Diabetes, obesity, and inactivity: A multi-linear regression of three interrelated health factors.

A Punchline Report

Issues:

We have sourced our healthcare data from the Centers for Disease Control and Prevention (CDC) through their Diabetes Atlas of Social Determinants of Health (SDOH) platform, accessible at the following link: https://gis.cdc.gov/grasp/diabetes/diabetesatlas-sdoh.html. This wonderful resource provides an in-depth examination of diabetes prevalence and its relationship to many social determinants of health. In our analysis, we focus on the interaction between diabetes and major factors like obesity and inactivity, on their deep relationships and public health implications. We hope to contribute to a better knowledge of these health issues using linear regression models and to inform evidence-based actions to promote the well-being of impacted individuals and communities by harnessing this data.

Based on the information gathered, we answer the following queries or issues.

- To improve the accuracy of our model, how can we effectively deal with the problem of skewed data with significant kurtosis in the columns?
- How can we improve the effectiveness of our model in this project by effectively utilizing the correlation function to determine which predictor variables ('YEAR', 'FIPS', 'COUNTY', 'STATE', '% OBESE', '% INACTIVE') are significantly connected with the target variable ('% DIABETIC')?
- How to measure the combined or joint effects of two or more independent variables on the dependent variable and how does the relationship between one independent variable and the dependent variable change?

Findings

We tried Log, Power, SQRT, and BOX-COX transformation techniques to handle the OBESE column which was left skewed, and the Kurtosis value of 15.9, by using Box-Cox transformation we got the Kurtosis to 2.6 which is near to the ideal kurtosis value of 3.

Predictors ('% OBESE', '% INACTIVE') - As the Predictor variable and the target variable are Quantitative, we have used the correlation function, they have correlation values of 0.38% and 0.56% respectively, which can be used for modeling.
Predictors ('YEAR', 'FIPS') - we can remove the 'YEAR' and 'FIPS' columns as it has no effect on the target variable.
Predictors (COUNTY', 'STATE') - we have used the ANOVA test for the two categorical variables in the '% DIABETIC' column, which gave the P-value for COUNTY is 0.95 which is NOT correlated with % DIABETIC and the P-value for STATE is 8.27e-32 which correlated with % DIABETIC.so we have removed the COUNTY column and created PD dummies for the STATE Column.

We will be using the Interaction term concept to capture the combined effect of two independent variables ('% OBESE', '% INACTIVE') on the Target variables using the interaction term.
result_df['Interaction_Term'] = result_df['% OBESE'] * result_df['% INACTIVE']

Appendix A: Methods

First, I discovered that the data was clean and that there wasn't much to interact with it. However, because the lengths of the three data frames varied, we had to merge them using the primary key "FIPS", once we merged that we had 354 data sets to play with. Even with those data, there were numerous unnecessary pieces of information for computation, such as category data, so we put everything in a dictionary and deleted the rest.

Secondly, the obesity was right skewed which can affect the total model building or the accuracy of the model so we tried to transform the data using a statistical method called the Box-Cox transformation can give non-normal dependent variables a normal shape.

The linear regression for these models, where the target value is Diabetic in each model, still had some backlogs after we built the model using diabetes and obesity, diabetes, and inactivity. Therefore, we applied multi-linear regression to this, using diabetes as the goal value and obesity and inactivity as the predictor variables.

Our primary objective is to increase the effectiveness of the overall model. To do this, we tried to gather information about the initial portion of the model using scatter plots, values for R squared, and mean squared errors, and then we tried to interact with categorical variables by determining the relationship between them. Only one of them, the state, showed a correlation, thus we tried to link it to numerical variables using the Dummies approach. There were huge improvements in the model.

The model's efficiency was then significantly increased by including the interaction terms (x*y), or obesity and inactivity, which had a significant positive impact. Additionally, we attempted to enhance the model by using K-fold cross-validation, a technique used to evaluate machine learning models by resampling data.

Appendix B: Results

From the data set of 354 data points containing Diabetics, Obesity, and Inactivity

On applying descriptive statistics on all three data points we find the mean, median, standard deviation, skewness, and Kurtosis for all.

In the next part, we find the Correlation between all three variables, to make a call on deciding which all variables we need for computing the model

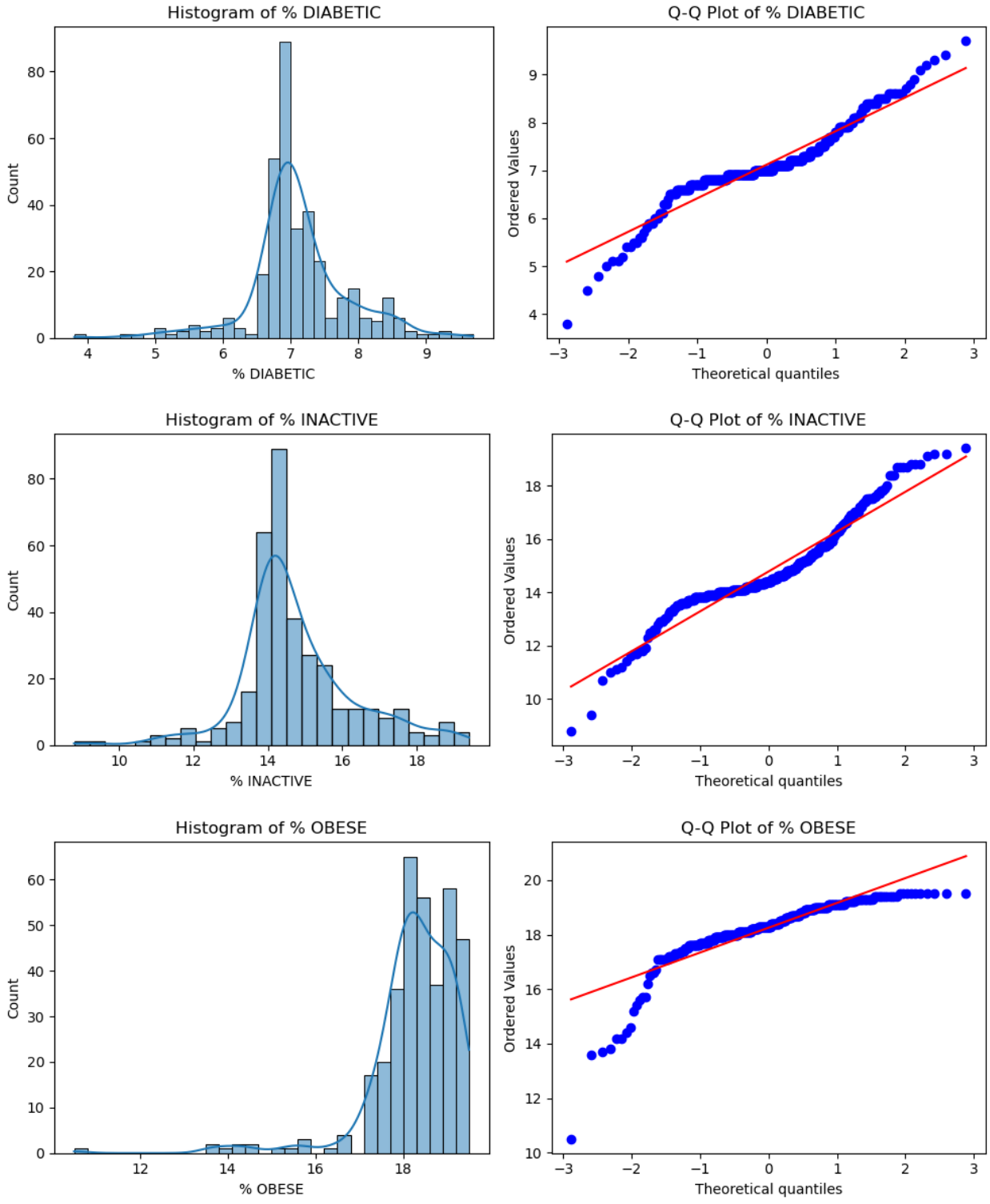
```
print("Kurtosis = ",round(stats.kurtosis(result_df['% OBESE'],fisher=False),5))
```

```
Descriptive Statistics
% DIABETIC
Median =  7.0
Mean =  7.11582
Stdev =  0.72741
Skewness =  -0.04881
Kurtosis =  5.78842
% INACTIVE
Median =  14.4
Mean =  14.77627
Stdev =  1.54236
Skewness =  0.42571
Kurtosis =  4.6136
% OBESE
Median =  18.3
Mean =  18.25254
Stdev =  1.02803
Skewness =  -2.75189
Kurtosis =  15.93152
```

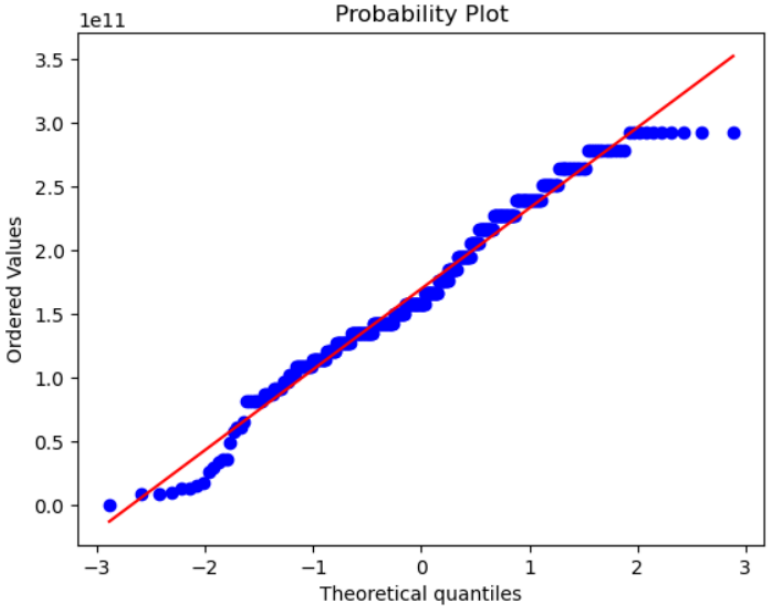
```
result_df.iloc[:,-3:].corr()
```

	% DIABETIC	% OBESE	% INACTIVE
% DIABETIC	1.000000	0.389941	0.567104
% OBESE	0.389941	1.000000	0.472656
% INACTIVE	0.567104	0.472656	1.000000

After this, we plotted the Histogram and the Q-Q plot distribution which shows the distribution of the data

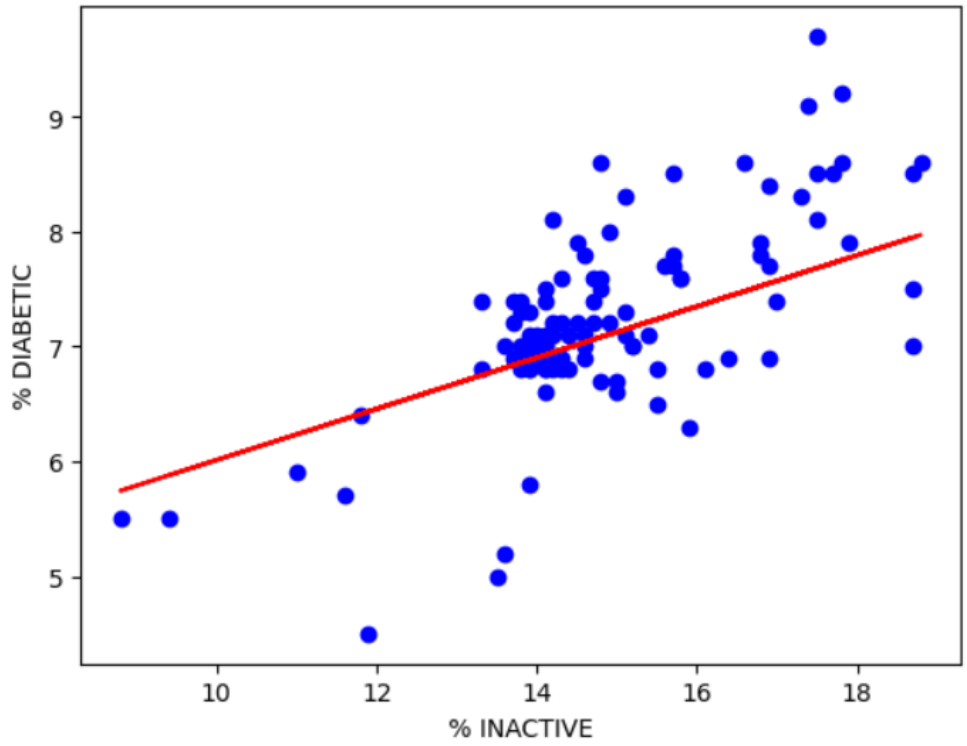


The Obesity from the above graph is left skewed so we tried to normalize the data using the Box-Cox method the below graph shows the probability graph after normalization

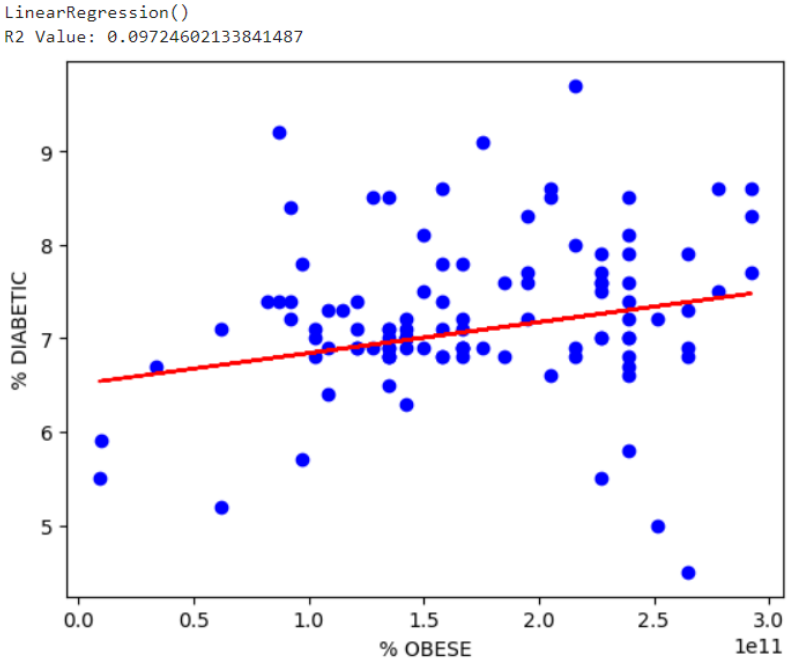


Scatter plot for the Linear regression model of Inactivity vs Diabetic

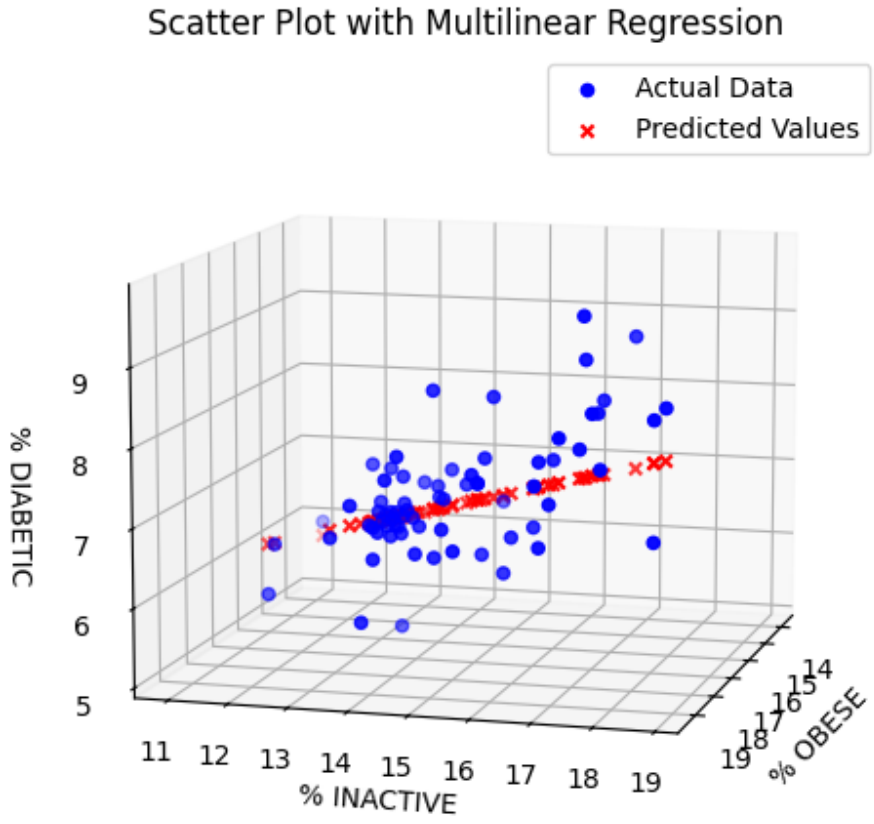
```
LinearRegression()
R2 Value: 0.2322780423451284
```



Scatter plot for the Linear regression model of Obesity vs Diabetic



Scatter plot for the Multi-linear regression model of Inactivity, Obesity vs Diabetic



The final result of our model after 5 k-fold cross-validation

```
Lagrange multiplier statistic: 89.14332976218978
Lagrange multiplier p-value:   2.116511912730936e-06
F-statistic:                   3.0579967042900766
P-value of F-statistic:        1.0238970068375836e-07
```

Concluded results

```
LinearRegression()
R2 Value: 0.6088867828206832

Accuracy values for 9-fold Cross Validation:
 [92.3975073  90.59163795 94.97238697 91.08393082 94.26022422 98.2084383
 97.40664731 98.15251719 90.51641866]

Final Average Accuracy of the model: 94.18
```

In this study, our findings suggest that inactivity emerges as a more significant indicator for predicting diabetes compared to obesity, as evidenced by the correlation matrix analysis. Following the training of the linear model using this dataset, we determined that our model exhibited a commendable R2 value of 0.688 and an accuracy of 94.18.

Appendix C: Python Code

Implementation of the Box-Cox concept for normalizing the data

```
#BOX COX
print("Before Box-Cox, Kurtosis value = ",round(stats.kurtosis(result_df['% OBESE'],fisher=False),5))
result_df['% OBESE'],parameters=stats.boxcox(result_df['% OBESE'])
print("After Box-Cox, Kurtosis value = ",round(stats.kurtosis(result_df['% OBESE'],fisher=False),5))
print(parameters)
stats.probplot(result_df['% OBESE'], dist="norm", plot=plt)
```

Implementation of ANOVA function for all categorical columns

```
def FunctionAnova(inpData, TargetVariable, CategoricalPredictorList):
    # Creating an empty list of final selected predictors
    SelectedPredictors=[]

    for predictor in CategoricalPredictorList:

CategoryGroupLists=inpData.groupby(predictor)[TargetVariable].apply(list)
        AnovaResults = f_oneway(*CategoryGroupLists)
        # If the ANOVA P-Value is <0.05, that means we reject H0
        if (AnovaResults[1] < 0.05):
                print(predictor, 'is correlated with', TargetVariable, '| P-Value:', AnovaResults[1])
            SelectedPredictors.append(predictor)
        else:
             print(predictor, 'is NOT correlated with', TargetVariable, '| P-Value:', AnovaResults[1])

    return(SelectedPredictors)

# Calling the function to check which categorical variables are correlated with target
CategoricalPredictorList=['FIPS', 'COUNTY', 'STATE']
FunctionAnova(inpData=result_df,        TargetVariable='%        DIABETIC', CategoricalPredictorList=CategoricalPredictorList)

result_df.drop(['FIPS','COUNTY'],axis=1, inplace=True)
result_df.head(5)
```

Implementing the Multi-linear Regression

```
#MLR

merged_df2=result_df[['% INACTIVE','% DIABETIC','% OBESE']]
# Separate Target Variable and Predictor Variables
TargetVariable='% DIABETIC'
Predictors=['% OBESE', '% INACTIVE']

X=merged_df2[Predictors].values
y=merged_df2[TargetVariable].values

# Split the data into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

###### Linear Regression in Python #######
RegModel = LinearRegression()

#Printing all the parameters of Linear regression
print(RegModel)

#Creating the model on Training Data
LREG=RegModel.fit(X_train,y_train)
prediction=LREG.predict(X_test)

#Measuring Goodness of fit in Training data
print('R2 Value:',metrics.r2_score(y_train, LREG.predict(X_train)))
```

Implementation of Multi linear regression using interaction terms and Cross-validation

```
#MLR with state, interaction term with Cross-Validation

TargetVariable='% DIABETIC'
Predictors=['%              OBESE',              '%              INACTIVE',
'STATE_Alabama'......'Interaction_Term']

X=merged_df2[Predictors].values
y=merged_df2[TargetVariable].values

####### K-fold cross-validation #######
# Defining a custom function to calculate accuracy
# Make sure there are no zeros in the Target variable if you are using MAPE
def Accuracy_Score(orig,pred):
    MAPE = np.mean(100 * (np.abs(orig-pred)/orig))
    return(100-MAPE)

# Custom Scoring MAPE calculation
custom_Scoring=make_scorer(Accuracy_Score, greater_is_better=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=428)

###### Linear Regression in Python #######
RegModel = LinearRegression()

#Creating the model on Training Data
LREG=RegModel.fit(X_train,y_train)
prediction=LREG.predict(X_test)

#Measuring Goodness of fit in Training data
print('R2 Value:',metrics.r2_score(y_train, LREG.predict(X_train)))

# Running 10-fold cross-validation on a given algorithm
Accuracy_Values=cross_val_score(RegModel,      X      ,      y,      cv=9,
scoring=custom_Scoring)
print('\nAccuracy values for 9-fold Cross Validation:\n',Accuracy_Values)
print('\nFinal      Average      Accuracy      of      the      model:',
round(Accuracy_Values.mean(),2))
```