_error(y_test_diabetic_vs_obese_inactive,
y_pred_diabetic_vs_obese_inactive)
rmse_diabetic_vs_obese_inactive =
metrics.mean_squared_error(y_test_diabetic_vs_obese_inactive,
y_pred_diabetic_vs_obese_inactive, squared=False)

# Calculate kurtosis and skewness
kurtosis_diabetic_vs_obese_inactive = kurtosis(residuals_diabetic_vs_obese_inactive)
skewness_diabetic_vs_obese_inactive = skew(residuals_diabetic_vs_obese_inactive)

# Calculate median and standard deviation
median_diabetic_vs_obese_inactive = np.median(residuals_diabetic_vs_obese_inactive)
std_dev_diabetic_vs_obese_inactive = np.std(residuals_diabetic_vs_obese_inactive)

# Print the results
print('Mean Square Error (DIABETIC vs OBESE and INACTIVE):',
mse_diabetic_vs_obese_inactive)
print('Mean Absolute Error (DIABETIC vs OBESE and INACTIVE):',
mae_diabetic_vs_obese_inactive)
print('Root Mean Squared Error (DIABETIC vs OBESE and INACTIVE):',
rmse_diabetic_vs_obese_inactive)
print('Kurtosis (DIABETIC vs OBESE and INACTIVE):', kurtosis_diabetic_vs_obese_inactive)
print('Skewness (DIABETIC vs OBESE and INACTIVE):',
skewness_diabetic_vs_obese_inactive)
print('Median (DIABETIC vs OBESE and INACTIVE):', median_diabetic_vs_obese_inactive)
print('Standard Deviation (DIABETIC vs OBESE and INACTIVE):',
std_dev_diabetic_vs_obese_inactive)
```

```
# Create a linear regression model
model_multiple_regression = LinearRegression()

# Train the model
model_multiple_regression.fit(X_train_diabetic_vs_obese_inactive,
y_train_diabetic_vs_obese_inactive)

# Add a constant to the predictor variables (required for statsmodels)
X_with_constant_multiple_regression = sm.add_constant(X_train_diabetic_vs_obese_inactive)

# Create an OLS (Ordinary Least Squares) model
ols_model_multiple_regression = sm.OLS(y_train_diabetic_vs_obese_inactive,
X_with_constant_multiple_regression)

# Fit the OLS model
ols_results_multiple_regression = ols_model_multiple_regression.fit()

# Print the summary which contains p-values
print(ols_results_multiple_regression.summary())

# Select the independent variable (X) and dependent variable (y)
X_obese = df_merged[['% OBESE']]
y_diabetic = df_merged['% DIABETIC']

# Split the data into training and testing sets
X_train_obese, X_test_obese, y_train_diabetic, y_test_diabetic = train_test_split(
    X_obese, y_diabetic, test_size=0.2, random_state=0)

# Create a linear regression model
model_obese_vs_diabetic = LinearRegression()

# Train the model
model_obese_vs_diabetic.fit(X_train_obese, y_train_diabetic)

# Make predictions on the test set
y_pred_diabetic = model_obese_vs_diabetic.predict(X_test_obese)

# Print the coefficients
print('Coefficients:', model_obese_vs_diabetic.coef_)
print('Intercept:', model_obese_vs_diabetic.intercept_)

# Evaluate the model
mse = mean_squared_error(y_test_diabetic, y_pred_diabetic)
print('Mean Squared Error:', mse)
```

```
# Plot the regression line
plt.scatter(X_test_obese, y_test_diabetic, color='black', label='Actual')
plt.plot(X_test_obese, y_pred_diabetic, color='blue', linewidth=3, label='Regression Line')
plt.xlabel('% OBESE')
plt.ylabel('% DIABETIC')
plt.title('Linear Regression: % OBESE vs % DIABETIC')
plt.legend()
plt.show()

# Display the summary with p-values
X_with_constant_obese = sm.add_constant(X_train_obese)
ols_model_obese_vs_diabetic = sm.OLS(y_train_diabetic, X_with_constant_obese)
ols_results_obese_vs_diabetic = ols_model_obese_vs_diabetic.fit()
print(ols_results_obese_vs_diabetic.summary())

# Select the independent variable (X) and dependent variable (y)
X_inactive = df_merged[['% INACTIVE']]
y_diabetic = df_merged['% DIABETIC']

# Split the data into training and testing sets
X_train_inactive, X_test_inactive, y_train_diabetic, y_test_diabetic = train_test_split(
    X_inactive, y_diabetic, test_size=0.2, random_state=0)

# Create a linear regression model
model_inactive_vs_diabetic = LinearRegression()

# Train the model
model_inactive_vs_diabetic.fit(X_train_inactive, y_train_diabetic)

# Make predictions on the test set
y_pred_diabetic = model_inactive_vs_diabetic.predict(X_test_inactive)

# Print the coefficients
print('Coefficients:', model_inactive_vs_diabetic.coef_)
print('Intercept:', model_inactive_vs_diabetic.intercept_)

# Evaluate the model
mse = mean_squared_error(y_test_diabetic, y_pred_diabetic)
print('Mean Squared Error:', mse)

# Plot the regression line
plt.scatter(X_test_inactive, y_test_diabetic, color='black', label='Actual')
plt.plot(X_test_inactive, y_pred_diabetic, color='blue', linewidth=3, label='Regression Line')
plt.xlabel('% INACTIVE')
plt.ylabel('% DIABETIC')
plt.title('Linear Regression: % INACTIVE vs % DIABETIC')
```

```
plt.legend()
plt.show()

# Display the summary with p-values
X_with_constant_inactive = sm.add_constant(X_train_inactive)
ols_model_inactive_vs_diabetic = sm.OLS(y_train_diabetic, X_with_constant_inactive)
ols_results_inactive_vs_diabetic = ols_model_inactive_vs_diabetic.fit()
print(ols_results_inactive_vs_diabetic.summary())
# Calculate kurtosis after linear regression
residuals_diabetic = y_test_diabetic - y_pred_diabetic
residuals_obesity = y_test_obesity - y_pred_obesity
residuals_inactive = y_test_inactive - y_pred_inactive

kurtosis_diabetic_after = kurtosis(residuals_diabetic)
kurtosis_obesity_after = kurtosis(residuals_obesity)
kurtosis_inactive_after = kurtosis(residuals_inactive)

print('Kurtosis after linear regression - Diabetic:', kurtosis_diabetic_after)
print('Kurtosis after linear regression - Obesity:', kurtosis_obesity_after)
print('Kurtosis after linear regression - Inactive:', kurtosis_inactive_after)

# Calculate skewness after linear regression

skew_diabetic_after = skew(residuals_diabetic)
print('Skewness after linear regression - Diabetic:', skew_diabetic_after)
skew_obese_after = skew(residuals_obese)
print('Skewness after linear regression - Obesity:', skew_obese_after)
skew_inactive_after = skew(residuals_inactive)
print('Skewness after linear regression - Inactivity:', skew_inactive_after)

# Diabetes
residuals_diabetic = y_test_diabetic_vs_obese_inactive - y_pred_diabetic_vs_obese_inactive
mse_diabetic_after = mean_squared_error(y_test_diabetic_vs_obese_inactive,
y_pred_diabetic_vs_obese_inactive)
rmse_diabetic_after = np.sqrt(mse_diabetic_after)
print('MSE after linear regression - Diabetes:', mse_diabetic_after)
print('RMSE after linear regression - Diabetes:', rmse_diabetic_after)

# Obesity
residuals_obese = y_test_obese - y_pred_obese
mse_obese_after = mean_squared_error(y_test_obese, y_pred_obese)
rmse_obese_after = np.sqrt(mse_obese_after)
print('MSE after linear regression - Obesity:', mse_obese_after)
print('RMSE after linear regression - Obesity:', rmse_obese_after)

# Inactivity
```

```
residuals_inactive = y_test_inactive - y_pred_inactive
mse_inactive_after = mean_squared_error(y_test_inactive, y_pred_inactive)
rmse_inactive_after = np.sqrt(mse_inactive_after)
print('MSE after linear regression - Inactivity:', mse_inactive_after)
print('RMSE after linear regression - Inactivity:', rmse_inactive_after)
# Extract the % DIABETIC data from your DataFrame
diabetic_data = df_merged['% DIABETIC']

# Calculate mean and standard deviation
mean = diabetic_data.mean()
std_dev = diabetic_data.std()

# Create a histogram to visualize the distribution
plt.hist(diabetic_data, bins=30, density=True, alpha=0.6, color='g')

# Create a range of x values
x = np.linspace(mean - 3 * std_dev, mean + 3 * std_dev, 100)
# Calculate the corresponding probability density function (PDF)
pdf = stats.norm.pdf(x, mean, std_dev)

# Plot the PDF as a smooth curve
plt.plot(x, pdf, 'k-', linewidth=2)

# Add labels and a title
plt.xlabel('% DIABETIC')
plt.ylabel('Probability Density')
plt.title('Normal Distribution of % DIABETIC')

# Show the plot
plt.show()
# Extract the % OBESE data from your DataFrame
obese_data = df_merged['% OBESE']

# Calculate mean and standard deviation
mean = obese_data.mean()
std_dev = obese_data.std()

# Create a histogram to visualize the distribution
plt.hist(obese_data, bins=30, density=True, alpha=0.6, color='b')

# Create a range of x values
x = np.linspace(mean - 3 * std_dev, mean + 3 * std_dev, 100)
# Calculate the corresponding probability density function (PDF)
pdf = stats.norm.pdf(x, mean, std_dev)

# Plot the PDF as a smooth curve
```

```
plt.plot(x, pdf, 'k-', linewidth=2)

# Add labels and a title
plt.xlabel('% OBESE')
plt.ylabel('Probability Density')
plt.title('Normal Distribution of % OBESE')

# Show the plot
plt.show()

# Extract the % INACTIVE data from your DataFrame
inactive_data = df_merged['% INACTIVE']

# Calculate mean and standard deviation
mean = inactive_data.mean()
std_dev = inactive_data.std()

# Create a histogram to visualize the distribution
plt.hist(inactive_data, bins=30, density=True, alpha=0.6, color='r')

# Create a range of x values
x = np.linspace(mean - 3 * std_dev, mean + 3 * std_dev, 100)
# Calculate the corresponding probability density function (PDF)
pdf = stats.norm.pdf(x, mean, std_dev)

# Plot the PDF as a smooth curve
plt.plot(x, pdf, 'k-', linewidth=2)

# Add labels and a title
plt.xlabel('% INACTIVE')
plt.ylabel('Probability Density')
plt.title('Normal Distribution of % INACTIVE')

# Show the plot
plt.show()
#correlational analysis
# Scatter plot: % DIABETIC vs % OBESE
plt.scatter(df_merged['% OBESE'], y_diabetic_vs_obese_inactive, color='blue')
plt.title('% DIABETIC vs % OBESE')
plt.xlabel('% OBESE')
plt.ylabel('% DIABETIC')
plt.show()

# Scatter plot: % DIABETIC vs % INACTIVE
plt.scatter(df_merged['% INACTIVE'], y_diabetic_vs_obese_inactive, color='green')
plt.title('% DIABETIC vs % INACTIVE')
```

```
plt.xlabel('% INACTIVE')
plt.ylabel('% DIABETIC')
plt.show()

import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Assuming df_merged contains the DataFrame with columns '% DIABETIC', '% OBESE', '%
INACTIVE'
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
# Scatter plot
ax.scatter(df_merged['% OBESE'], df_merged['% INACTIVE'], df_merged['% DIABETIC'])
# Set labels
ax.set_xlabel('% OBESE')
ax.set_ylabel('% INACTIVE')
ax.set_zlabel('% DIABETIC')
# Set title
ax.set_title('3D Scatter Plot of % DIABETIC, % OBESE, and % INACTIVE')
plt.show()
```

8 Reference:

1. Reference: “An Introduction to Statistical Learning with Applications in Python”, Chapter 4.
2. Applied Logistic Regression, Hosmer & Lemeshow.