STA 160: Midterm Project

Analysis of Feature Importance for Heart Disease, Stroke, and Diabetes Across Sub-Populations

Andrew Muench, Michelle Tsang, Connor Young

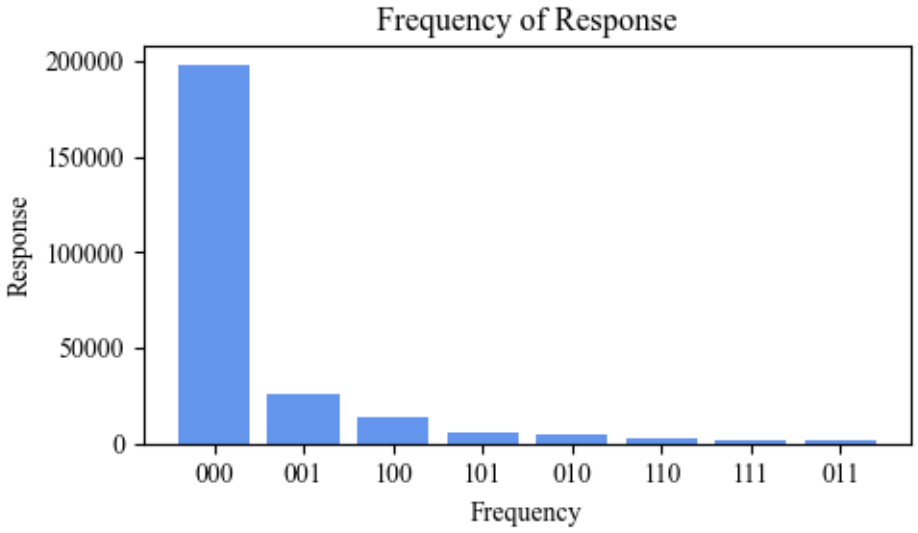
Introduction

According to the Center of Disease Control and Prevention (2023, November 20), "Heart disease is the leading cause of death for men, women, and people of most racial and ethnic groups in the United States." The CDC (2023, May 15) states that "heart disease cost the United States $239.9 Billion each year from 2018 to 2019. This includes the cost of healthcare services, medicines, and lost productivity due to death." Heart disease drains the United States of our loved ones and our wealth. It is in the interest of all to prevent heart disease. Therefore, it is important that we investigate how different personal attributes relate to heart disease and stroke.

Diabetes is a national public health concern as it is quite common and is growing more prevalent over time. According to the CDC, "38% of the U.S. population has diabetes" and "prevalence estimates for total diabetes were 10.3% in 2001–2004 and 13.2% in 2017–2020." The CDC (2022, July 7) also states that diabetes is a chronic disease with no cure that can cause heart problems, kidney problems and vision loss. And According to the American Diabetes Association (2023), diabetes costs $12,022 annually for the average person with diabetes. Diabetes can cause serious health issues for people and cost them and their families a large amount of money. Because diabetes is a prevalent and a rising concern for the nation with serious consequences in ignoring the disease, it is within everyone's interest to investigate how different personal attributes relate to diabetes.

Data Cleaning and Summaries

In our project, we recoded several features to make them easier to work with and analyze. For Sex, we recoded 0 to be Female and 1 to be Male. We also recoded some features from integer codes to the what they represented. For age ranges, we changed the integer codes of 1-13 to age buckets of "0-18" up to "65+." For income, we change the coding from 1-8 to "$0-$10,000" up to "$75,000+." For education, we changed the coding from 1-6 to "No school or only kindergarten" up to "College graduate". Finally, we created the response variable by combining three columns `HeartDiseaseorAttack`, `Stroke`, and `Diabetes` columns of the original data. The response variable is a 3 digit binary variable with each digit representing the presence of one of the diseases. 0 represents the absence of the disease and 1 represents the presence of the disease. The first digit represents `HeartDiseaseorAttack`, the second digit represents `Stroke`, and the third digit represents `Diabetes`.



Above is the histogram of the counts of each response variable type ranked from greatest to least. Looking at the histogram, people with no diseases had more responses than all the other response types combined. Of the responses that included diseases, `Diabetes`, `Heart Disease`, `Heart Disease and Diabetes` were the three largest response.

Methodology

For this project, we want to understand how the available health features affect the presence of `Heart Disease or Attack`, `Stroke`, and `Diabetes`. We will use several methods from information theory how information of these response variable are explained by the health features.

We use entropy which is the average amount of uncertainty that is part of the features' different outcomes. We mainly use entropy as part of formulas for other methods that we use in the project.

The formula for entropy is:

$$H(X) = -\sum_{x \in X} p(x) \log(x)$$

We use conditional entropy to understand whether a categorical feature affects the response variable. The conditional entropy is used to understand how much information is needed to explain the response variable by the features. In the project, we compare conditional entropy of the features and the response variable to see which features are the best at explaining the response variable.

The formula for Conditional Entropy is:

$$H(Y|X) = -\sum_{x \in X,\, y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)}$$

We use mutual information to observe how different features relate with the response variable. Mutual information tells us the amount of information described in the response variable by observing the variable of one of the health indicators. In the project, we compare the mutual information of the several features and the response variable to see which features are the best at explaining the response variable.

The formula for mutual information is

$$I(X,Y) = H(Y) - H(Y|X)$$

We use mutual conditional entropy to observe how different features relate with the response variable. Mutual conditional entropy is a scale-free/standardized version of the Mutual Information in order to create a symmetric association. In the project, we compare the mutual conditional information of the several features and the response variable to see which features are the best at explaining the response variable.

The formula for mutual conditional entropy is created citing the entropy and conditional entropy formulas above and is:

$$MCE(Y,X) = \frac{1}{2} \cdot \left(\frac{H(Y) - H(Y|X)}{H(Y)} + \frac{H(X) - H(X|Y)}{H(X)} \right)$$

We use the joint conditional entropy as a way to calculate the joint mutual conditional entropy which will be explained after this. The joint conditional entropy is used to understand how much information is needed

to explain the response variable by the interaction of two features.

The formula for joint conditional entropy is:

$$H(Y|X,Z) = \sum_{x\in X, y\in Y, z\in Z} p(x,y,z)\log\left(\frac{p(x,y,z)}{p(x,z)}\right)$$

We use joint mutual conditional entropy to observe how different interactions of features relate to the response variable. The joint mutual conditional entropy tells us the amount of information described in the response variable by observing the interaction of two features. In the project, we compare the joint mutual conditional entropy of the several interactions of features and the response variable to see which features are the best at explaining the response variable.

The formula for joint mutual conditional entropy is made from the mutual conditional entropy and joint conditional entropy formulas and is

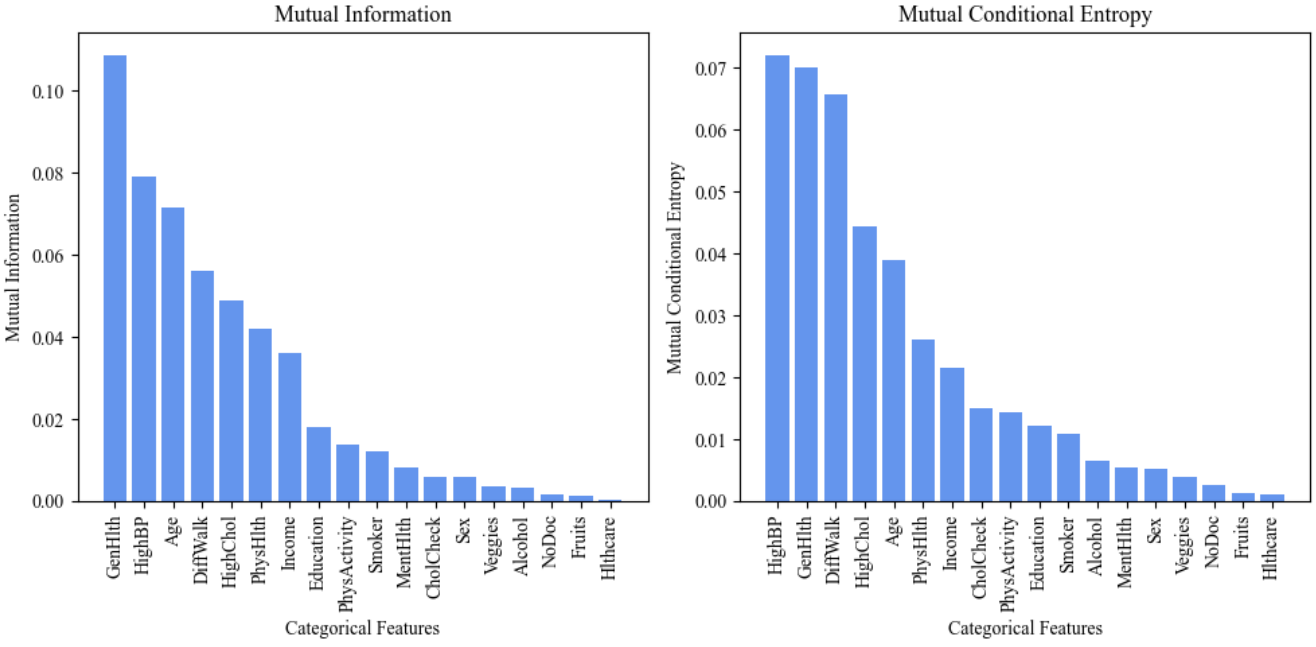
$$MCE(Y,X,Z) = \frac{1}{2}\cdot\left(\frac{H(Y)-H(Y|X,Z)}{H(Y)} + \frac{H(X,Z)-H(X,Z|Y)}{H(X,Z)}\right)$$

Results and Discussion

Population Level Entropy Analysis

We begin our analysis by calculating the mutual information and mutual conditional entropy between the response variable and each categorical feature using the entire population. These plots will tell us which features may provide the most information in determining the response variable. Thus, the features with the highest mutual information and MCE will be features of interest. The plots mutual information and mutual conditional entropy are shown below.

Mutual Information and Conditional Entropy of Response Given Categorical Variables

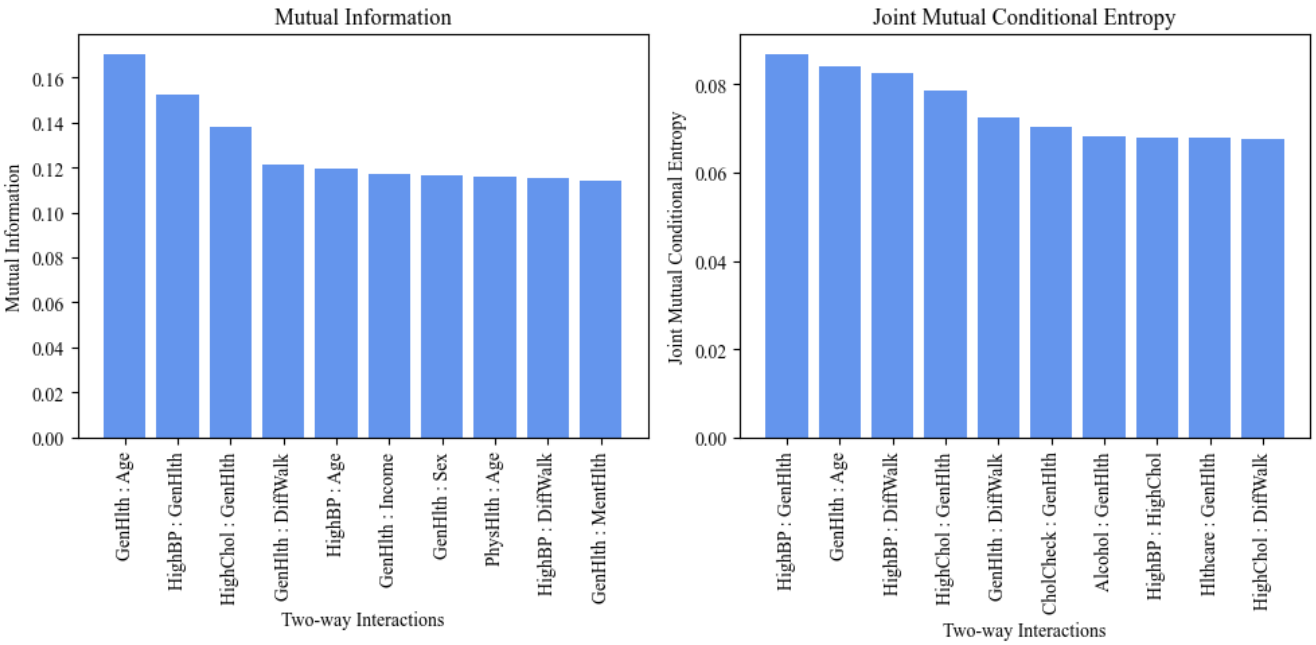


From the plot of mutual information, we can see that `GenHlth` has the highest mutual information by a substantial marginal compared to the other categorical features. `HighBP`, `Age`, and `DiffWalk` have the next highest mutual informations. Thus, we suspect these four features to be important indicators for diabetes, heart disease, and stroke.

From the plot of MCE, we can see that `HighBP`, `GenHlth`, and `DiffWalk` have the highest MCE out of all the categorical features (with the three features having relatively similar MCE). `HighChol` and `Age` have the next highest MCE. The insights from this plot further our suspicion that `GenHlth`, `HighBP`, `Age`, and `DiffWalk` are important features for the response variable.

We also want to investigate whether interactions between two categorical features are good indicators for diseases, so we will calculate the mutual information and mutual conditional entropy for two-way interactions. The top ten mutual information and MCE of the response variable given two-way interactions are shown in the graphs below.

Mutual Information and Conditional Entropy of Response Given Two-way Interactions



From the plot of mutual information, we can see that `GenHlth` is the most commonly paired categorical feature among the top ten interactions for mutual information. `GenHlth` is paired with `Age`, `HighBP`, `HighChol`, and `DiffWalk`, which are the same features with relatively high mutual information and MCE discussed previously. Thus, `GenHlth` interacting with other categorical features reduces the most uncertainty about the response variable.

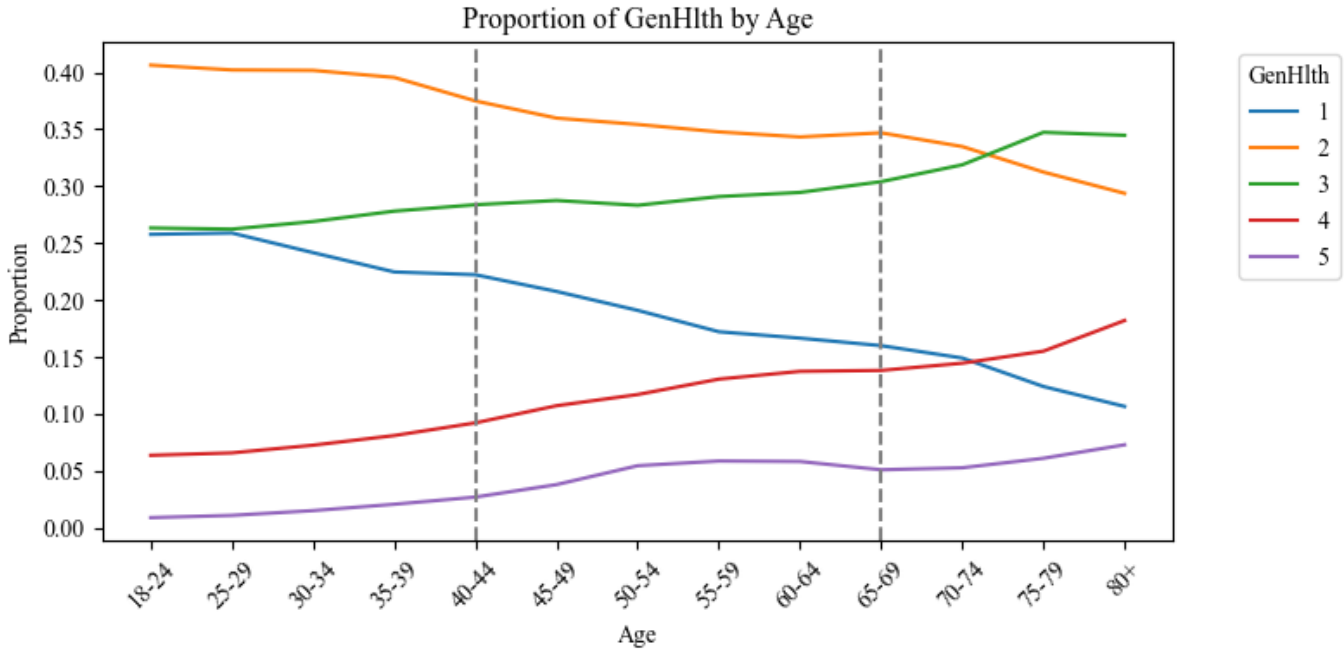
From the plot of MCE, we see that, again, `GenHlth` is the most commonly paired categorical feature among the top ten interactions for MCE. `HighBP`, `Age`, `DiffWalk`, and `HighChol` are commonly seen interacting with other categorical features or with each other. Interactions between these five features provides the most information for the response variable.

Based on these findings, we will conduct further entropy analysis on `GenHlth`, the categorical feature that

reduces the most entropy, while also providing the most influence over the response variable by itself and in interactions. To do this, We will create sub-populations from `Age` and look closely at how `HighBP`, `DiffWalk`, and `HighChol` changes within the sub-populations.

Age Regrouping

The `Age` feature has 13 categories by default, meaning that if we create sub-populations for each possible age group and level of `GenHlth`, there would be 65 sub-populations. This would be far too many sub-populations to properly analyze, so we want to reduce the number of age categories by grouping similar ages together. Since we will divide the population by `GenHlth`, we will also use `GenHlth` to reduce `Age`. We will compare the the proportion of each general health category per each age group to find similarities and trends that may indicate "natural" separations in the age categories. A plot of these proportions is shown below.

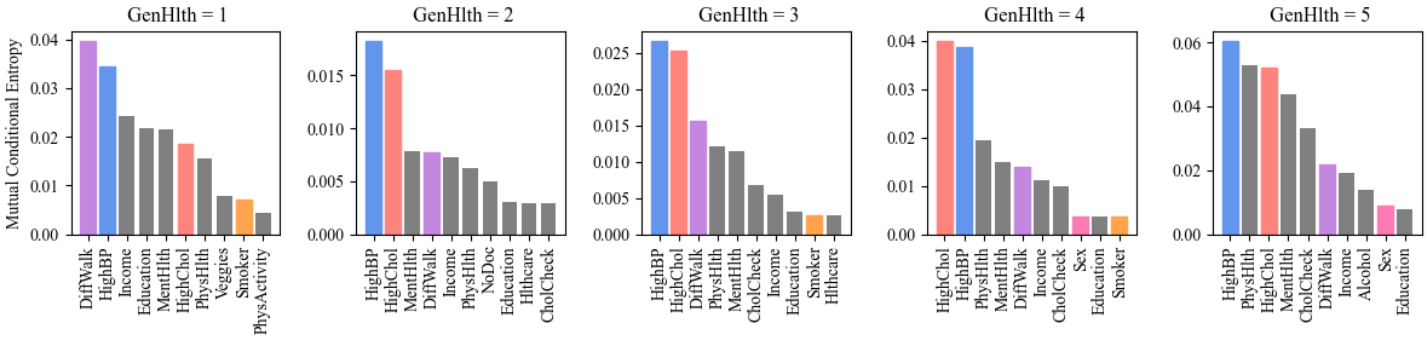


In the above plot we see two dotted grey lines, one at 40-44 and the other at 65-69. These lines indicate the separations we derived from the trends in the plot. The separation at 40-44 was determined by the downwards trend in the proportions of general health 1 and 2, and the upwards trend in the proportions of general health 3, 4, and 5. The separation at 65-69 was determined by another downwards trend in the proportions of general health 1 and 2, and another upwards in the proportions of general health 3, 4, and 5. Thus, we reduced the `Age` feature to 3 categories: 18-44, 45-64, and 65+.

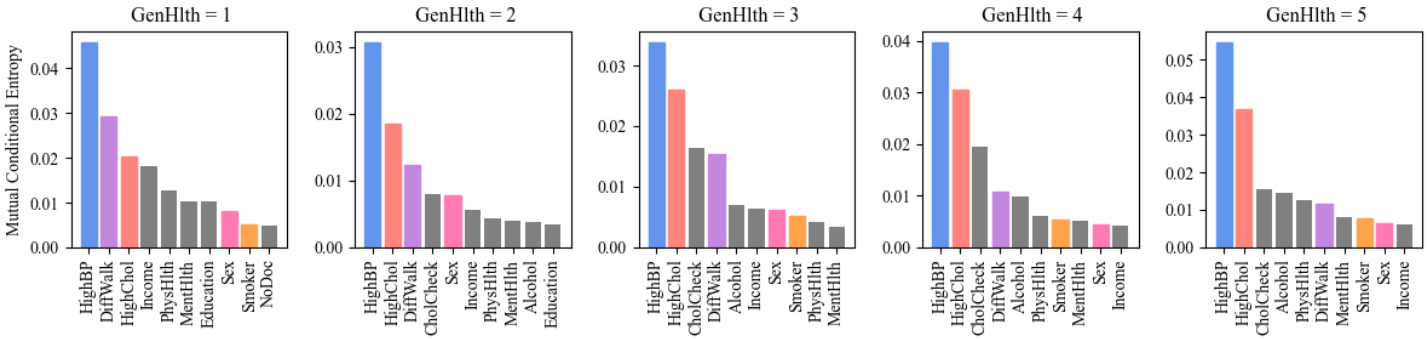
GenHlth and Age Sub-Population Entropy Analysis

Now that the `Age` feature has been reduced to a manageable amount, we can divide the population by each `GenHlth` and `Age` category into 15 unique sub-populations. For each sub-population, we calculate the MCE of the response given each categorical feature and report the 10 features with the highest MCE. The plots of entropies for each sub-population are shown below. Additionally, certain features of interest have been colored: blue is `HighBP`, purple is `DiffWalk`, red is `HighChol`, orange is `Smoker`, and pink is `Sex`.

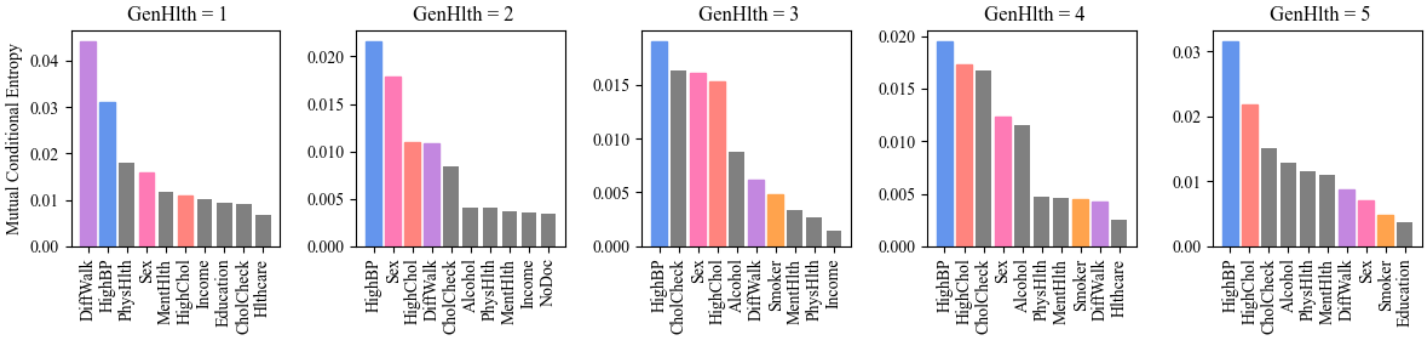
Mutual Conditional Entropy of Response Given Categorical Features for 18-44 Age Group



Mutual Conditional Entropy of Response Given Categorical Features for 45-64 Age Group



Mutual Conditional Entropy of Response Given Categorical Features for 65+ Age Group



In the above plots, we want to see what the top features are for each sub-population and how the top features may change between sub-populations.

For the 18-44 age group, we can see that for general health 2, 3, 4, and 5, the top three features include `HighBP` and `HighChol`. We can also see that `DiffWalk` has relatively high importance for the better general health of 1, 2, and 3. We can also see that `Smoker` and `Sex` do not provide much information for this age group since for all general health values, these features are either not highly ranked or not in the top 10 at all.

For the 45-64 age group, we can see that the top 3 features for each general health value are similar to those of the 18-44 age group, with `HighBP`, `HighChol`, and `DiffWalk` ranking highly. Notably, `HighBP` has the highest MCE for all general health values. Another observation we can make similar to the 18-44 age group is that as general health worsens, the MCE of `DiffWalk` seems to decrease. In contrast to the 18-44 age group, we see that `Sex` has relatively higher MCE, being in the top 10 features for all general health values. `Sex` also has the fourth highest MCE for the sub-population with a general health of two.

For the 65+ age group, we can see that `HighBP`, `HighChol`, and `DiffWalk` continue to have high MCE. However, for general health 4 and 5, `DiffWalk` is lower in the top 10 compared to the younger age groups. We can also see that `HighBP` continues to rank at or near number 1 for all general health values. Similar to the 45-64 age group, we can see that `Sex` has higher rankings compared to the 18-44 age group, with `Sex` being in the top 5 highest MCE for all general health values but 5.

From these plots and observations, we can see that some features have relatively high MCE and information across most or all sub-populations (`HighBP`, `HighChol`, `DiffWalk`), while other features only have relatively high MCE and information in specific sub-populations (`Sex`). We can also see how the same feature will have different MCE and provide different levels of information in different sub-populations.

Conclusion

In order to investigate how the different explanatory features affect the response variable we looked at the mutual information and conditional entropies. From the mutual information and conditional entropy plots, we found that `GenHlth`, `HighBP`, `Age`, `DiffWalk` and `HighChol` consistently had the highest mutual information and MCE. Then, we found the interactions between the explanatory features to observe how they jointly affected the response variable. We found that the two interactions that had the highest mutual information and joint mutual conditional entropy were `GenHlth:Age` and `HighBP:GenHlth`.

We investigated the relationships in the different sub-populations for higher values of mutual conditional entropy. We chose the sub-populations of `GenHlth` and `Age` because they had greater values of mutual information and joint mutual conditional entropy. There were too many levels for the age category so we combined age categories together by similarity of proportions or trends. We created three levels: 18-44, 45-64 and 65+. Then we found the mutual conditional entropy between the explanatory features and the response variable for the sub-population combinations of each level of `GenHlth` and `Age`.

We noticed some features retained high degrees of importance regardless of sub-population such as `HighBP` and `HighChol`. Other features change significantly between the different sub-populations such as `DiffWalk`, `Sex`, and `Smoker`. Overall, diseases are incredible complex and it is naive to expect a single feature to provide meaningful information in determining the presence of three diseases. Thus, it is vital to look at smaller, specific sub-populations to find patterns that may be hidden when looking at the population as a single entity.

References

Centers for Disease Control and Prevention. (2022, July 7). What is Diabetes? Centers for Disease Control and Prevention. https://www.cdc.gov/diabetes/basics /diabetes

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

National Diabetes Statistics Report — Diabetes — CDC. (2023, November 20). www.cdc.gov/diabetes/data/statistics-report/index

Parker, E. D., Jyh Dong Lin, Mahoney, T. J., Nwanneamaka Ume, Yang, G., Gabbay, R. A., ElSayed, N. A., & Bannuru, R. R. (2023). Economic Costs of Diabetes in the U.S. in 2022. Diabetes Care, 47(1). https://doi.org/10.2337/dci23-0085

Appendix

GitHub Repository

Data Processing

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data_original = pd.read_csv('./Data/heart_disease_health_indicators_BRFSS2015.csv')

### Data Value Clarifications

# change diabetes == 1 to 0 and diabetes == 2 to 1
data_modified = data_original.copy()
data_modified['Diabetes'] = data_modified['Diabetes'].replace(1, 0)
data_modified['Diabetes'] = data_modified['Diabetes'].replace(2, 1)

# change age ordinal values to age ranges
age_map = {1: '18-24', 2: '25-29', 3: '30-34', 4: '35-39', 5: '40-44', 6: '45-49', 7: '50-54
    ', 8: '55-59', 9: '60-64', 10: '65-69', 11: '70-74', 12: '75-79', 13: '80+'}
data_modified['Age'] = data_modified['Age'].map(age_map)

# change education ordinal values to descriptions
education_map = {1: 'No school or ony kindergarten', 2: 'Elementary', 3: 'Some high school',
     4: 'High school graduate', 5: 'Some college or technical school', 6: 'College graduate'
    }
data_modified['Education'] = data_modified['Education'].map(education_map)

# change income ordinal values to income ranges
income_map = {1: '$0-$10,000', 2: '$10,000-$15,000', 3: '$15,000-$20,000', 4: '$20,000-$25
    ,000', 5: '$25,000-$35,000', 6: '$35,000-$50,000', 7: '$50,000-$75,000', 8: '$75,000+'}
data_modified['Income'] = data_modified['Income'].map(income_map)

# make new column for all 3 response variables
data_modified['Response'] = data_modified['HeartDiseaseorAttack'].astype(int).astype(str) +
    data_modified['Stroke'].astype(int).astype(str) + data_modified['Diabetes'].astype(int).
    astype(str)

# change sex 0, 1 to female, male
sex_map = {0: 'Female', 1: 'Male'}
data_modified['Sex'] = data_modified['Sex'].map(sex_map)

# rename long variable names
data_modified = data_modified.rename(columns={'HvyAlcoholConsump': 'Alcohol', 'AnyHealthcare
    ': 'Hlthcare', 'NoDocbcCost': 'NoDoc'})

data_response = data_modified.drop(columns=['HeartDiseaseorAttack', 'Stroke', 'Diabetes'])
data_response.to_csv('./Data/STA160_Midterm_Data_Processed.csv', index=False)
```

EDA

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt

matplotlib.rcParams['font.sans-serif'] = "Times New Roman"
```

```
matplotlib.rcParams['font.family'] = "sans-serif"

data = pd.read_csv('./Data/STA160_Midterm_Data_Processed.csv', dtype={'Response': str})

columns = list(data.columns)
categorical_vars = columns[:3] + columns[4:-1]

### EDA

### Response Variable

data['Response'].value_counts().sort_values(ascending=False)

plt.figure(figsize=(5, 3))
plt.bar(data['Response'].value_counts().index, data['Response'].value_counts().values, color
     = '#6495ed')
plt.title('Frequency of Response')
plt.xlabel('Frequency')
plt.ylabel('Response')
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()

### Binary Variables
# HighBP
data['Response'].groupby(data['HighBP']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by HighBP')
plt.xlabel('HighBP')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# HighChol
data['Response'].groupby(data['HighChol']).value_counts().unstack().plot(kind='bar', stacked
    =False)
plt.title('Frequency of Response by HighChol')
plt.xlabel('HighChol')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# CholCheck
data['Response'].groupby(data['CholCheck']).value_counts().unstack().plot(kind='bar',
    stacked=False)
plt.title('Frequency of Response by CholCheck')
plt.xlabel('CholCheck')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# Smoker
data['Response'].groupby(data['Smoker']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by Smoker')
plt.xlabel('Smoker')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# PhysActivity
data['Response'].groupby(data['PhysActivity']).value_counts().unstack().plot(kind='bar',
```

```
    stacked=False)
plt.title('Frequency of Response by PhysActivity')
plt.xlabel('PhysActivity')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# Fruits
data['Response'].groupby(data['Fruits']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by Fruits')
plt.xlabel('Fruits')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# Veggies
data['Response'].groupby(data['Veggies']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by Veggies')
plt.xlabel('Veggies')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# HvyAlcoholConsump
data['Response'].groupby(data['Alcohol']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by HvyAlcoholConsump')
plt.xlabel('HvyAlcoholConsump')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# AnyHealthcare
data['Response'].groupby(data['Hlthcare']).value_counts().unstack().plot(kind='bar', stacked
    =False)
plt.title('Frequency of Response by AnyHealthcare')
plt.xlabel('AnyHealthcare')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# NoDocbcCost
data['Response'].groupby(data['NoDoc']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by NoDocbcCost')
plt.xlabel('NoDocbcCost')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# DiffWalk
data['Response'].groupby(data['DiffWalk']).value_counts().unstack().plot(kind='bar', stacked
    =False)
plt.title('Frequency of Response by DiffWalk')
plt.xlabel('DiffWalk')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# sex
```

```
data['Response'].groupby(data['Sex']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by Sex')
plt.xlabel('Sex')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# GenHlth
data['Response'].groupby(data['GenHlth']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by GenHlth')
plt.xlabel('GenHlth')
plt.ylabel('Frequency')
plt.xticks(rotation=0)
plt.show()

# Age
data['Response'].groupby(data['Age']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()

# Education
data['Response'].groupby(data['Education']).value_counts().unstack().plot(kind='bar',
    stacked=False)
plt.title('Frequency of Response by Education')
plt.xlabel('Education')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.show()

# Income
data['Response'].groupby(data['Income']).value_counts().unstack().plot(kind='bar', stacked=
    False)
plt.title('Frequency of Response by Income')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.xticks(rotation=90)
plt.show()
```

Entropy Analysis

```
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import itertools

matplotlib.rcParams['font.sans-serif'] = "Times New Roman"
matplotlib.rcParams['font.family'] = "sans-serif"

data = pd.read_csv('./Data/STA160_Midterm_Data_Processed.csv', dtype={'Response': str})

columns = list(data.columns)
categorical_vars = columns[:3] + columns[4:-1]

### Age Grouping
```

```
gen1 = []
gen2 = []
gen3 = []
gen4 = []
gen5 = []

for age_group in sorted(data['Age'].unique()):
    data_age_group = data[data['Age'] == age_group]

    proportions = data_age_group['GenHlth'].value_counts(normalize=True).sort_index()

    gen1.append(proportions.iloc[0])
    gen2.append(proportions.iloc[1])
    gen3.append(proportions.iloc[2])
    gen4.append(proportions.iloc[3])
    gen5.append(proportions.iloc[4])

# plot proportions of genhlth by age group
plt.figure(figsize = (8,4))
plt.plot(sorted(data['Age'].unique()), gen1, label='1')
plt.plot(sorted(data['Age'].unique()), gen2, label='2')
plt.plot(sorted(data['Age'].unique()), gen3, label='3')
plt.plot(sorted(data['Age'].unique()), gen4, label='4')
plt.plot(sorted(data['Age'].unique()), gen5, label='5')
plt.title('Proportion of GenHlth by Age')
plt.xlabel('Age')
plt.ylabel('Proportion')
plt.legend(title='GenHlth', bbox_to_anchor=(1.05, 1.0), loc='upper left')
plt.xticks(rotation=45)
# vertical line at 40-44
plt.axvline(x=4, color='grey', linestyle='--')
# vertical line at 65-69
plt.axvline(x=9, color='grey', linestyle='--')
plt.tight_layout()
plt.show()

age_group_map = {'18-24': '18-44', '25-29': '18-44', '30-34': '18-44', '35-39': '18-44', '
    40-44': '18-44', '45-49': '45-64', '50-54': '45-64', '55-59': '45-64', '60-64': '45-64',
     '65-69': '65+', '70-74': '65+', '75-79': '65+', '80+': '65+'}
data['Age Group'] = data['Age'].map(age_group_map)

### Entropy

#### Functions
def entropy(data, y):
    probs = list(data[y].value_counts(normalize=True))

    n = len(probs)

    e_sum = 0

    for i in range(n):
        if probs[i] != 0:
            e_sum += probs[i] * np.log2(probs[i])

    return -e_sum

def twoway_joint_entropy(data, x1, x2):
    cross = pd.crosstab(data[x1], data[x2], margins=True, normalize='all')

    num_x1 = cross.shape[0] - 1
    num_x2 = cross.shape[1] - 1
```

```
76
77      e_sum = 0
78
79      for i in range(num_x1):
80          for j in range(num_x2):
81              if cross.iloc[i, j] != 0:
82                  e_sum += cross.iloc[i, j] * np.log2(cross.iloc[i, j])
83
84      return -e_sum
85
86  def cond_entropy(data, y, x):
87      cross = pd.crosstab(data[y], data[x], margins=True, normalize='all')
88
89      num_response = cross.shape[0] - 1
90      num_cat = cross.shape[1] - 1
91
92      ce_sum = 0
93
94      for i in range(num_response):
95          for j in range(num_cat):
96              if cross.iloc[i, j] != 0:
97                  ce_sum += cross.iloc[i, j] * np.log2(cross.iloc[i, j] / cross.iloc[
    num_response, j])
98
99      return -ce_sum
100
101 def two_cond_entropy(data, y, x, z):
102     cross = pd.crosstab(data[y], [data[x], data[z]], margins=True, normalize='all')
103
104     num_response = cross.shape[0] - 1
105     num_cols = cross.shape[1] - 1
106
107     ce_sum = 0
108
109     for i in range(num_response):
110         for j in range(num_cols):
111             if cross.iloc[i, j] != 0:
112                 ce_sum += cross.iloc[i, j] * np.log2(cross.iloc[i, j] / cross.iloc[
    num_response, j])
113
114     return -ce_sum
115
116 def joint_cond_entropy(data, x, z, y):
117     cross = pd.crosstab(data[y], [data[x], data[z]], margins=True, normalize='all')
118
119     num_response = cross.shape[0] - 1
120     num_cols = cross.shape[1] - 1
121
122     ce_sum = 0
123
124     for i in range(num_response):
125         for j in range(num_cols):
126             if cross.iloc[i, j] != 0:
127                 ce_sum += cross.iloc[i, j] * np.log2(cross.iloc[i, j] / cross.iloc[i,
    num_cols])
128
129     return -ce_sum
130
131 def mutual_conditional_entropy(data, y, x):
132
133     ce_y = entropy(data, y)
134     ce_x = entropy(data, x)
```

```
135
136     ce_y_x = cond_entropy(data, y, x)
137     ce_x_y = cond_entropy(data, x, y)
138
139     mce_y_x = (((ce_y - ce_y_x) / ce_y) + ((ce_x - ce_x_y) / ce_x)) / 2
140
141     return mce_y_x
142
143 def twoway_mutual_conditional_entropy(data, y, x, z):
144
145     h_y = entropy(data, y)
146     h_xz = twoway_joint_entropy(data, x, z)
147
148     ce_y_xz = two_cond_entropy(data, y, x, z)
149     ce_xz_y = joint_cond_entropy(data, x, z, y)
150
151     mce_y_x = (((h_y - ce_y_xz) / h_y) + ((h_xz - ce_xz_y) / h_xz)) / 2
152
153     return mce_y_x
154
155 ### Entropy Results
156 response_entropy = entropy(data, 'Response')
157
158 conditional_entropies = {var: cond_entropy(data, 'Response', var) for var in
        categorical_vars}
159
160 # sorted conditional entropies
161 sorted_conditional_entropies = dict(sorted(conditional_entropies.items(), key=lambda x: x
        [1], reverse=False))
162
163 # bar plot of conditional entropy
164 plt.bar(sorted_conditional_entropies.keys(), sorted_conditional_entropies.values(), color =
        '#6495ed')
165 plt.title('Conditional Entropy of Response given Categorical Features')
166 plt.xlabel('Categorical Features')
167 plt.ylabel('Conditional Entropy')
168 plt.xticks(rotation=90)
169 # line for CE[Y]
170 plt.axhline(y=response_entropy, color='r', linestyle='--')
171 plt.show()
172
173 # mutual information
174 mutual_information = {var: response_entropy - cond_en for var, cond_en in
        conditional_entropies.items()}
175
176 # sorted mutual information
177 sorted_mutual_information = dict(sorted(mutual_information.items(), key=lambda item: item
        [1], reverse=True))
178
179 # bar plot of mutual information
180 plt.bar(sorted_mutual_information.keys(), sorted_mutual_information.values(), color = '#6495
        ed')
181 plt.title('Mutual Information of Response given Categorical Features')
182 plt.xlabel('Categorical Features')
183 plt.ylabel('Mutual Information')
184 plt.xticks(rotation=90)
185 plt.annotate(f'H[Y] = {response_entropy:.2f}', xy=(0, 0), xytext=(0.8, 0.95), textcoords='
        axes fraction')
186 plt.show()
187
188 mutual_conditional_entropies = {var: mutual_conditional_entropy(data, 'Response', var) for
        var in categorical_vars}
```

```
sorted_mutual_conditional_entropies = dict(sorted(mutual_conditional_entropies.items(), key=lambda x: x[1], reverse=True))

plt.bar(sorted_mutual_conditional_entropies.keys(), sorted_mutual_conditional_entropies.values(), color = '#6495ed')
plt.title(f'Mutual Conditional Entropy of Response given Categorical Features')
plt.xlabel('Categorical Features')
plt.ylabel('Mutual Conditional Entropy')
plt.xticks(rotation=90)
plt.show()

# side-by-side plots of mutual information and mutual conditional entropy
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].bar(sorted_mutual_information.keys(), sorted_mutual_information.values(), color = '#6495ed')
ax[0].set_title('Mutual Information')
ax[0].set_xlabel('Categorical Features')
ax[0].set_ylabel('Mutual Information')
ax[0].xaxis.set_tick_params(rotation=90)

ax[1].bar(sorted_mutual_conditional_entropies.keys(), sorted_mutual_conditional_entropies.values(), color = '#6495ed')
ax[1].set_title(f'Mutual Conditional Entropy')
ax[1].set_xlabel('Categorical Features')
ax[1].set_ylabel('Mutual Conditional Entropy')
ax[1].xaxis.set_tick_params(rotation=90)
plt.tight_layout()
plt.show()

### Entropy Interaction Results
variable_pairs = list(itertools.combinations(categorical_vars, 2))

# condtional entropy for all two way interactions
conditional_entropies_twoway = {f'{pair[0]} : {pair[1]}': two_cond_entropy(data, 'Response', pair[0], pair[1]) for pair in variable_pairs}
sorted_conditional_entropies_twoway = dict(sorted(conditional_entropies_twoway.items(), key=lambda x: x[1], reverse=False)[:10])

# plot of conditional entropy for all two way interactions
plt.bar(sorted_conditional_entropies_twoway.keys(), sorted_conditional_entropies_twoway.values(), color = '#6495ed')
plt.title('Conditional Entropy of Response given Two-way Interactions of Categorical Features')
plt.xlabel('Two-way Interactions')
plt.ylabel('Conditional Entropy')
plt.axhline(y=response_entropy, color='r', linestyle='--')
plt.xticks(rotation=90)
plt.show()

# mutual information for all two way interactions
mutual_information_twoway = {pair: response_entropy - cond_en for pair, cond_en in conditional_entropies_twoway.items()}
sorted_mutual_information_twoway = dict(sorted(mutual_information_twoway.items(), key=lambda item: item[1], reverse=True)[:10])

# plot of mutual information for all two way interactions
plt.bar(sorted_mutual_information_twoway.keys(), sorted_mutual_information_twoway.values(), color = '#6495ed')
plt.title('Mutual Information of Response given Two-way Interactions of Categorical Variables')
plt.xlabel('Two-way Interactions')
plt.ylabel('Mutual Information')
```

```
plt.xticks(rotation=90)
plt.annotate(f'H[Y] = {response_entropy:.2f}', xy=(0, 0), xytext=(0.8, 0.95), textcoords='
    axes fraction')
plt.show()

# twoway mutual conditional entropy for all interactions
twoway_mutual_conditional_entropies = {f'{pair[0]} : {pair[1]}':
    twoway_mutual_conditional_entropy(data, 'Response', pair[0], pair[1]) for pair in
    variable_pairs}
sorted_twoway_mutual_conditional_entropies = dict(sorted(twoway_mutual_conditional_entropies
    .items(), key=lambda x: x[1], reverse=True)[:10])

# plot of twoway mutual conditional entropy for all interactions
plt.bar(sorted_twoway_mutual_conditional_entropies.keys(),
    sorted_twoway_mutual_conditional_entropies.values(), color = '#6495ed')
plt.title('Joint Mutual Conditional Entropy of Response given Categorical Features')
plt.xlabel('Two-way Interactions')
plt.ylabel('Joint Mutual Conditional Entropy')
plt.xticks(rotation=90)
plt.show()

# side-by-side plots of mutual information and mutual conditional entropy
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].bar(sorted_mutual_information_twoway.keys(), sorted_mutual_information_twoway.values()
    , color = '#6495ed')
ax[0].set_title('Mutual Information')
ax[0].set_xlabel('Two-way Interactions')
ax[0].set_ylabel('Mutual Information')
ax[0].xaxis.set_tick_params(rotation=90)

ax[1].bar(sorted_twoway_mutual_conditional_entropies.keys(),
    sorted_twoway_mutual_conditional_entropies.values(), color = '#6495ed')
ax[1].set_title('Joint Mutual Conditional Entropy')
ax[1].set_xlabel('Two-way Interactions')
ax[1].set_ylabel('Joint Mutual Conditional Entropy')
ax[1].xaxis.set_tick_params(rotation=90)
plt.tight_layout()
plt.show()

### GenHlth and Age Sub-Populations
variables_of_interest = ['HighBP', 'DiffWalk', 'HighChol', 'PhysHlth', 'Income', 'CholCheck'
    , 'PhysActivity', 'Education', 'Smoker', 'Alcohol', 'MentHlth', 'Sex', 'Veggies', 'NoDoc
    ', 'Fruits', 'Hlthcare']

def plot_mutual_conditional_entropy_per_genhlth_and_age_group(data, age_group):

    data = data[data['Age Group'] == age_group]

    fig, axs = plt.subplots(1, 5, figsize=(12, 3))

    for val in sorted(data['GenHlth'].unique()):
        val = int(val)
        data_val = data[data['GenHlth'] == val]

        mutual_conditional_entropies = {var: mutual_conditional_entropy(data_val, var, '
    Response') for var in variables_of_interest}

        sorted_mutual_conditional_entropies = dict(sorted(mutual_conditional_entropies.items
    (), key=lambda x: x[1], reverse=True)[:10])

        plot = axs[val-1].bar(sorted_mutual_conditional_entropies.keys(),
    sorted_mutual_conditional_entropies.values(), color = 'grey')
```

```
289
290         # indices of desired variables
291         # and change color
292         var_lst = list(sorted_mutual_conditional_entropies.keys())
293         if 'HighBP' in sorted_mutual_conditional_entropies:
294             HighBP_idx = var_lst.index('HighBP')
295             plot[HighBP_idx].set_color('#6495ed')
296         if 'DiffWalk' in sorted_mutual_conditional_entropies:
297             DiffWalk_idx = var_lst.index('DiffWalk')
298             plot[DiffWalk_idx].set_color('#c387e0')
299         if 'HighChol' in sorted_mutual_conditional_entropies:
300             HighChol_idx = var_lst.index('HighChol')
301             plot[HighChol_idx].set_color('#ff847e')
302         if 'Smoker' in sorted_mutual_conditional_entropies:
303             Smoker_idx = var_lst.index('Smoker')
304             plot[Smoker_idx].set_color('#ffa34d')
305         if 'Sex' in sorted_mutual_conditional_entropies:
306             Sex_idx = var_lst.index('Sex')
307             plot[Sex_idx].set_color('#fe7ab5')
308
309         axs[val-1].set_title(f' GenHlth = {val}')
310         axs[val-1].xaxis.set_tick_params(rotation=90)
311
312     axs[0].set_ylabel('Mutual Conditional Entropy')
313     plt.tight_layout()
314     plt.show()
315     return
316
317 plot_mutual_conditional_entropy_per_genhlth_and_age_group(data, '18-44')
318
319 plot_mutual_conditional_entropy_per_genhlth_and_age_group(data, '45-64')
320
321 plot_mutual_conditional_entropy_per_genhlth_and_age_group(data, '65+')
```