# Obesity Trends in Children Ages 2-19

Ly Nguyen, Teresa Nguyen, Tracy Nguyen
CIC-PCUBED 2023 | Faculty Mentor: Dr. Doina Bein

# Steps:

1. Learned **Python**, **Machine Learning** and **Panda** on Kaggle
2. Searched **cdc.gov** for dataset
3. Downloaded as CSV and save to Google Drive
4. Save to Google Drive and link drive to **Google Colab**
5. Reviewed dataset and **predict trends** to be observed
6. Preprocess data for data visualization and **model training**
7. Create charts, graphs, and models

# Goals:

## 01 Training

Split the data and train the machine

## 02 Model Fitting

Create model to estimate obesity percentages

## 03 Model Analysis

Observe trends in data

## 04 Prediction

Predict the rate of obesity for future years

# Loading Data

1. Uploaded files to Google Drive and mounted our Drive to Google CoLab

```
from google.colab import drive
drive.mount('/content/drive')
```

2. Imported the necessary libraries into one section

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import sklearn.datasets
from sklearn.model_selection import train_test_split
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

# Loading Data

Panda is used to read the data into the obesity_data

```
obesity_file_path = '../content/drive/MyDrive/Summer_Project/Obesity_among_children_and_adolescents_aged_2_19_years__by_

obesity_data = pd.read_csv(obesity_file_path)
obesity_data = obesity_data.drop(['INDICATOR','PANEL','PANEL_NUM','UNIT','UNIT_NUM','SE','FLAG'], axis = 'columns')
```

Start of data preprocessing

↳

| | STUB_NAME | STUB_NAME_NUM | STUB_LABEL_NUM | STUB_LABEL | YEAR | YEAR_NUM | AGE | AGE_NUM | ESTIMATE |
|---|---|---|---|---|---|---|---|---|---|
| 0 | Total | 0 | 0.0 | 2-19 years | 1988-1994 | 1 | 2-19 years | 0.0 | 10.0 |
| 1 | Total | 0 | 0.0 | 2-19 years | 1999-2002 | 2 | 2-19 years | 0.0 | 14.8 |
| 2 | Total | 0 | 0.0 | 2-19 years | 2001-2004 | 3 | 2-19 years | 0.0 | 16.3 |
| 3 | Total | 0 | 0.0 | 2-19 years | 2003-2006 | 4 | 2-19 years | 0.0 | 16.3 |
| 4 | Total | 0 | 0.0 | 2-19 years | 2005-2008 | 5 | 2-19 years | 0.0 | 16.2 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 835 | Percent of poverty level | 5 | 5.4 | 400% or more | 2007-2010 | 6 | 12-19 years | 0.3 | 14.0 |
| 836 | Percent of poverty level | 5 | 5.4 | 400% or more | 2009-2012 | 7 | 12-19 years | 0.3 | 13.8 |
| 837 | Percent of poverty level | 5 | 5.4 | 400% or more | 2011-2014 | 8 | 12-19 years | 0.3 | 13.7 |
| 838 | Percent of poverty level | 5 | 5.4 | 400% or more | 2013-2016 | 9 | 12-19 years | 0.3 | 13.7 |
| 839 | Percent of poverty level | 5 | 5.4 | 400% or more | 2015-2018 | 10 | 12-19 years | 0.3 | 11.0 |

# Data Preprocessing

Data is separated into groups

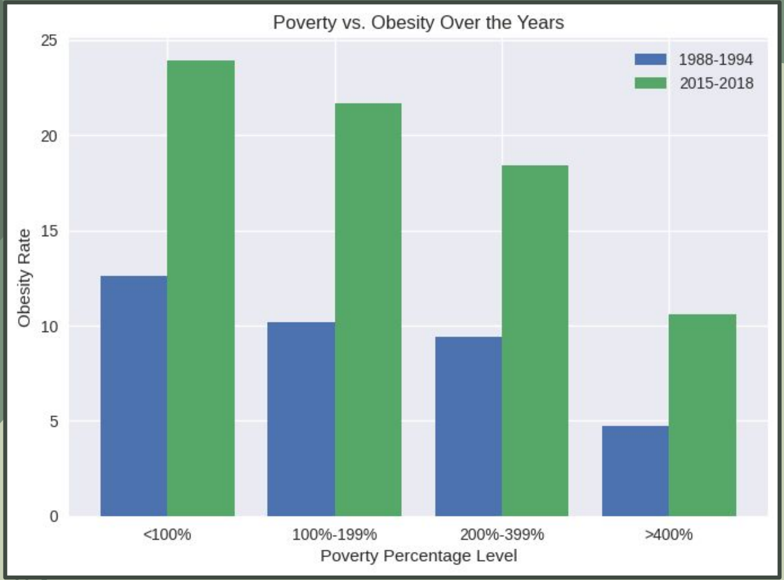
```
gender_data = obesity_data.loc[obesity_data['STUB_NAME_NUM'] == 1]
age_data = obesity_data.loc[obesity_data['STUB_NAME_NUM'] == 2]
race_origin_data = obesity_data.loc[obesity_data['STUB_NAME_NUM'] == 3]
race_origin_sex_data = obesity_data.loc[obesity_data['STUB_NAME_NUM'] == 4]
poverty_data = obesity_data.loc[obesity_data['STUB_NAME_NUM'] == 5]
```

Data is cleaned again and organized into smaller chunks

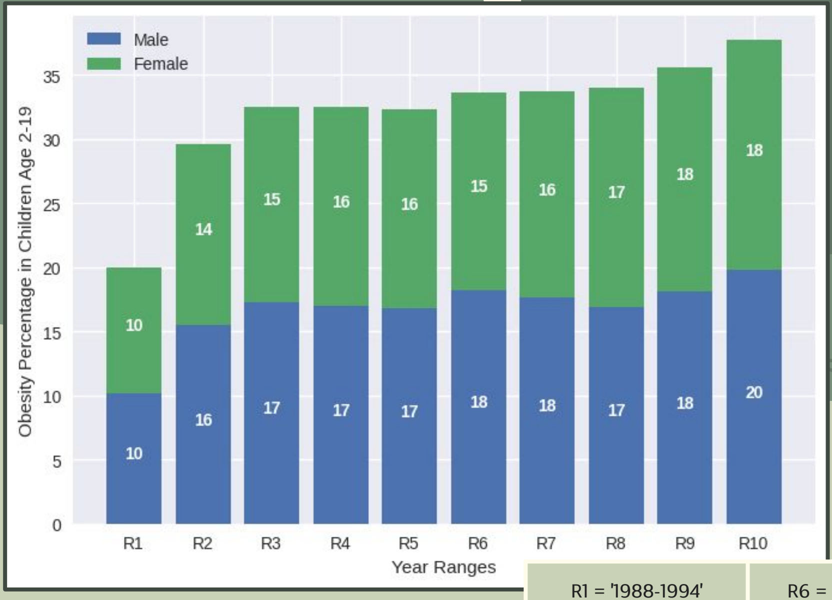
```
final_origin_data = race_origin_data.loc[ race_origin_data['AGE_NUM'] == 0.0].drop(['AGE','STUB_NAME','STUB_NAME_NUM','STUB_LABEL'], axis = 'columns')

white_only = final_origin_data.loc[final_origin_data['STUB_LABEL_NUM'] == 3.11]
black_and_african = final_origin_data.loc[final_origin_data['STUB_LABEL_NUM'] == 3.12]
asian_only = final_origin_data.loc[final_origin_data['STUB_LABEL_NUM'] == 3.13]
hispanic_all = final_origin_data.loc[final_origin_data['STUB_LABEL_NUM'] == 3.21]
mexican_only = final_origin_data.loc[final_origin_data['STUB_LABEL_NUM'] == 3.22]
```

# Bar Graphs

Twin Bar Graph



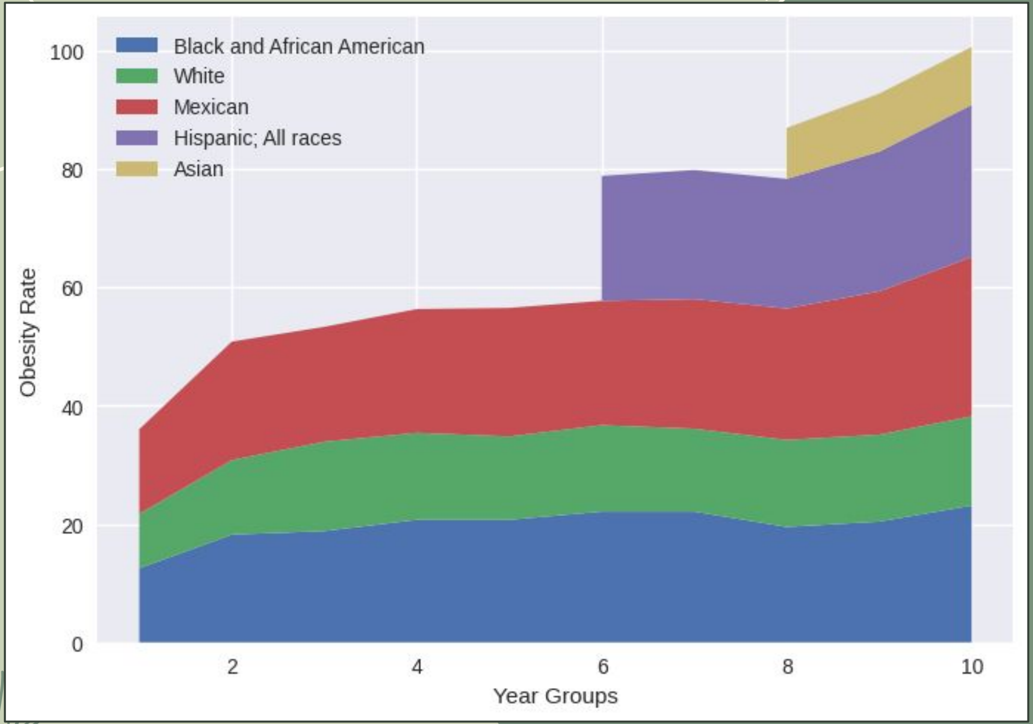

Stacked Bar Graph




| | |
|---|---|
| R1 = '1988-1994'<br>R2 = '1999-2002'<br>R3 = '2001-2004'<br>R4 = '2003-2006'<br>R5 = '2005-2008' | R6 = '2007-2010'<br>R7 = '2009-2012'<br>R8 = '2011-2014'<br>R9 = '2013-2016'<br>R10 = '2015-2018' |

Poverty level contributes to the obesity rate

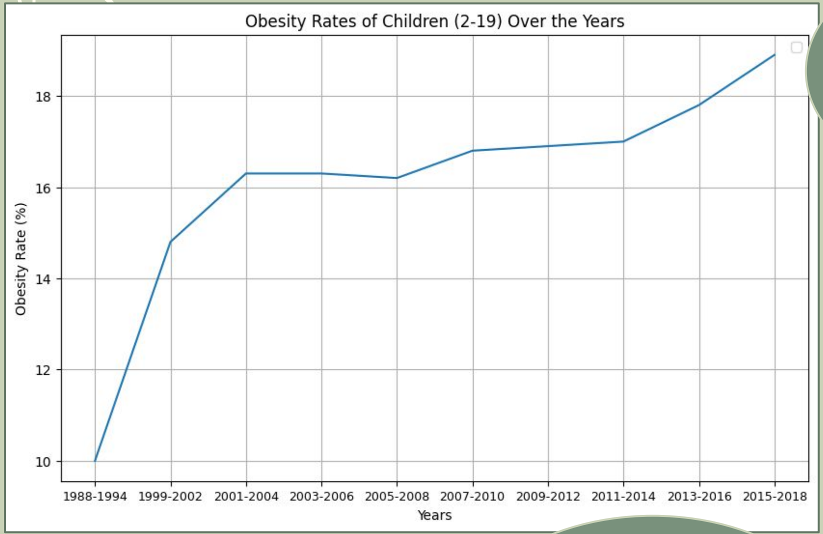
# Area Chart



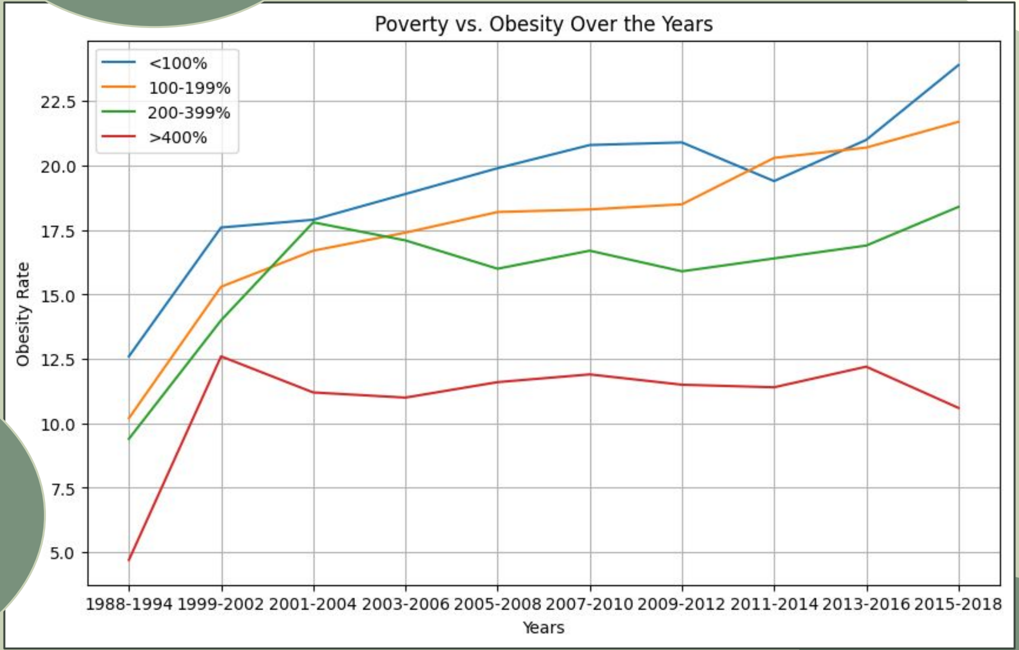

Obesity rates of children (2-19) over the years and ethnicities of the children

Hispanic, Mexican, and African American areas have highest obesity rates

# Line Graphs




-Higher poverty level = lower obesity rate
-Lower poverty level = higher obesity rate




Advancements of society, more resources, rise of junk food/fast food are some contributions to the increase of obesity in children

# Cleaning the Data

- used .loc[ ] to use only the entries of kids ages 2-19 that have poverty level data
- used .drop( ) to remove any column that I didn't need in my analysis

```
# SPLITTING/CLEANING DATA
my_data = obesity_data.loc[obesity_data.AGE_NUM == 0][obesity_data.STUB_NAME_NUM == 5]
my_data = my_data.drop(['AGE_NUM','STUB_NAME', 'STUB_NAME_NUM', 'STUB_LABEL', 'YEAR', 'AGE'], axis = 'columns')
```

| | STUB_LABEL_NUM | YEAR_NUM | ESTIMATE |
|---|---|---|---|
| 171 | 5.1 | 1 | 12.6 |
| 172 | 5.1 | 2 | 17.6 |
| 173 | 5.1 | 3 | 17.9 |
| 174 | 5.1 | 4 | 18.9 |
| 175 | 5.1 | 5 | 19.9 |
| 176 | 5.1 | 6 | 20.8 |
| 177 | 5.1 | 7 | 20.9 |
| 178 | 5.1 | 8 | 19.4 |
| 179 | 5.1 | 9 | 21.0 |
| 180 | 5.1 | 10 | 23.9 |
| 181 | 5.2 | 1 | 10.2 |
| 182 | 5.2 | 2 | 15.3 |
| 183 | 5.2 | 3 | 16.7 |
| 184 | 5.2 | 4 | 17.4 |
| 185 | 5.2 | 5 | 18.2 |
| 186 | 5.2 | 6 | 18.3 |
| 187 | 5.2 | 7 | 18.5 |
| 188 | 5.2 | 8 | 20.3 |
| 189 | 5.2 | 9 | 20.7 |
| 190 | 5.2 | 10 | 21.7 |
| 191 | 5.3 | 1 | 9.4 |
| 192 | 5.3 | 2 | 14.0 |
| 193 | 5.3 | 3 | 17.8 |
| 194 | 5.3 | 4 | 17.1 |
| 195 | 5.3 | 5 | 16.0 |
| 196 | 5.3 | 6 | 16.7 |
| 197 | 5.3 | 7 | 15.9 |
| 198 | 5.3 | 8 | 16.4 |
| 199 | 5.3 | 9 | 16.9 |
| 200 | 5.3 | 10 | 18.4 |
| 201 | 5.4 | 1 | 4.7 |
| 202 | 5.4 | 2 | 12.6 |
| 203 | 5.4 | 3 | 11.2 |
| 204 | 5.4 | 4 | 11.0 |
| 205 | 5.4 | 5 | 11.6 |
| 206 | 5.4 | 6 | 11.9 |
| 207 | 5.4 | 7 | 11.5 |
| 208 | 5.4 | 8 | 11.4 |
| 209 | 5.4 | 9 | 12.2 |
| 210 | 5.4 | 10 | 10.6 |

# Splitting the Dataset

```
predictors = ['STUB_LABEL_NUM', 'YEAR_NUM']
x = my_data[predictors]
y = my_data['ESTIMATE']
```

```
# SPLITTING THE DATASET -->
#     use 50% of the dataframe to train the machine and create a model
#     use 40% of data the machine has never seen before to validate the model
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.40,  random_state = 1)

# EXPLORING THE TRAIN AND TEST DATASETS
print("x_train shape: ", x_train.shape)
print("x_train mean: ", np.mean(x_train), "\n")

print("x_test shape: ", x_test.shape)
print("x_test mean: ", np.mean(x_test), "\n")

print("y_train shape: ", y_train.shape)
print("y_train mean: ", np.mean(y_test), "\n")
```

```
x_train shape:  (24, 2)
x_train mean:  STUB_LABEL_NUM    5.216667
YEAR_NUM          5.375000
dtype: float64

x_test shape:  (16, 2)
x_test mean:  STUB_LABEL_NUM    5.3000
YEAR_NUM          5.6875
dtype: float64

y_train shape:  (24,)
y_train mean:  14.9875
```

- with train_test_split( ), the data is split into 50% training data and 40% testing data

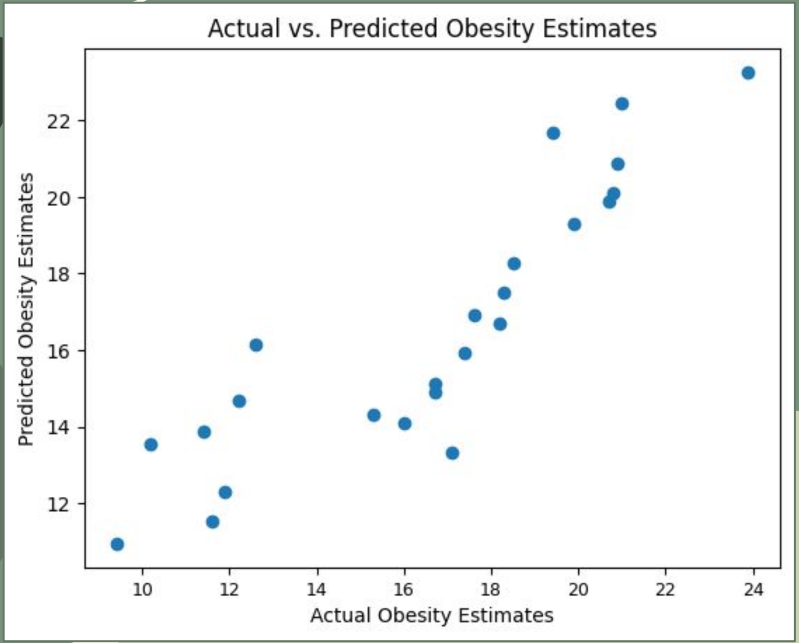
# Testing the Model

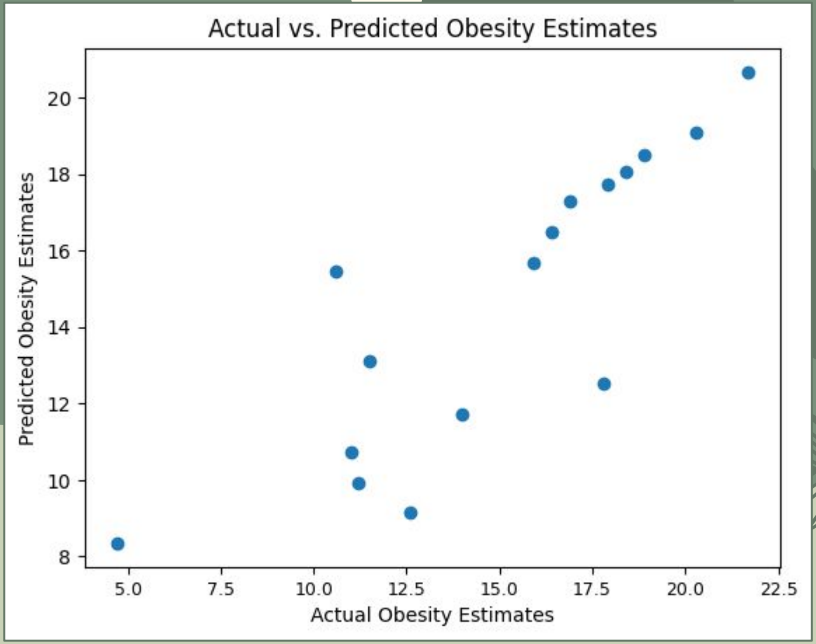
```python
# ACTUAL OBESITY ESTIMATES VS. PREDICTED OBESITY ESTIMATES (Y_TRAIN VS Y_PREDICT_TRAIN)
# USE X_TRAIN DATA TO PREDICT THE Y_TRAIN VALUES
y_predict_train = lr.predict(x_train)

plt.scatter(y_train, y_predict_train)
plt.xlabel("Actual Obesity Estimates")
plt.ylabel("Predicted Obesity Estimates")
plt.title("Actual vs. Predicted Obesity Estimates")
plt.show()

rscore1 = r2_score(y_train, y_predict_train)

# ACTUAL OBESITY ESTIMATES VS. PREDICTED OBESITY ESTIMATES (Y_TEST VS Y_PREDICT_TEST)
# USE X_TEST DATA TO PREDICT THE Y_TEST VALUES
y_predict_test = lr.predict(x_test)

plt.scatter(y_test, y_predict_test)
plt.xlabel("Actual Obesity Estimates")
plt.ylabel("Predicted Obesity Estimates")
plt.title("Actual vs. Predicted Obesity Estimates")
plt.show()

rscore2 = r2_score(y_test, y_predict_test)

print('Train Prediction Accuracy: ', rscore1)   # MORE ACCURATE
print('Test Prediction Accuracy: ', rscore2)
```

x_train is used to train the model

the model is then tested for accuracy

this process is repeated for the x_test values as well




```
Train Prediction Accuracy:  0.7832127497787615
Test Prediction Accuracy:  0.6997019443206081
```

Graph 1: Predicted obesity estimates using x_train values

Graph 2: Predicted obesity estimates using x_test values

Based on the calculated accuracy, using the predicted y_train values generate a more precise obesity estimate.

# Analyzing the Model

```python
# ANALYZING THE MODEL
ME = mean_absolute_error(y_test, y_predict_test)
print('Mean Absolute Error: ', ME)

MSE = mean_squared_error(y_test, y_predict_test, squared=False)
print('Mean Squared Error: ', MSE)

MSE = mean_squared_error(y_test, y_predict_test, squared=True)
print('Mean Squared Error (Squared): ', MSE)

RSQ = lr.score(x_test, y_test)
print('R-Squared: ', RSQ)
```

```
Mean Absolute Error:  1.6556547724927964
Mean Squared Error:  2.3620946113464423
Mean Squared Error (Squared):  5.5794909529519
R-Squared:  0.6997019443206081
```

| | |
|---|---|
| **Mean Absolute Error (ME)** | 1.7 |
| **Mean Squared Error (MSE)** | 2.4 |
| **MSE (Squared)** | 5.6 |
| **R-Squared (RSQ)** | 0.7 |

# Predicting Estimates for Future Years

## Creating New Values

To create new values, I use the YEAR_NUM from the x_train data and add 10 to calculate the next range of years

```
x_new = x_train

# GENERATE FUTURE YEARS
count = 0
for i in x_new.YEAR_NUM:
    x_new.YEAR_NUM.values[count] = i + 10
    count = count + 1
print(x_new)
```

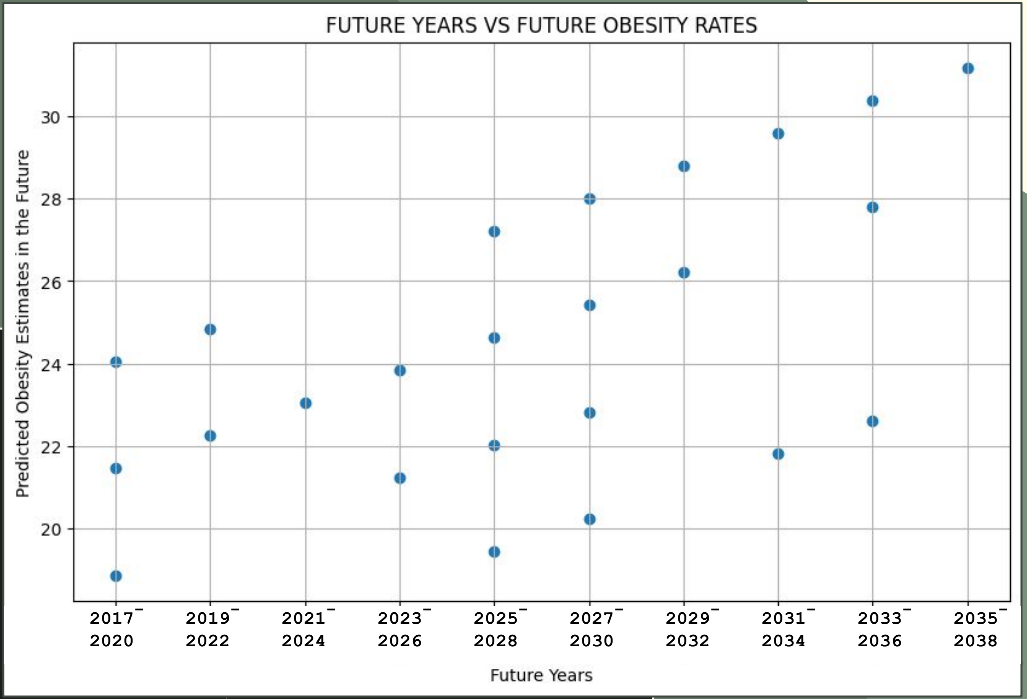
```
# 11 = '2017-2020'
# 12 = '2019-2022'
# 13 = '2021-2024'
# 14 = '2023-2026'
# 15 = '2025-2028'
# 16 = '2027-2030'
# 17 = '2029-2032'
# 18 = '2031-2034'
# 19 = '2033-2036'
# 20 = '2035-2038'
```

| STUB_LABEL_NUM | YEAR_NUM |
|---|---|
| 5.1 | 15 |
| 5.2 | 15 |
| 5.2 | 11 |
| 5.4 | 16 |
| 5.3 | 14 |
| 5.3 | 15 |
| 5.4 | 15 |
| 5.3 | 11 |
| 5.2 | 19 |
| 5.3 | 16 |
| 5.1 | 17 |
| 5.2 | 14 |
| 5.1 | 18 |
| 5.4 | 19 |
| 5.1 | 12 |
| 5.2 | 17 |
| 5.1 | 11 |
| 5.2 | 16 |
| 5.1 | 16 |
| 5.2 | 12 |
| 5.1 | 20 |
| 5.1 | 19 |
| 5.2 | 13 |
| 5.4 | 18 |

# Future Years vs. Obesity Rates

Predicted Obesity Rates for Years 2017-2038

```
# USE FUTURE YEARS (X_NEW) TO PREDICT FUTURE OBESITY RATES
y_predict_new = lr.predict(x_new)

def major_x_formatter(x, pos):
    if pos is not None:
        return f"{x_labels_list[pos]}"
    x_r = int(round(x))
    if x_r in x_labels:
        return f"{x:.0f}:{x_labels[x_r]}"
    else:
        return f"{x:.2f}"

# PLOT FUTURE YEARS VS FUTURE OBESITY RATES
plt.figure(figsize = (10, 6))

plt.scatter(x_new.YEAR_NUM, y_predict_new)
plt.xlabel("Future Years")
plt.ylabel("Predicted Obesity Estimates in the Future")
plt.title("FUTURE YEARS VS FUTURE OBESITY RATES")

x_labels1 = {11 : "2017-2020", 12 : "2019-2022", 13: "2021-2024" , 14: "2023-2026", 15: "2025-2028", 16 : "2027-2030"
x_labels_list1 = list(x_labels1.values())
plt.xticks(list(x_labels1.keys()))
```

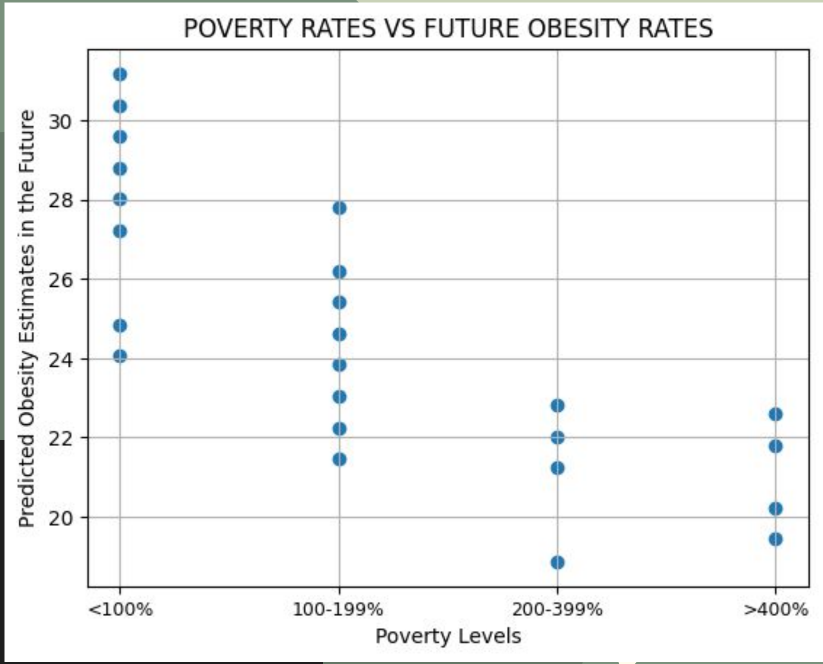



01

# Poverty Levels vs. Obesity Rates

Predicted Obesity Rates (2017-2038)
for each Poverty Level

```
# PLOT POVERTY RATES VS FUTURE OBESITY RATES

plt.scatter(x_new.STUB_LABEL_NUM, y_predict_new)
plt.xlabel("Poverty Levels")
plt.ylabel("Predicted Obesity Estimates in the Future")
plt.title("POVERTY RATES VS FUTURE OBESITY RATES")

x_labels = { 5.1 : "<100%", 5.2 : "100-199%", 5.3: "200-399%" , 5.4: ">400%" }
x_labels_list = list(x_labels.values())
plt.xticks(list(x_labels.keys()))
plt.gca().xaxis.set_major_formatter(major_x_formatter)

plt.grid()
plt.show()
```




02

# Thank You

CDC Obesity Dataset

Google Colab Notebook