Association of Demographic, Lifestyle, and Health Indicators with Risk of Heart Disease, Diabetes, and Stroke

Cindy Chen, JT Herren, Niha Suravarjjala

May 12th, 2024

1 Introduction

Our project seeks to understand the prevalence of heart disease, diabetes, and stroke in the United States and collect insights into risk behaviors and demographic factors that relate to these three diseases. We aim to summarize important factors that may affect the chance of having these three conditions so that we can minimize the risk of disease across the U.S. population.

The project's data is sourced from the Behavioral Risk Factor Surveillance System (BRFSS) from 2015. The BRFSS is an annual survey conducted by the Centers for Disease Control and Prevention (CDC) that asks about various existing or chronic health conditions, risk behaviors, and demographic data.

According to the CDC (2023), heart disease is the leading cause of death in the U.S., with one person dying from heart disease every 33 seconds ("Heart Disease", 2023). Diabetes is also extremely common in the U.S., where it is the eighth leading cause of death ("Diabetes", 2023). Additionally, nearly 800,000 people in the U.S. suffer from a stroke each year ("Stroke", 2023).

Fortunately, heart disease, type 2 diabetes, and stroke are all preventable. We want to understand what factors lead to these conditions so that we can suggest how to prevent heart disease, diabetes, and stroke. We believe the results of our project will be useful to a wide range of people due to the prevalence of these conditions in the United States and across the world.

2 Data Cleaning & Summaries

Our dataset is sourced from Kaggle and was previously cleaned to contain 253,680 survey responses and 22 features. We chose three of these features to be our response variables: heart disease, diabetes, and stroke. We then categorized the remaining explanatory features into three distinct groups including demographic variables (i.e. sex, age, and income), lifestyle (i.e. smokers, consumption of fruit, physical activity), and health indicators (i.e. blood pressure, cholesterol, and BMI) to better understand their contribution to the response.

Two of the three response variables were binary. After loading the dataset we replaced instances of the value 2 to 1 for the Diabetes variable to make it a binary variable like the other response variables. Originally, 2 equates to diabetes and 1 equates to pre-diabetes. By changing this category, the Diabetes variable now represents whether a person has signs of Diabetes to a varying extent or not. A summary of the columns available in our data set is included in Figure 1.

In addition to converting Diabetes to a binary variable, we created a new column combining the information from the 3 response variables into a string where there are 3 letters representing if

the given individual has one of the conditions. In order, we chose H for HeartDiseaseorAttack, D for Diabetes, and S for Stroke. If the letter in a position is O that means the individual does not have that condition. We also added a column in which the combined information of the response variables is encoded and stored as an ordinal variable (levels 1-8). Each level represents one of the combinations of the conditions.

Variables	Type	Levels
HeartDiseaseorAttack	Binary	2
Stroke	Binary	2
Diabetes	Binary	2
HighBP	Binary	2
HighChol	Binary	2
CholCheck	Binary	2
BMI	Continuous	N/A
Smoker	Binary	2
PhysActivity	Binary	2
Fruits	Binary	2
Veggies	Binary	2

Variables	Type	Levels
HvyAlcoholConsump	Binary	2
AnyHealthcare	Binary	2
NoDocbcCost	Binary	2
GenHlth	Ordinal	5
MentHlth	Ordinal	30
PhysHlth	Ordinal	30
DiffWalk	Binary	2
Sex	Binary	2
Age	Ordinal	13
Education	Ordinal	6
Income	Ordinal	6

Figure 1: Summary of variables, variable types, and number of levels.

3 Methodology

In our preliminary investigations, we created bar plots for each of the response variables against each of the predictor variables in which the X-axis contained the binary values 0 and 1 (0 signifying not having the condition and 1 the opposite). The Y-axis contained the value counts for each group within the predictor variables. This allowed us to gain a better initial understanding of the data by giving a visual representation of the distribution of each predictor variable in association with each response variable.

During the early exploration of the data, we attempted odds calculations between the response and predictor variables to assess the strength and direction of their relationships. We ultimately decided against using this method due to most of our calculated odds ratios being relatively close to 1. This suggests that the relationships between these variables and the predictor are not significant.

$$Odds = P(A)/(1 - P(A))$$

To identify which predictor variables are significantly related to the response variables, we conducted calculations of mutual information and conditional entropy. This involved calculating these values for each predictor against the encoded response variable. The results were then sorted by highest to lowest conditional entropy and visualized through a plot. The formulas for entropy, conditional entropy, and mutual information are listed respectively.

$$CE[X] = -\sum_{x} p(x) \log p(x)$$

$$CE[Y|X] = -\sum_{x} \frac{n_x}{N} CE[Y|X = x]$$

$$I[X, Y] = CE[X] - CE[X|Y]$$

Initially, this calculation was only done for individual predictor variables, but to explore the data further we also performed the calculations for pairs of predictor variables against the encoded response variable. The pairs were chosen based on the mutual information calculations for the individual predictors. There was a noticeable drop after the Income variable, which resulted in the

top 8 predictor variables being chosen for our paired analysis. We concatenated the categorical variables through a combination function, and then repeated the same calculation as we did for the single predictors. Calculating conditional entropy and mutual information for pairs of variables allowed us to identify interaction effects between predictor variables. This is important to discover if one predictor variable against the response variable is reliant on the value of another predictor, which can help identify more complex patterns within the data. Finally, we explored a few of our most important pairs of explanatory variables by plotting them against each other in a bar plot to visualize how they interact.

4 Results & Discussion

In our initial bar plots, we were able to explore how the distribution of Y changes with different X values. We display and discuss three notable examples here. First, we noticed those with high blood pressure had higher proportions of heart disease, diabetes, and stroke occurrence (Figure 2). Another example was general health (Figure 3). Although there was a small overall count of people who ranked their health as 5 (the worst on the scale), the proportion of that category that suffers from a combination of heart disease, diabetes, or stroke is higher than those who ranked their health as 1 (the best). Similarly, in the Age plot, we can see that the occurrence of our response variables is much more frequent in the older age groups (Figure 4).

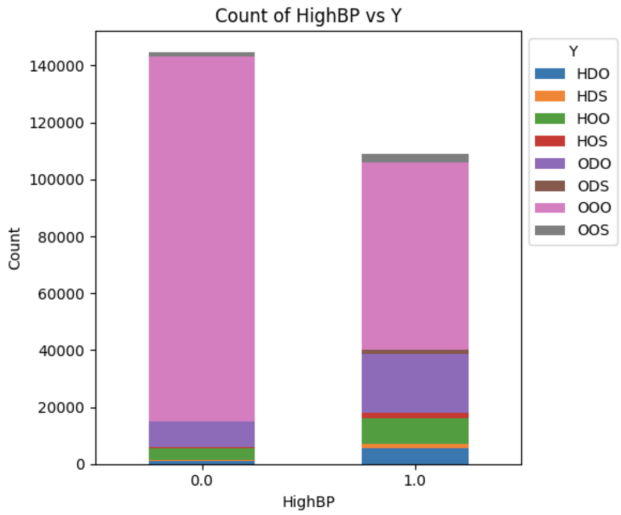

Figure 2: Count plot of high blood pressure vs Y.

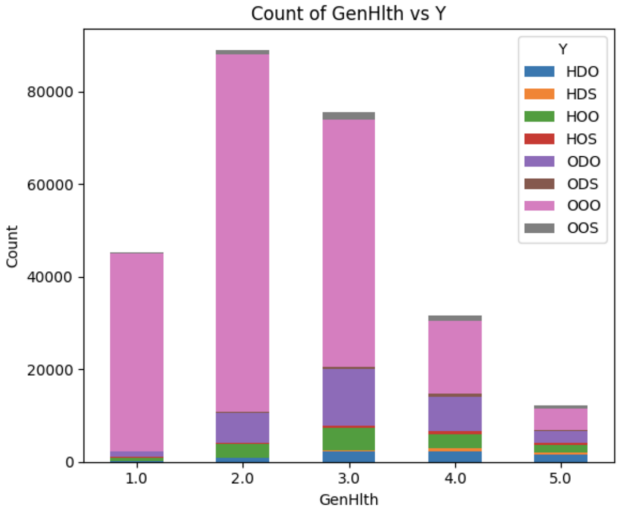

Figure 3: Count plot of general health vs Y.

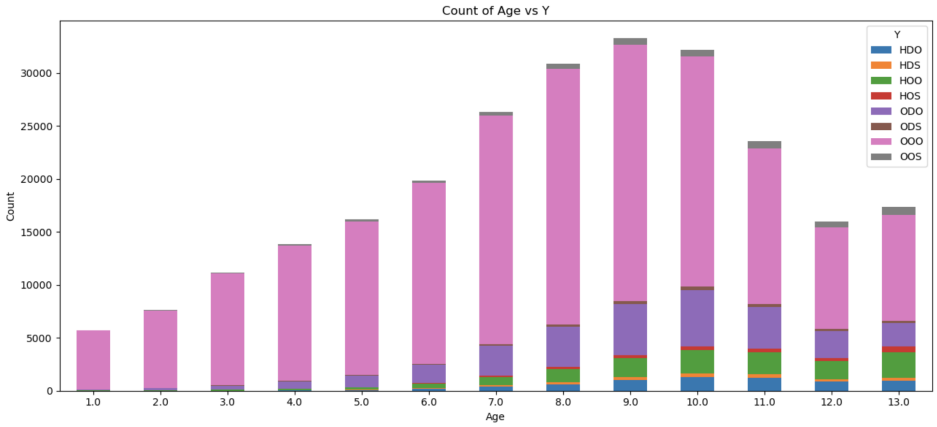

Figure 4: Count plot of age vs Y.

The analysis of conditional entropy and mutual information provides valuable insights into the predictive power of various health, lifestyle, and demographic indicators in relation to heart disease, diabetes, and stroke. Looking at the conditional entropy plot in Figure 5, variables such as "GenHlth," "HighBP," "Age," "DiffWalk," and "HighChol" emerge as significant contributors to predicting the response variable. These variables exhibit relatively low uncertainty, implying that they offer reliable information for making predictions. For instance, "GenHlth" and "HighBP" demonstrate the lowest conditional entropy values of approximately 0.812 and 0.833, respectively, suggesting a high degree of predictability based on these factors. Similarly, "Age," "DiffWalk," and "HighChol" also exhibit low uncertainty, with conditional entropy values around 0.839, 0.850, and 0.854, respectively, indicating their potential to reliably influence the response variable.

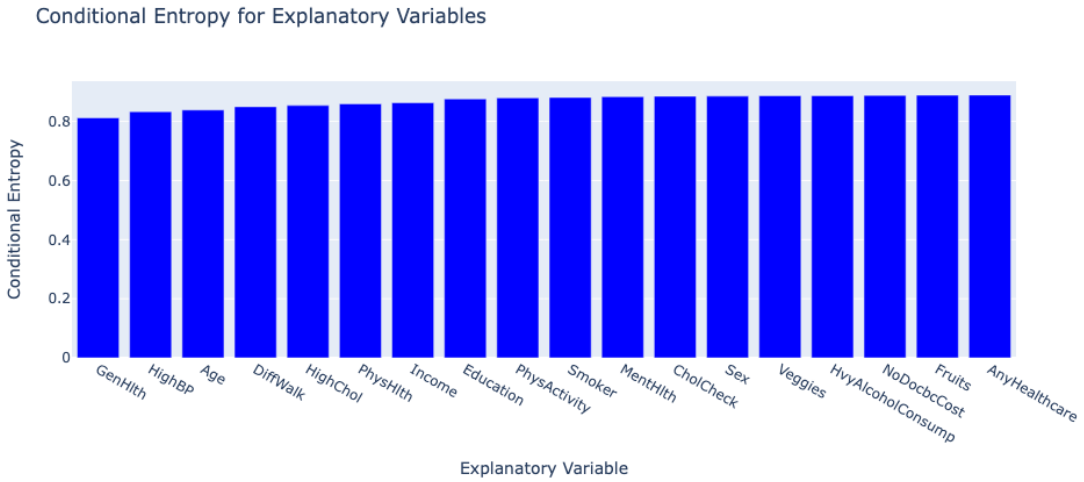

Figure 5: Conditional entropy plot of explanatory variables vs Y.

Examining the mutual information plot highlights the strength of the relationship between each explanatory variable and the response variable (Figure 6). Variables such as "GenHlth," "HighBP," "Age," "DiffWalk," and "HighChol" consistently show high mutual information values, indicating significant associations with the response variable. Notably, "GenHlth" stands out with the highest mutual information of approximately 0.077, suggesting a strong relationship with the response variable. Similarly, "HighBP," "Age," "DiffWalk," and "HighChol" exhibit substantial mutual information values, reinforcing their importance in predicting the response variable.

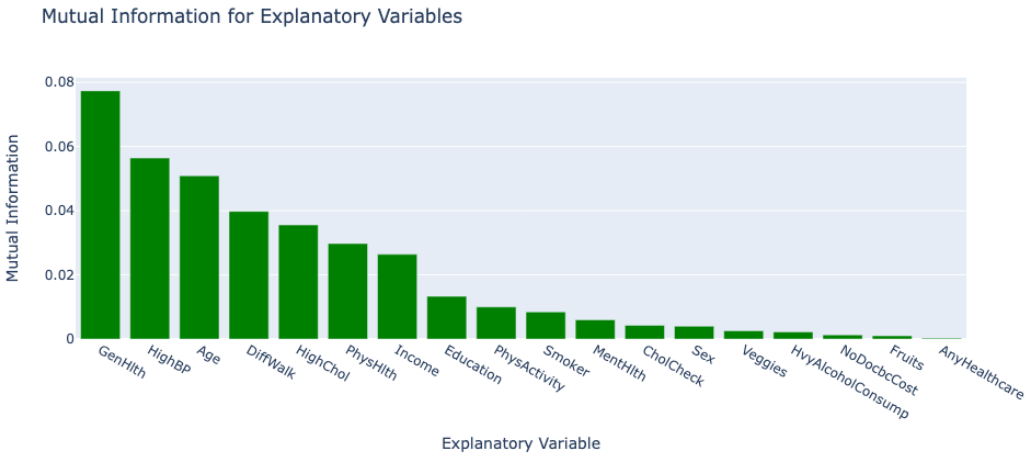

Figure 6: Mutual information plot of explanatory variables vs Y.

Next, looking at the conditional entropy plot with paired X variables, interactions such as "GenHlth—Age," "GenHlth—HighBP," and "GenHlth—HighChol" appear to be particularly informative for predicting the response variable (Figure 7). These interactions exhibit relatively low uncertainty, with conditional entropy values around 0.768, 0.781, and 0.791, respectively, indicating a high degree of predictability based on these paired variables. Additionally, interactions like "Age—BMI" and "GenHlth—DiffWalk" also demonstrate low uncertainty, with conditional entropy values around 0.796 and 0.804, respectively.

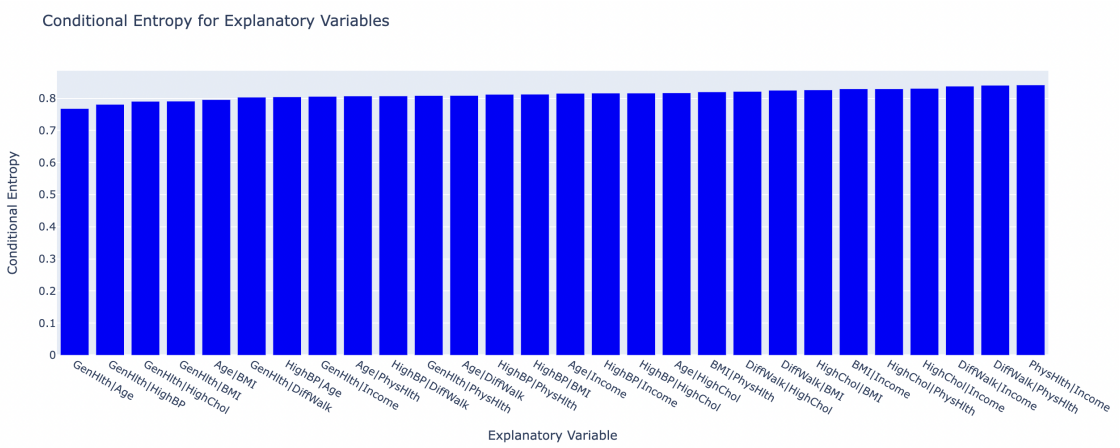

Figure 7: Conditional entropy plot of paired explanatory variables vs Y.

Examining the mutual information plot reveals the strength of associations between pairs of explanatory variables and the response variable (Figure 8). Interactions such as "GenHlth—Age," "GenHlth—HighBP," and "GenHlth—HighChol" consistently show high mutual information values, indicating strong relationships with the response variable. Notably, "GenHlth—Age" stands out with the highest mutual information of approximately 0.121. Similarly, interactions like "HighBP—Age" and "GenHlth—Income" exhibit substantial mutual information values, reinforcing their importance in influencing the response variable.

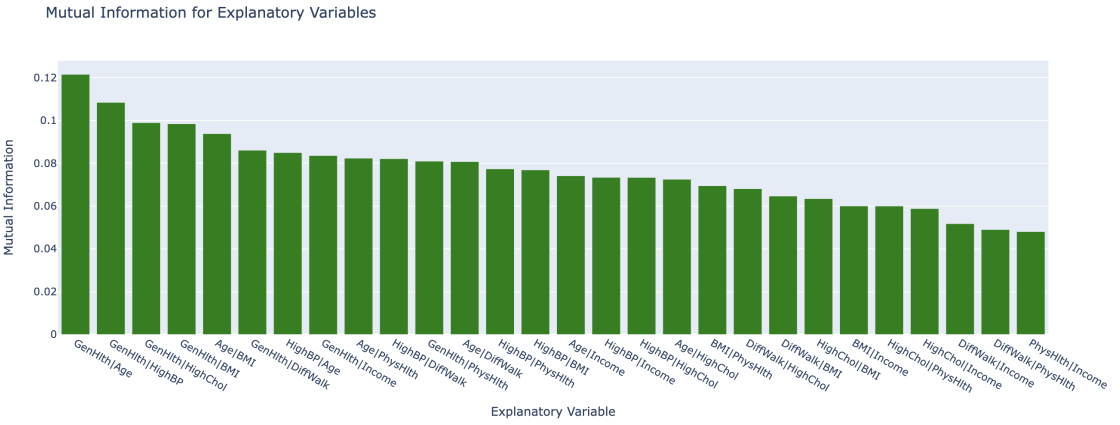

Figure 8: Mutual information plot of paired explanatory variables vs Y.

Finally, we looked at some of the important pairs above, like "GenHlth—Age" and "GenHlth—HighBP", by plotting them in stacked count plots. This helped us visualize and understand how our explanatory variables interact. For example, we can see that the people reviewing their general health as worse (levels 4 or 5) tend to be in the older age groups (Figure 9). Also, more than half of the individuals in the category with worst general health (level 5) report high blood pressure (Figure 10). We can see how the trends between our explanatory variables would then also result in high

association with our response Y.

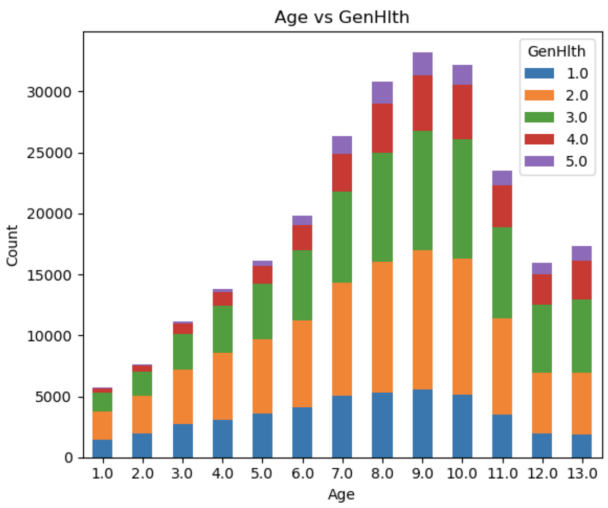

Figure 9: Count plot of age vs general health.

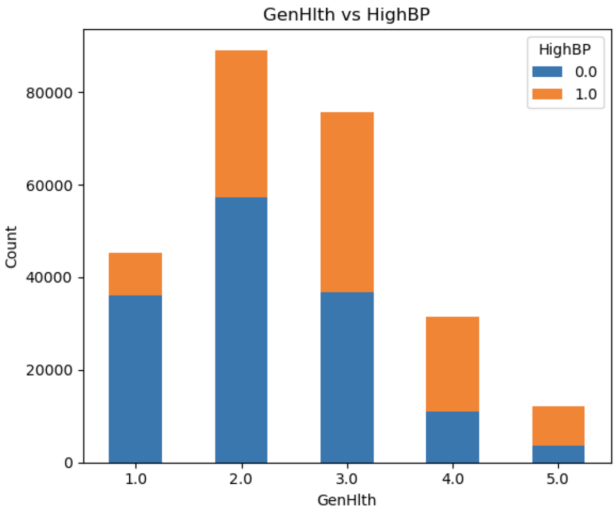

Figure 10: Count plot of general health vs BP.

5 Conclusion

From the examination of conditional entropy and mutual information, certain variables appear to be consistently associated with our response Y, such as general health, age, and high blood pressure. The figures highlight how people with lower general health perception, advanced age, and high blood pressure have higher chances of the conditions encompassed by Y. Additionally, our analysis of the interaction between various variables has shown that the pairings of general health, age, and high blood pressure have a significant effect on each other. Generally, as the perception of general health decreases the other two variables increase. This interplay between variables suggests that advanced age and high blood pressure catalyze lowered health perception. Furthermore, while alone these variables can heighten the chances of these conditions, together they can have an even greater effect.

As mentioned initially, we wanted to identify risk factors to help prevent heart disease, diabetes, and strokes. With our findings, we suggest that people strive to live healthy lifestyles and avoid indicators such as high blood pressure in order to have the best chance of preventing any disease. Although some of the important explanatory variables we identified, like older age, are not avoidable, it is still useful to keep in mind the relationship between aging and disease and we can still minimize risk when we are older if we maintain healthy living habits. These observations will be very helpful markers enabling health practitioners to identify people at higher risk of developing heart disease, diabetes, and stroke.

6 References

Centers for Disease Control and Prevention. (2023, April 4). *Diabetes fast facts.* Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/basics/quick-facts.html

Centers for Disease Control and Prevention. (2023, May 15). *Heart disease facts.* Centers for Disease Control and Prevention. https://www.cdc.gov/heartdisease/facts.htm

Centers for Disease Control and Prevention. (2023, May 4). *Stroke facts.* Centers for Disease Control and Prevention. https://www.cdc.gov/stroke/facts.htm

7 Appendix

```
"""STA 160 Collab.ipynb"""

Automatically generated by Colab.

Original file is located at
    https://colab.research.google.com/drive/1
       OSTeAbFLnRGhp03crsX3qfNpUIZT3mh6

"""# Read Data + Preprocessing"""

# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import entropy
import numpy as np
import plotly.express as px
import plotly.graph_objects as go
from itertools import combinations

# Read in data
df = pd.read_csv('/content/drive/MyDrive/STA 160/
   heartdisease_healthindicators.csv')

# Preprocess data
# Transform diabetes to binary variable
df['Diabetes'] = df['Diabetes'].replace(2,1)

# Create singular Y column encoded with H, D, and S for the three
   response variables
df['HeartDiseaseorAttackH'] = df['HeartDiseaseorAttack'].replace
   ([0,1],['O','H'])
df['DiabetesD'] = df['Diabetes'].replace([0,1],['O','D'])
df['StrokeS'] = df['Stroke'].replace([0,1],['O','S'])
df['Y'] = df['HeartDiseaseorAttackH']+df['DiabetesD']+df['StrokeS']

# Encode Y column to ordinal variable (levels 1 to 8)
df['Y_encoded'] = df['Y'].replace(['OOO','HOO','ODO','OOS','HDO','HOS
   ','ODS','HDS'],[1,2,3,4,5,6,7,8])
df.head()

"""# EDA + Data Visualization"""
"""## Lifestyle Variables"""

# Plotting explanatory vs response countplots - NOT STACKED
response_variable = "Y"

explanatory_variables = ["Smoker", "PhysActivity", "Fruits", "Veggies",
    "HvyAlcoholConsump", "DiffWalk"]
```

```
def plot_explanatory_vs_response(df):
    num_explanatory = len(explanatory_variables)
    fig, axes = plt.subplots(nrows=1, ncols=num_explanatory, figsize
        =(18, 6))

    for i, expl_var in enumerate(explanatory_variables):
        sns.countplot(data=df, x=expl_var, hue=response_variable, ax=
            axes[i])
        axes[i].set_title(f'{expl_var} vs {response_variable}')
        axes[i].set_xlabel(expl_var)
        axes[i].set_ylabel('Count')
        axes[i].legend(title=response_variable)

    plt.tight_layout()
    plt.show()

plot_explanatory_vs_response(df)

# Plotting stacked countplots
for var in explanatory_variables:
    # Define explanatory variable
    explanatory_variable = var

    # Define response variable
    response_variable = "Y"

    # Get unique Y groups
    unique_y_groups = sorted(df['Y'].unique())

    # Set colors for each unique value of the response variable
    response_colors = sns.color_palette("tab10", n_colors=len(
        unique_y_groups))

    # Group data by explanatory variable and response variable, and
        count occurrences
    counts = df.groupby([explanatory_variable, response_variable])[
        response_variable].count().unstack().fillna(0)

    # Initialize figure and axes
    fig, ax = plt.subplots(figsize=(6, 4))

    # Plot stacked bars
    counts.plot(kind='bar', stacked=True, color=response_colors, ax=ax)

    # Set labels and legend
    ax.set_xlabel(explanatory_variable)
    ax.set_ylabel('Count')
    ax.legend(title=response_variable, bbox_to_anchor=(1, 1))
    ax.set_title('Count of '+explanatory_variable+' vs Y')

    plt.xticks(rotation=0)
    plt.tight_layout()
```

```
    plt.show()

"""## Health Variables"""

# Plotting stacked countplots
for var in ['HighBP','HighChol','CholCheck','BMI','GenHlth','PhysHlth
    ','MentHlth']:
    # Define explanatory variable
    explanatory_variable = var

    # Define response variable
    response_variable = "Y"

    # Get unique Y groups
    unique_y_groups = sorted(df['Y'].unique())

    # Set colors for each unique value of the response variable
    response_colors = sns.color_palette("tab10", n_colors=len(
        unique_y_groups))

    # Group data by explanatory variable and response variable, and
        count occurrences
    counts = df.groupby([explanatory_variable, response_variable])[
        response_variable].count().unstack().fillna(0)

    # Initialize figure and axes
    fig, ax = plt.subplots(figsize=(6, 5))

    # Plot stacked bars
    counts.plot(kind='bar', stacked=True, color=response_colors, ax=ax)

    # Set labels and legend
    ax.set_xlabel(explanatory_variable)
    ax.set_ylabel('Count')
    ax.legend(title=response_variable, bbox_to_anchor=(1, 1))
    ax.set_title('Count of '+explanatory_variable+' vs Y')

    plt.xticks(rotation=0)
    plt.tight_layout()
    plt.show()

"""## Demographic Variables"""

# Creating Demographic Data Subset
demographic_data = df[['HeartDiseaseorAttack', 'Stroke', 'Diabetes',
'AnyHealthcare','NoDocbcCost', 'Sex', 'Age','Education', 'Income']]

# Getting Value Counts for HeartDiseaseorAttack vs Sex
sex_heartattack = demographic_data.groupby(['HeartDiseaseorAttack', '
    Sex']).size().unstack(fill_value=0)
sex_stroke = demographic_data.groupby(['Stroke', 'Sex']).size().unstack
    (fill_value=0)
```

```
sex_diabetes = demographic_data.groupby(['Diabetes', 'Sex']).size().
    unstack(fill_value=0)

# Defining and Populating Plots
fig, ax = plt.subplots(1, 3, figsize=(15, 5))

# HeartDiseaseorAttack
sex_heartattack.plot(kind='bar', ax=ax[0], stacked=False)
ax[0].set_title('HeartDiseaseorAttack vs Sex')
ax[0].set_xlabel('Heart Attack')
ax[0].set_ylabel('Count')
ax[0].legend(title='Sex', labels=['Female', 'Male'])

# Stroke
sex_stroke.plot(kind='bar', ax=ax[1], stacked=False)
ax[1].set_title('Stroke vs Sex')
ax[1].set_xlabel('Stroke')
ax[1].set_ylabel('Count')
ax[1].legend(title='Sex', labels=['Female', 'Male'])

# Diabetes
sex_diabetes.plot(kind='bar', ax=ax[2], stacked=False)
ax[2].set_title('Diabetes vs Sex')
ax[2].set_xlabel('Diabetes')
ax[2].set_ylabel('Count')
ax[2].legend(title='Sex', labels=['Female', 'Male'])
plt.show()

# Plotting stacked countplots
for var in ['AnyHealthcare','NoDocbcCost', 'Sex', 'Age','Education', '
    Income']:
    # Define explanatory variable
    explanatory_variable = var

    # Define response variable
    response_variable = "Y"

    # Get unique Y groups
    unique_y_groups = sorted(df['Y'].unique())

    # Set colors for each unique value of the response variable
    response_colors = sns.color_palette("tab10", n_colors=len(
        unique_y_groups))

    # Group data by explanatory variable and response variable, and
        count occurrences
    counts = df.groupby([explanatory_variable, response_variable])[
        response_variable].count().unstack().fillna(0)

    # Initialize figure and axes
    fig, ax = plt.subplots(figsize=(6, 5))
```

```
    # Plot stacked bars
    counts.plot(kind='bar', stacked=True, color=response_colors, ax=ax)

    # Set labels and legend
    ax.set_xlabel(explanatory_variable)
    ax.set_ylabel('Count')
    ax.legend(title=response_variable, bbox_to_anchor=(1, 1))
    ax.set_title('Count of '+explanatory_variable+' vs Y')

    plt.xticks(rotation=0)
    plt.tight_layout()
    plt.show()

"""# Odds"""

def odds_calculation(condition, predictor):
    '''
    Calculates odds given one response and predictor variable
    '''
    cross_table = pd.crosstab(df[condition], df[predictor])
    odds = cross_table.iloc[1] / cross_table.iloc[0]
    ratio = odds.iloc[1] / odds.iloc[0]
    print(f'Odds ratio for {condition} vs {predictor} is {ratio}')

# Heart Disease/Attack odds
explanatory_variables = ['HighBP','HighChol','CholCheck','Smoker','
   PhysActivity','Fruits','Veggies','HvyAlcoholConsump',
          'AnyHealthcare','NoDocbcCost','GenHlth','MentHlth','PhysHlth
             ','DiffWalk','Sex','Age','Education','Income']

for var in explanatory_variables:
  odds_calculation('HeartDiseaseorAttack',var)

"""# TA Helper Functions (Entropy + Concat Var)"""

# Helper functions from TA starter code
##Entropy
def entropy(Y):
    """
    Also known as Shanon Entropy
    Reference: https://en.wikipedia.org/wiki/Entropy_(
       information_theory)
    Args: Y -> A 1D vector/array
    Return: en -> The entropy of the input Y
    """
    unique, count = np.unique(Y, return_counts=True, axis=0)
    prob = count/len(Y)
    en = np.sum((-1)*prob*np.log(prob))
    return en

#Joint Entropy
def jEntropy(Y,X):
```

```
    """
    H(Y;X)
    Reference: https://en.wikipedia.org/wiki/Joint_entropy
    Args: Y & X -> Two 1D vectors/arrays
    Return: entropy of a new variable YX, derived from concatenating Y
       and X
    """
    YX = np.c_[Y,X]
    return entropy(YX)

#Conditional Entropy
def cEntropy(Y, X):
    """
    conditional entropy = Joint Entropy - Entropy of X
    H(Y|X) = H(Y;X) - H(X)
    Reference: https://en.wikipedia.org/wiki/Conditional_entropy
    """
    return jEntropy(Y, X) - entropy(X)

#Information Gain
def gain(Y, X):
    """
    Information Gain, I(Y;X) = H(Y) - H(Y|X)
    Reference: https://en.wikipedia.org/wiki/
       Information_gain_in_decision_trees#Formal_definition
    """
    return entropy(Y) - cEntropy(Y,X)

# Helper function from TA start code to concatenate two variables into
   single variable
def concat_var(df, vars, new_var_name = ''):
    """
    Combine multiple categorical variables into a single variable

    Args: df -> dataframe containing the variables to be concatenated
          vars -> list of variables to be concatenated
          new_var_name -> the name of the newly created variable from
             the cacatenation process
    Return: df -> df with an addition column containing the newly
       created variable
    """
    if new_var_name == '':
        new_var_name = '|'.join(vars)
    df_ = df.copy()
    new_vars =  np.c_[df_[vars]]
    df_[new_var_name] = ["|".join(item) for item in new_vars.astype('
       str')]
    return df_

"""# CE + MI for individual X against Y"""

explanatory_variables = ['HighBP','HighChol','CholCheck','Smoker','
```

```
    PhysActivity','Fruits','Veggies','HvyAlcoholConsump',
           'AnyHealthcare','NoDocbcCost','GenHlth','MentHlth','PhysHlth
              ','DiffWalk','Sex','Age','Education','Income','BMI']

# Calculate conditional entropy and mutual information
CE_MI_results = []
for var in explanatory_variables:
    CE = cEntropy(df['Y_encoded'], df[var])
    MI = gain(df['Y_encoded'], df[var])
    CE_MI_results.append([var, CE, MI])

# Create a DataFrame for the results
CE_MI_df = pd.DataFrame(CE_MI_results, columns=['Variable', '
    Conditional Entropy', 'Mutual Information'])
CE_MI_df.sort_values('Mutual Information', ascending=False, inplace=
    True)

CE_MI_df

# Plots to visualize the conditional entropy and mutual information

# Create trace for CE
ce_trace = go.Bar(x=CE_MI_df['Variable'], y=CE_MI_df['Conditional
    Entropy'], name='Conditional Entropy', marker_color='blue')

# Create the figure
fig = go.Figure(data=[ce_trace])

# Update layout
fig.update_layout(
    title='Conditional Entropy for Explanatory Variables',
    xaxis_title='Explanatory Variable',
    yaxis_title='Conditional Entropy',
)

# Show the plot
fig.show()

# Create trace for MI
mi_trace = go.Bar(x=CE_MI_df['Variable'], y=CE_MI_df['Mutual
    Information'], name='Mutual Information', marker_color='green')

# Create the figure
fig = go.Figure(data=[mi_trace])

# Update layout
fig.update_layout(
    title='Mutual Information for Explanatory Variables',
    xaxis_title='Explanatory Variable',
    yaxis_title='Mutual Information',
)
```

```
# Show the plot
fig.show()

"""# CE + MI for paired X against Y"""

# Taking top 8 most important X variables from above analysis
impt_variables = ['GenHlth','HighBP','Age','DiffWalk','HighChol','BMI
    ','PhysHlth','Income']

# Get all possible pairs of these 8 variables and concatenate them into
     a single variable
for combo in combinations(impt_variables, 2):
    df = concat_var(df,list(combo))

# Create new data frame that only has the concatenated variables and
    our Y variable
concat_df = df.iloc[:,-30:]
concat_df.head()

# Calculate conditional entropy and mutual information for combined
    variables

explanatory_paired = concat_df.columns.values.tolist()[2:]

CE_MI_results = []
for var in explanatory_paired:
    CE = cEntropy(concat_df['Y_encoded'], concat_df[var].tolist())
    MI = gain(concat_df['Y_encoded'], concat_df[var].tolist())
    CE_MI_results.append([var, CE, MI])

# Create a DataFrame for the results
CE_MI_concat_df = pd.DataFrame(CE_MI_results, columns=['Variable', '
    Conditional Entropy', 'Mutual Information'])
CE_MI_concat_df.sort_values('Mutual Information', ascending=False,
    inplace=True)

CE_MI_concat_df

# Plot conditional entropy and mutual information for combined
    variables

import plotly.graph_objects as go

# Create trace for CE
ce_trace = go.Bar(x=CE_MI_concat_df['Variable'], y=CE_MI_concat_df['
    Conditional Entropy'], name='Conditional Entropy', marker_color='
    blue')

# Create the figure
fig = go.Figure(data=[ce_trace])

# Update layout
```

```
fig.update_layout(
    title='Conditional Entropy for Explanatory Variables',
    xaxis_title='Explanatory Variable',
    yaxis_title='Conditional Entropy',
)

# Show the plot
fig.show()

# Create trace for MI
mi_trace = go.Bar(x=CE_MI_concat_df['Variable'], y=CE_MI_concat_df['
    Mutual Information'], name='Mutual Information', marker_color='green
    ')

# Create the figure
fig = go.Figure(data=[mi_trace])

# Update layout
fig.update_layout(
    title='Mutual Information for Explanatory Variables',
    xaxis_title='Explanatory Variable',
    yaxis_title='Mutual Information',
)

# Show the plot
fig.show()

"""# Data visualization of paired X"""
# Plot Age vs GenHlth
explanatory_variable = "Age"
response_variable = "GenHlth"

# Get unique Y groups
unique_y_groups = sorted(df['GenHlth'].unique())

# Set colors for each unique value of the response variable
response_colors = sns.color_palette("tab10", n_colors=len(
    unique_y_groups))

# Group data by explanatory variable and response variable, and count
    occurrences
counts = df.groupby([explanatory_variable, response_variable])[
    response_variable].count().unstack().fillna(0)

# Initialize figure and axes
fig, ax = plt.subplots(figsize=(6, 5))

# Plot stacked bars
counts.plot(kind='bar', stacked=True, color=response_colors, ax=ax)

# Set labels and legend
ax.set_xlabel(explanatory_variable)
```

```
ax.set_ylabel('Count')
ax.legend(title=response_variable, bbox_to_anchor=(1, 1))
ax.set_title('Age vs GenHlth')

plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

# Plot GenHlth vs HighBP
explanatory_variable = "GenHlth"
response_variable = "HighBP"

# Get unique Y groups
unique_y_groups = sorted(df['HighBP'].unique())

# Set colors for each unique value of the response variable
response_colors = sns.color_palette("tab10", n_colors=len(
    unique_y_groups))

# Group data by explanatory variable and response variable, and count
    occurrences
counts = df.groupby([explanatory_variable, response_variable])[
    response_variable].count().unstack().fillna(0)

# Initialize figure and axes
fig, ax = plt.subplots(figsize=(6, 5))

# Plot stacked bars
counts.plot(kind='bar', stacked=True, color=response_colors, ax=ax)

# Set labels and legend
ax.set_xlabel(explanatory_variable)
ax.set_ylabel('Count')
ax.legend(title=response_variable, bbox_to_anchor=(1, 1))
ax.set_title('GenHlth vs HighBP')

plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```