Machine Learning Analysis of Diabetes, Obesity, & Inactivity: U.S. Counties via CDC Data

Co-Authors:

Sakthi Swarup Vasanadu Kasi - 02085794

Shalabh Singh Yadav – 02130778

Teja Naga- 02122966

The issues:

Diabetes is swiftly becoming a major health concern in the United States, having a deeper look into related health factors such as obesity and inactivity. Our goal is not just to identify this correlation, but also to understand how they interact and help to develop effective, data-driven health strategies. Understanding the relationship between diabetes, obesity, and inactivity will not only help healthcare professionals create more targeted intervention plans but also assist healthcare providers in forming strategies that address the divorced needs of different groups and areas. Eventually, our approach aims to combine thorough data analysis with strategic planning to tackle the growing issue of diabetes and related health challenges.

We address the following questions:

- What is the primary factor that leads to diabetes?
- Can we correlate diabetes with inactivity and obesity?
- How much effect do these variables have on diabetics?

Findings:

Data from the Centers for Disease Control and Prevention (CDC) have been thoroughly analyzed, and significant links between obesity, inactivity, and diabetes have been found in a range of counties. The combined effects of obesity and inactivity are strongly correlated with diabetes, as indicated by a correlation value of 73%. When we analyze further, our study separates this relation into two halves, showing that diabetics have a correlation with obesity of 38% and a correlation with inactivity of 56%, which is significantly higher. This suggests that a county with a higher percentage of inactivity is likely to also have a larger percentage of diabetics, providing that inactivity has a stronger influence on diabetes prevalence than obesity. These results support the critical necessity to address physical inactivity as a major risk factor in diabetic occurrences, thereby demonstrating patterns for future public health policies.

Discussions:

In analyzing the obtained dataset related to diabetes, notable inconsistency emerges due to inconsistent data points across various counties: 363 for obesity, 1370 for inactivity, and 3142 for diabetes. Consequently, to assure a thorough analysis, our evaluation focuses on the 354 data points that exhibit consistency across all parameters. The model, constructed to predict diabetes using inactivity and obesity as independent variables, displayed moderate accuracy, and was

refined using the weighted least square model (WLS) to reduce heteroscedasticity. Although refinements were made, substantial improvement remained difficult to track down, underlining the need for further research to refine the model and incorporate additional influencing factors.
Analyzing on average %diabetics based on county data projects certain trends, Although, it is necessary to acknowledge that the complexity of variables influencing diabetes extends beyond inactivity and obesity. Moving forward, incorporating additional variables, and addressing data gaps may offer enhanced predictive accuracy and deeper insights into the versatile nature of diabetes.

Appendix A: Method

We got our data from the CDC website, which we downloaded as an Excel sheet. This data shows the percentage of people with diabetes, the percentage of people who are inactive, and the percentage of people who are obese in different counties across the United States. We will use this data to explore and find patterns related to diabetes in various U.S. areas.

Starting with Data cleaning which is an important step in any data analysis process, especially when dealing with complexity. In our analysis of data cleaning and visualization techniques for diabetes, we utilized various computer tools such as pandas and numpy to effectively handle our data, and matplotlib and seaborn to generate informative charts and plots.

We used some computer tools (pandas and numpy) to help us work with our data and others (matplotlib and seaborn) to produce graphs and plots. We loaded our data, identified unique state and county values, and merged them into a different sheet. Then, we generated box plots for demonstrating the percentages of diabetes, inactivity, and obesity to analysis the trend and outliers. We removed unnecessary columns before saving the cleaned-up data.

In the modelling stage, we used a prediction model which is linear regression model, to investigate how the percentages of inactivity and obesity (our independent values) would predict the percentage of diabetes (our dependent value) on 354 data points. While building our model 80% of the data was taken in the training set and remaining 20% were use for testing, with each sample having 2 features (%inactive and %obese).

We analyzed residuals, which are the inequalities between our model's predictions and what actually occurred, in order to determine how accurate our model was. We plotted a graph with residuals vs fitted values, after closely analyzing it, we discovered a problem called heteroscedasticity, which means that our model's errors were not evenly distributed, suggesting that we might need to make some changes to make our model predicted more accurately. To confirm heteroscedasticity, we plotted a scale-location plot (sqrt (Residuals) vs fitted values).

To fix this, we utilized Weighted Least Squares (WLS). This technique allows us to assign different weights to different data points, which is especially useful when our data has diverse

reliability or patterns. Through this approach, we ensured that our model did not handle all data points equally and prioritized points, resulting in a more efficient model.

We also wanted to experiment, ordinary least squares (OLS), which tries to reduce the model's overall errors in predicting diabetes based on obesity and inactivity. We looked at the R-squared score to determine how well our model predicted the outcomes.

When we compared the two methods, OLS and WLS, we looked at how each estimated relationships between our variables, how confident we were in our estimates, and how well the model predicted the data. The essential distinction is that, although OLS treats all data points equally, WLS priorities them based on reliability.

In the context of improving model efficiency, specifically in enhancing the R-squared value achieved from the Weighted Least Squares (WLS) model, performing k-fold cross-validation and experimenting with polynomial degree.

We used K-fold validation (both 5-fold and 10-fold). This involves splitting our data into K parts, training our model K times, and always using a different part for testing its accuracy, ensuring a thorough check. The average RMSE is similar for k=5 and k=10, indicating comparable predictive accuracy across different numbers of folds. The root mean square error (RMSE) was used as an essential tool to assess the accuracy of predictions. This method allowed for a methodical assessment of model performance under various data partitioning configurations for training and validation purposes. The goal was to determine whether alternative data segmentation levels (i.e., the number of folds) had any discernible effects on the model's RMSE-measured predicted accuracy. The investigation was therefore focused on determining the ideal 'k' that would enable a reliable yet computationally efficient model validation, while upholding a strict standard of forecast accuracy and reliability.

Then we used polynomial regression, for finding the right polynomial degree, which is like choosing the right level of complexity for our model, is key to making accurate predictions. A useful technique to help us do this. We take turns using one-fold to test the model and the rest to train it, going through each fold once. This helps us see how well models with different polynomial degrees perform by looking at an error metric, often the Mean Squared Error (MSE), which we want to minimize while keeping the model simple to avoid overfitting. Once we find the best degree, we use it to build our final model using all the data and can test it with any separate test data we have. It is also important to remember to scale or normalize our features, especially when dealing with polynomial features, to keep our model stable.

Appendix B: Results

This project analyses 3 different health parameters throughout various US counties and states: Diabetes, obesity, and inactivity. The percentage distributions of diabetic, obese, and inactive reveals a wide range of trends, with a positive skew for diabetes, negative skew for obesity and

inactivity, revealing disparities among different state and counties as shown in **Figure-1**.

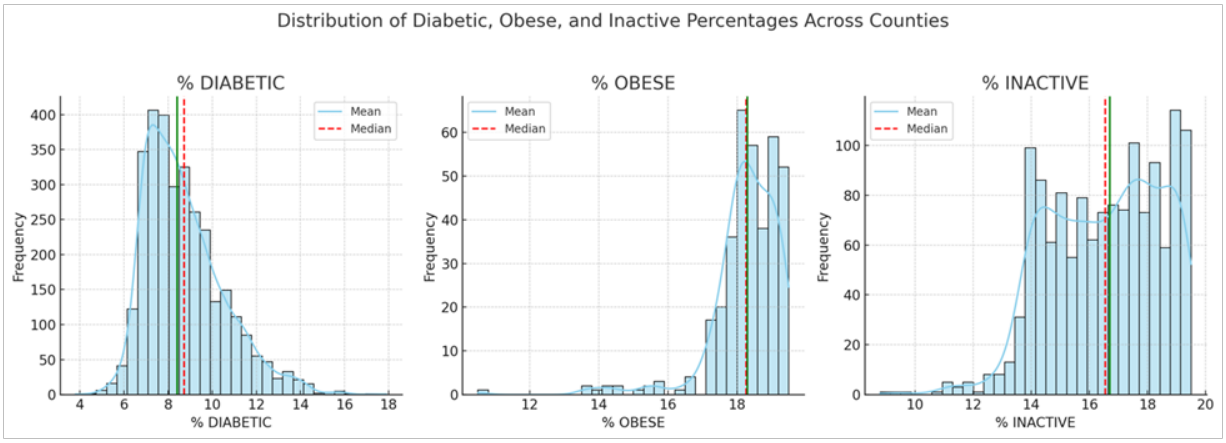


Fig1: Distribution of Diabetic, Obese and Inactive Percentage.

We developed a dataset of 354 data points which were analysed to develop the linear regression model after a comprehensive data cleaning process that addressed issues with missing values, outliers, and variable modifications. The created linear regression model, presented in the following Equation**: 1**, predicts "% DIABETIC" using the predictors "% OBESE" and "% INACTIVE".

$$\% \text{ DIABETIC} = 9.175 + 0.009 \text{ X } (\% \text{ OBESE}) - 0.016 \text{ X } (\% \text{ INACTIVE})$$

Equation:1 Prediction of Diabetes using Inactive and Diabetes.

As a result of the positive 0.009 coefficient, a higher "% OBESE" partially raises predicted "%DIABETIC". On the contrary, because of the negative -0.016 coefficient, an increased "% INACTIVE" significantly decreases predicted "% DIABETIC". However, both these variables have an impact on the prevalence of diabetes. The model's predictions for "% DIABETIC" are MSE - 0.400063 and R2 value of 0.394598. As a result, having our model accuracy of 39.4%, this model can predict % diabetes based on %inactivity as well as %obesity as shown in **Figure-2.**

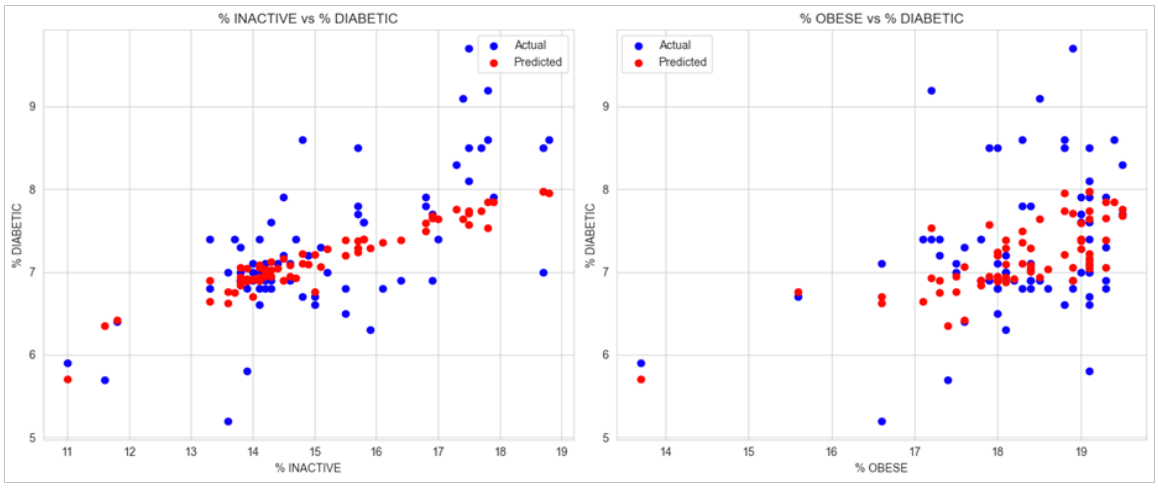


Fig2: Actual vs Predicted Diabetes rate.

To confirm the model's assumptions, residual plots were analysed, disclosing some patterns and heteroscedasticity. We examined the scale location plot and observed that there is heteroscedasticity since there isn't a constant variance as shown in **Figure-3.**

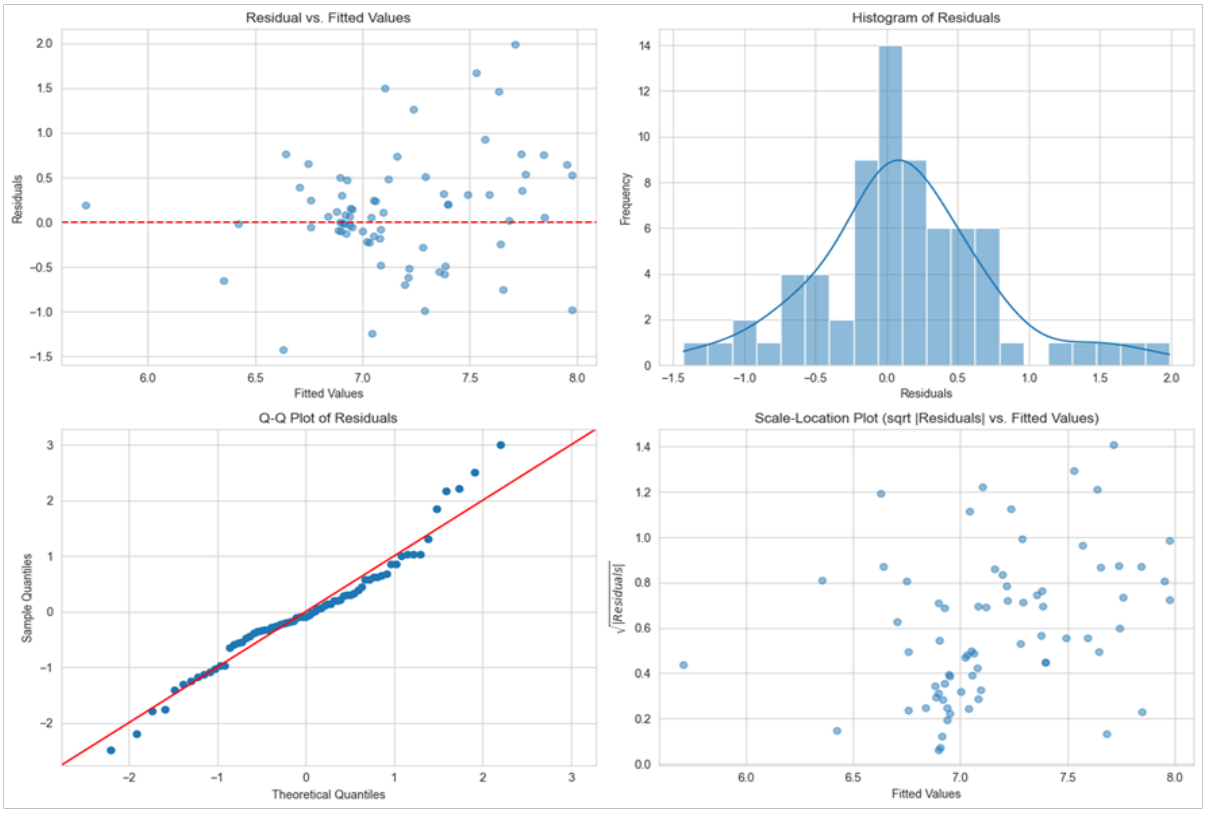


Fig3: Residual Plots

To address the issue of heteroscedasticity, Comparative analysis between OLS and WLS models was conducted. Comparing it to the OLS model, the WLS model has a significantly larger R-squared value (0.737) as shown in Figure-4, indicating that it explains a larger portion of the variability in the dependent variable in relation to the independent variables.

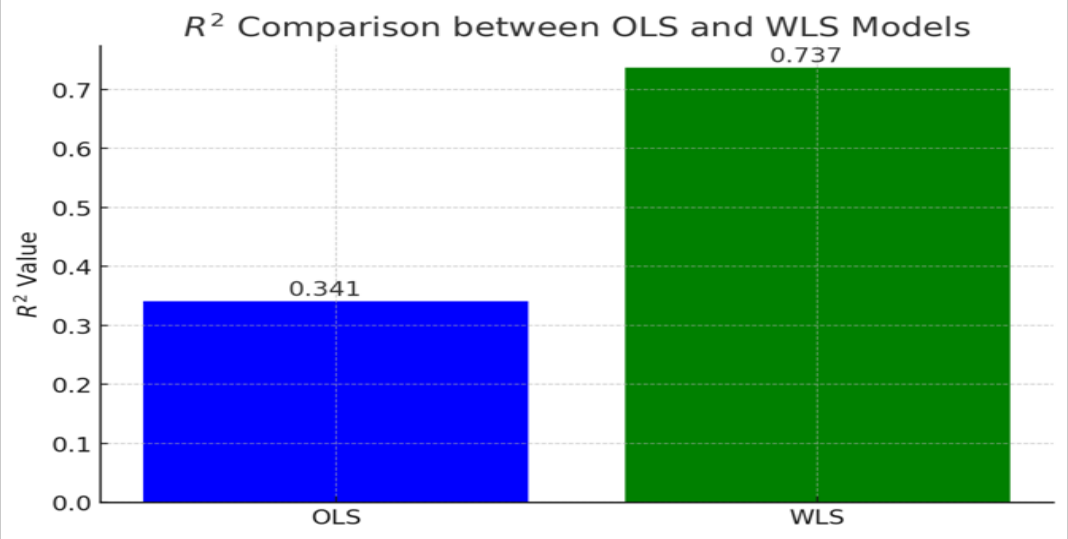


Figure4: R-Square value comparison.

While experimenting and aim to increase our model efficiency i.e., R square value we performed k fold cross validation and polynomial regression model. Employing k fold validation for 5 and 10 folds we discovered that the average RMSE shows equivalent accuracy in prediction for k=5 and k=10, despite the variation in folds. For k=5 and k=10, respectively, the average R square value demonstrates that the model explains for about 32.3% as well as 27.6% of the variation in the dependent variable, as the R-square values does not have much different for 5-fold and 10-fold does not have any discernible effects on the model RMSE therefore we considered.

Implementing 5-fold cross-validation to the same study for polynomial regression models (degrees 1 through 4). For each polynomial degree, we calculated the average RMSE and R-square values over the five folds. The observations are displayed in Figure -5.

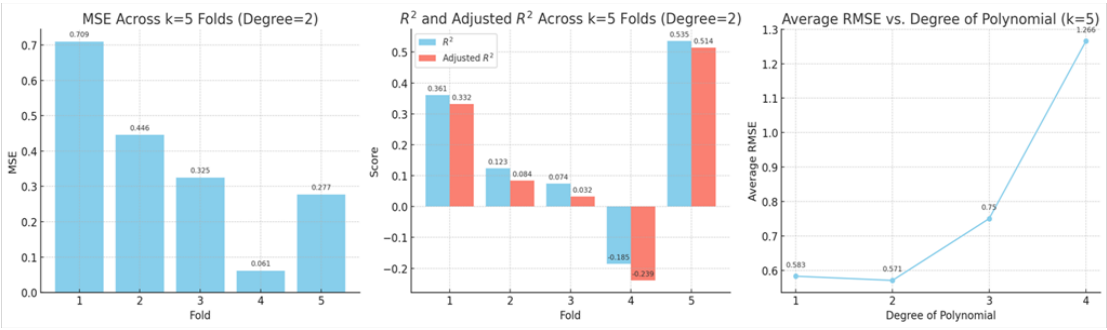


Figure 5: Results for Polynomial Regression Model.

Appendix C: Code

Importing the necessary libraries

```
[243]
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Reading the excel file and converting it into a dataframe

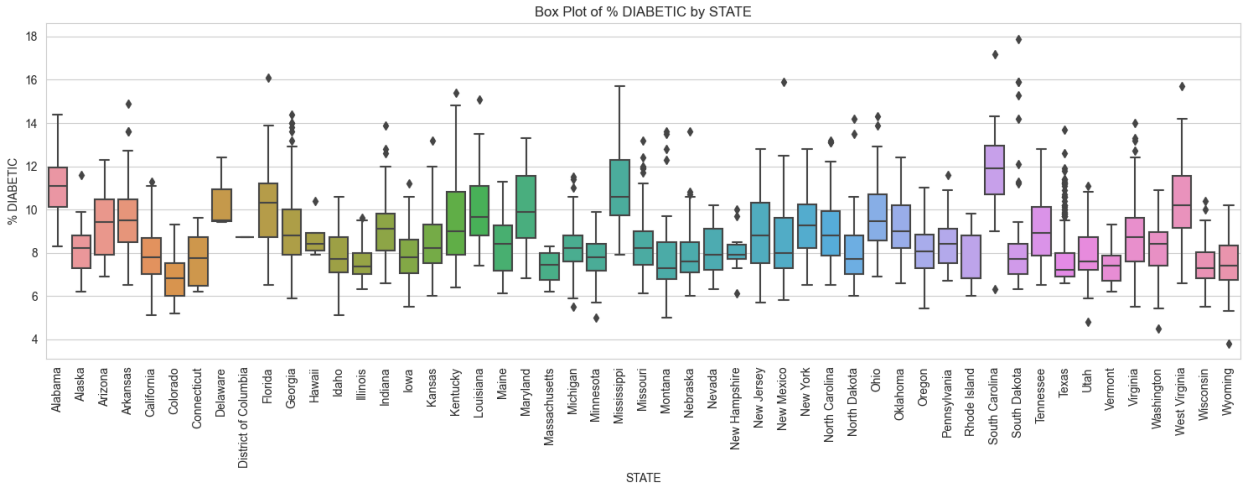
```
[244]
file_path = "./data.xlsx"
data = pd.read_excel(file_path)
data_diab=data= pd.read_excel(file_path, sheet_name="Diabetes")
data_obes=data= pd.read_excel(file_path, sheet_name="Obesity")
data_inact=data= pd.read_excel(file_path, sheet_name="Inactivity")
```

```
[249]
# Basic Descriptive Statistics for Diabetes
diabetes_stats=data_diab['% DIABETIC'].describe()
diabetes_stats['variance'] = data_diab['% DIABETIC'].var()
diabetes_stats['skewness'] = data_diab['% DIABETIC'].skew()
diabetes_stats['kurtosis'] = data_diab['% DIABETIC'].kurt()
print(diabetes_stats)
```

```
count       3142.000000
mean           8.719796
std            1.794854
min            3.800000
25%            7.300000
50%            8.400000
75%            9.700000
max           17.900000
variance       3.221499
skewness       0.974915
kurtosis       1.035291
Name: % DIABETIC, dtype: float64
```

Box Plot % diabetic vs State- Outlier

```
[253]
plt.figure(figsize=(15, 6))
sns.boxplot(x=data_diab['STATE'], y=data_diab['% DIABETIC'])
plt.xticks(rotation=90)
plt.title('Box Plot of % DIABETIC by STATE')
plt.xlabel('STATE')
plt.ylabel('% DIABETIC')
plt.tight_layout()
plt.show()
```

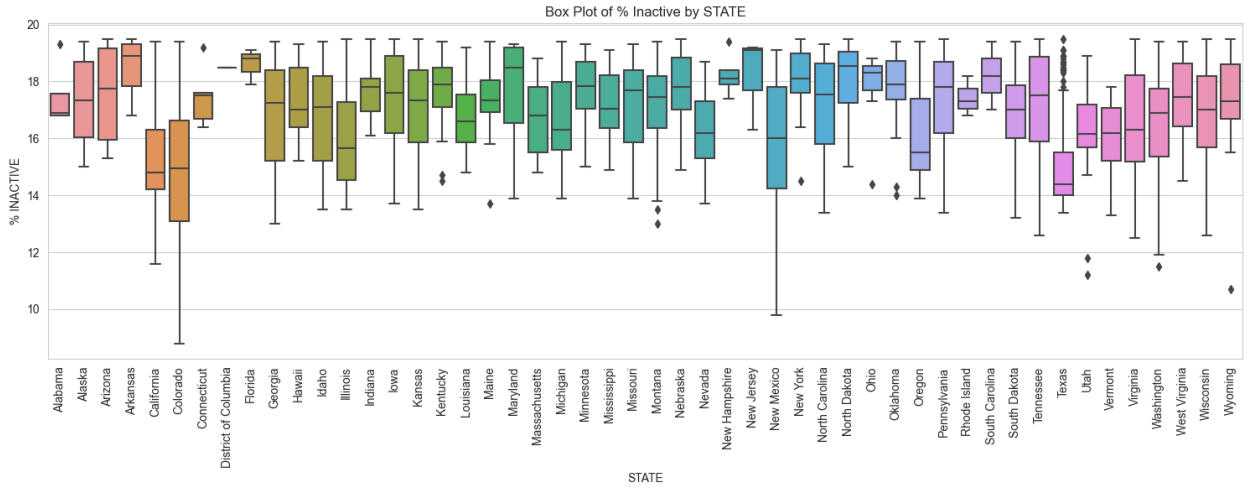


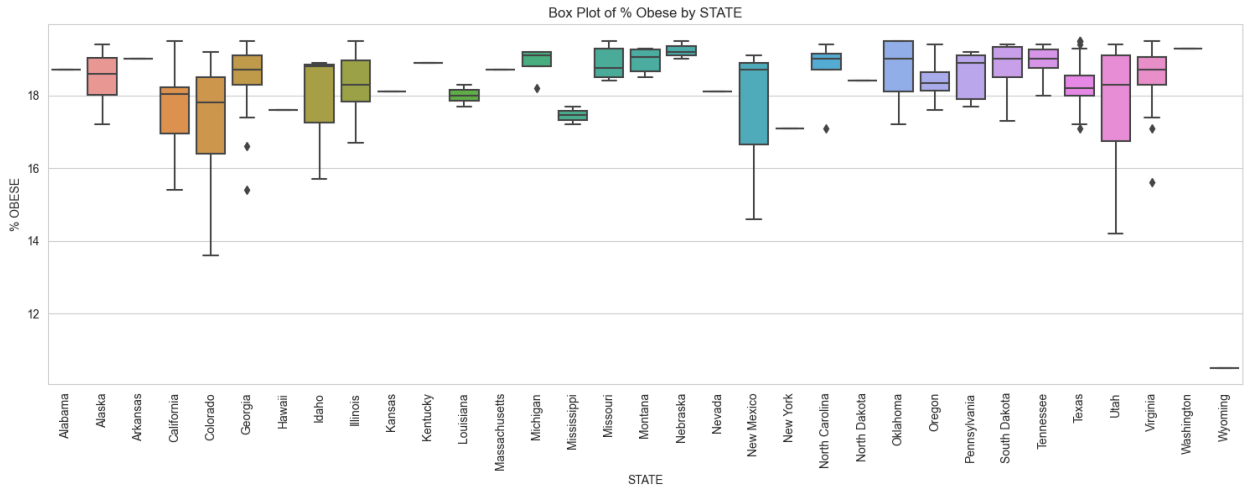

```
[263]
# Basic Descriptive Statistics for Inactivity
inactivity_stats=data_inact['% INACTIVE'].describe()
inactivity_stats['variance'] = data_inact['% INACTIVE'].var()
inactivity_stats['skewness'] = data_inact['% INACTIVE'].skew()
inactivity_stats['kurtosis'] = data_inact['% INACTIVE'].kurt()
print(inactivity_stats)
```

```
...  count       1370.000000
     mean          16.543358
     std            1.926010
     min            8.800000
     25%           15.000000
     50%           16.700000
     75%           18.100000
     max           19.500000
     variance       3.709514
     skewness      -0.342417
     kurtosis      -0.546649
     Name: % INACTIVE, dtype: float64
```

```
[264]
# Basic Descriptive Statistics for Obesity
obesity_stats=data_obes['% OBESE'].describe()
obesity_stats['variance'] = data_obes['% OBESE'].var()
obesity_stats['skewness'] = data_obes['% OBESE'].skew()
obesity_stats['kurtosis'] = data_obes['% OBESE'].kurt()
print(obesity_stats)
```

```
...  count       363.000000
     mean         18.264738
     std           1.038311
     min          10.500000
     25%          17.900000
     50%          18.300000
     75%          19.000000
     max          19.500000
     variance      1.078090
     skewness     -2.696210
     kurtosis     12.510652
     Name: % OBESE, dtype: float64
```


Box Plot of % Inactive by STATE
% INACTIVE
20
18
16
14
12
10
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
STATE


Box Plot of % Obese by STATE
% OBESE
18
16
14
12
Alabama
Alaska
Arkansas
California
Colorado
Georgia
Hawaii
Idaho
Illinois
Kansas
Kentucky
Louisiana
Massachusetts
Michigan
Mississippi
Missouri
Montana
Nebraska
Nevada
New Mexico
New York
North Carolina
North Dakota
Oklahoma
Oregon
Pennsylvania
South Dakota
Tennessee
Texas
Utah
Virginia
Washington
Wyoming
STATE


Corelation
# Calculate the correlation matrix
correlation_matrix = final_df[['% DIABETIC', '% OBESE', '% INACTIVE']].corr()
(variable) correlation_diabetic_obese: Scalar
'% OBESE' and '% INACTIVE'
correlation_diabetic_obese = correlation_matrix.loc['% DIABETIC', '% OBESE']
correlation_diabetic_inactive = correlation_matrix.loc['% DIABETIC', '% INACTIVE']
correlation_diabetic_obese, correlation_diabetic_inactive, correlation_matrix
[170]
0.0s
Python
(0.38994103875812935,
0.5671037350564135,
% DIABETIC % OBESE % INACTIVE
% DIABETIC 1.000000 0.389941 0.567104
% OBESE 0.389941 1.000000 0.472656
% INACTIVE 0.567104 0.472656 1.000000)

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import numpy as np
```

[286]

```
# Separate the dependent and independent variables
X = final_df[['% INACTIVE', '% OBESE']]
y = final_df['% DIABETIC']
X.head(), y.head()
```

[287]

```
(   % INACTIVE  % OBESE
 0        17.0     18.7
 1        16.2     18.9
 2        15.0     19.4
 3        17.8     17.2
 4        15.8     18.3,
 0    9.4
 1    6.8
 2    7.3
 3    9.2
 4    6.6
Name: % DIABETIC, dtype: float64)
```

```
# Initialize the Linear Regression model
linear_model = LinearRegression()

# Train the model on the training data
linear_model.fit(X_train, y_train)

# Make predictions on the test data
y_pred = linear_model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Coefficients and intercept
coefficients = linear_model.coef_
intercept = linear_model.intercept_

mse, r2, coefficients, intercept
```

[289]

```
(0.40006315354054606,
 0.3946987907298565,
 array([0.19406655, 0.1429442 ]),
 1.6154740957564178)
```

WLS vs OLS

```
import statsmodels.api as sm

# Prepare the data
X = final_df[['% INACTIVE', '% OBESE']]
y = final_df['% DIABETIC']

# Add a constant to the predictor variables matrix
X = sm.add_constant(X)

# Step 1: Fit an OLS model
ols_model = sm.OLS(y, X).fit()

# Step 2: Calculate residuals
residuals = ols_model.resid

# Step 3: Calculate weights as the inverse of the absolute residuals
weights = 1 / abs(residuals)

# Step 4: Fit a WLS model
wls_model = sm.WLS(y, X, weights=weights).fit()

# Display the summary of both models
ols_summary = ols_model.summary()
wls_summary = wls_model.summary()

ols_summary, wls_summary
```

93]

```
(<class 'statsmodels.iolib.summary.Summary'>
 """
                            OLS Regression Results
 ==============================================================================
 Dep. Variable:             % DIABETIC   R-squared:                       0.341
 Model:                            OLS   Adj. R-squared:                  0.337
 Method:                 Least Squares   F-statistic:                     90.71
 Date:                Fri, 06 Oct 2023   Prob (F-statistic):           1.76e-32
 Time:                        16:37:35   Log-Likelihood:                -315.89
 No. Observations:                 354   AIC:                             637.8
 Df Residuals:                     351   BIC:                             649.4
 Df Model:                           2
 Covariance Type:            nonrobust
 ==============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
 ------------------------------------------------------------------------------
 const          1.6536      0.562      2.941      0.003       0.548       2.759
 % INACTIVE     0.2325      0.023     10.023      0.000       0.187       0.278
 % OBESE        0.1111      0.035      3.192      0.002       0.043       0.180
 ==============================================================================
 Omnibus:                       17.281   Durbin-Watson:                   1.673
 Prob(Omnibus):                  0.000   Jarque-Bera (JB):               45.622
 Skew:                          -0.042   Prob(JB):                     1.24e-10
 Kurtosis:                       4.757   Cond. No.                         421.
 ==============================================================================
...
 ==============================================================================

 Notes:
 [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
 """)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
wls_summary
```

WLS Regression Results

Dep. Variable:	% DIABETIC	R-squared:	0.737
Model:	WLS	Adj. R-squared:	0.736
Method:	Least Squares	F-statistic:	492.4
Date:	Fri, 06 Oct 2023	Prob (F-statistic):	1.35e-102
Time:	16:37:35	Log-Likelihood:	-61.826
No. Observations:	354	AIC:	129.7
Df Residuals:	351	BIC:	141.3
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1.8309	0.192	9.544	0.000	1.454	2.208
% INACTIVE	0.2298	0.012	19.959	0.000	0.207	0.252
% OBESE	0.1033	0.013	8.042	0.000	0.078	0.129

Omnibus:	31.404	Durbin-Watson:	1.718
Prob(Omnibus):	0.000	Jarque-Bera (JB):	10.412
Skew:	0.043	Prob(JB):	0.00548
Kurtosis:	2.164	Cond. No.	503.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

```python
# Perform k-fold CV for k=5 and k=10
def perform_kfold_cv(k, X, y, weights):
    kf = KFold(n_splits=k, shuffle=True, random_state=1)
    mse_list = []
    for train_index, test_index in kf.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        model = sm.WLS(y_train, X_train, weights=weights.iloc[train_index]).fit()
        predictions = model.predict(X_test)
        mse = mean_squared_error(y_test, predictions)
        mse_list.append(mse)
    return mse_list

mse_k5 = perform_kfold_cv(5, X, y, weights)
mse_k10 = perform_kfold_cv(10, X, y, weights)

# Function to calculate RMSE and R2 for each fold
def calculate_rmse_r2(mse_list, k, X, y, weights):
    kf = KFold(n_splits=k, shuffle=True, random_state=1)
    rmse_list = np.sqrt(mse_list)
    r2_list = []
    for train_index, test_index in kf.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]
        model = sm.WLS(y_train, X_train, weights=weights.iloc[train_index]).fit()
        predictions = model.predict(X_test)
        r2 = r2_score(y_test, predictions)
        r2_list.append(r2)
    return rmse_list, r2_list
# Calculating RMSE and R2 for k=5 and k=10
rmse_k5, r2_k5 = calculate_rmse_r2(mse_k5, 5, X, y, weights)
rmse_k10, r2_k10 = calculate_rmse_r2(mse_k10, 10, X, y, weights)
# Calculating average RMSE and R2 for k=5 and k=10
avg_rmse_k5, avg_rmse_k10 = np.mean(rmse_k5), np.mean(rmse_k10)
avg_r2_k5, avg_r2_k10 = np.mean(r2_k5), np.mean(r2_k10)
```

```
print(avg_rmse_k5, avg_r2_k5, avg_rmse_k10, avg_r2_k10)
```

```
0.5882357352380638 0.32297976782880156 0.5853731007694261 0.27571241919572803
```

Polynomial Degree

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import cross_val_score

# Function to create polynomial regression model
def polynomial_regression(degree):
    return make_pipeline(PolynomialFeatures(degree), LinearRegression())
# Predictor and response variables
X_poly = final_df[['% INACTIVE', '% OBESE']]
y_poly = final_df['% DIABETIC']
# Define the degrees for the polynomial
degrees = [1, 2, 3, 4]
# Store the average RMSE and R2 for each polynomial degree
avg_rmse_per_degree = []
avg_r2_per_degree = []

# Perform 10-fold CV for polynomial regression of degrees 1 through 4
for degree in degrees:
    model = polynomial_regression(degree)
    mse_scores = -cross_val_score(model, X_poly, y_poly, cv=5, scoring='neg_mean_squared_error')
    r2_scores = cross_val_score(model, X_poly, y_poly, cv=5, scoring='r2')

    avg_rmse_per_degree.append(np.mean(np.sqrt(mse_scores)))
    avg_r2_per_degree.append(np.mean(r2_scores))

# Results
list(zip(degrees, avg_rmse_per_degree, avg_r2_per_degree))
```

```
[(1, 0.5830814784052398, 0.15480999642745846),
 (2, 0.5708151968001751, 0.18169056467242567),
 (3, 0.7503669432319742, -0.4264743403306305),
 (4, 1.265691563257321, -3.5696038733466553)]
```

Contribution:

The project was clearly a collective effort, wherein every team member equally contributed at all stages - from brainstorming to plotting graphs, from coding to drafting the report. Every phase witnessed an equal infusion of input and expertise from each member, ensuring a balanced and thorough collaborative endeavor.