Data Visualization of Diabetes Incidence Dataset

Data Visualization was used to carry out exploratory data analysis of the 50-50 split Diabetes Incidence Dataset sourced from Kaggle. The dataset contains 22 columns, 21 of which are the lifestyle predictors of the target variable - Diabetes Diagnosis. In order to better understand the data, both univariate and bivariate analysis was carried out along with the exploration of the relationship between the feature variables and the target variable.

Importing Necessary Libraries

In order to carry out data visualization of the dataset, Pandas Numpy, Matplotlib.pyplot and Seaborn were imported.

In [1]:
```
# import packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading and Initial Data Exploration

The dataset is loaded into a Pandas Dataframe and .head() is used to check that it was correctly imported and loaded.

In [3]:
```
# load and read dataset into dataframe
diabetes_df = pd.read_csv('/content/drive/MyDrive/diabetes.csv')
diabetes_df.head()
```

Out[3]:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHeal
0	0.0	1.0	0.0	1.0	26.0	0.0	0.0	0.0	1.0	0.0	...	
1	0.0	1.0	1.0	1.0	26.0	1.0	1.0	0.0	0.0	1.0	...	
2	0.0	0.0	0.0	1.0	26.0	0.0	0.0	0.0	1.0	1.0	...	
3	0.0	1.0	1.0	1.0	28.0	1.0	0.0	0.0	1.0	1.0	...	
4	0.0	0.0	0.0	1.0	29.0	1.0	0.0	0.0	1.0	1.0	...	

5 rows × 22 columns

In [4]:
```
# get information regarding the datatypes, columns, null value counts etc.
diabetes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70692 entries, 0 to 70691
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Diabetes_binary       70692 non-null  float64
 1   HighBP                70692 non-null  float64
 2   HighChol              70692 non-null  float64
 3   CholCheck             70692 non-null  float64
 4   BMI                   70692 non-null  float64
 5   Smoker                70692 non-null  float64
 6   Stroke                70692 non-null  float64
 7   HeartDiseaseorAttack  70692 non-null  float64
 8   PhysActivity          70692 non-null  float64
 9   Fruits                70692 non-null  float64
 10  Veggies               70692 non-null  float64
 11  HvyAlcoholConsump     70692 non-null  float64
 12  AnyHealthcare         70692 non-null  float64
 13  NoDocbcCost           70692 non-null  float64
 14  GenHlth               70692 non-null  float64
 15  MentHlth              70692 non-null  float64
 16  PhysHlth              70692 non-null  float64
 17  DiffWalk              70692 non-null  float64
 18  Sex                   70692 non-null  float64
 19  Age                   70692 non-null  float64
 20  Education             70692 non-null  float64
 21  Income                70692 non-null  float64
dtypes: float64(22)
memory usage: 11.9 MB
```

We can see that there are 22 columns of which the first is the target variable - Diabetes_binary and the rest are the lifestyle factors. All columns are of the float data type. Additionally, there are no null objects in any of the columns which is as expected since the dataset was cleaned prior to loading.

In [5]:
```
# basic statistical details about the data
# transformed for better visualization
```

```
diabetes_df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	max
Diabetes_binary	70692.0	0.500000	0.500004	0.0	0.0	0.5	1.0	1.0
HighBP	70692.0	0.563458	0.495960	0.0	0.0	1.0	1.0	1.0
HighChol	70692.0	0.525703	0.499342	0.0	0.0	1.0	1.0	1.0
CholCheck	70692.0	0.975259	0.155336	0.0	1.0	1.0	1.0	1.0
BMI	70692.0	29.856985	7.113954	12.0	25.0	29.0	33.0	98.0
Smoker	70692.0	0.475273	0.499392	0.0	0.0	0.0	1.0	1.0
Stroke	70692.0	0.062171	0.241468	0.0	0.0	0.0	0.0	1.0
HeartDiseaseorAttack	70692.0	0.147810	0.354914	0.0	0.0	0.0	0.0	1.0
PhysActivity	70692.0	0.703036	0.456924	0.0	0.0	1.0	1.0	1.0
Fruits	70692.0	0.611795	0.487345	0.0	0.0	1.0	1.0	1.0
Veggies	70692.0	0.788774	0.408181	0.0	1.0	1.0	1.0	1.0
HvyAlcoholConsump	70692.0	0.042721	0.202228	0.0	0.0	0.0	0.0	1.0
AnyHealthcare	70692.0	0.954960	0.207394	0.0	1.0	1.0	1.0	1.0
NoDocbcCost	70692.0	0.093914	0.291712	0.0	0.0	0.0	0.0	1.0
GenHlth	70692.0	2.837082	1.113565	1.0	2.0	3.0	4.0	5.0
MentHlth	70692.0	3.752037	8.155627	0.0	0.0	0.0	2.0	30.0
PhysHlth	70692.0	5.810417	10.062261	0.0	0.0	0.0	6.0	30.0
DiffWalk	70692.0	0.252730	0.434581	0.0	0.0	0.0	1.0	1.0
Sex	70692.0	0.456997	0.498151	0.0	0.0	0.0	1.0	1.0
Age	70692.0	8.584055	2.852153	1.0	7.0	9.0	11.0	13.0
Education	70692.0	4.920953	1.029081	1.0	4.0	5.0	6.0	6.0
Income	70692.0	5.698311	2.175196	1.0	4.0	6.0	8.0	8.0

Since a lot of the factors have binary inputs this step doesn't give us too much information.

Univariate Analysis

The frequency distribution of each individual variable in the dataset was generated.

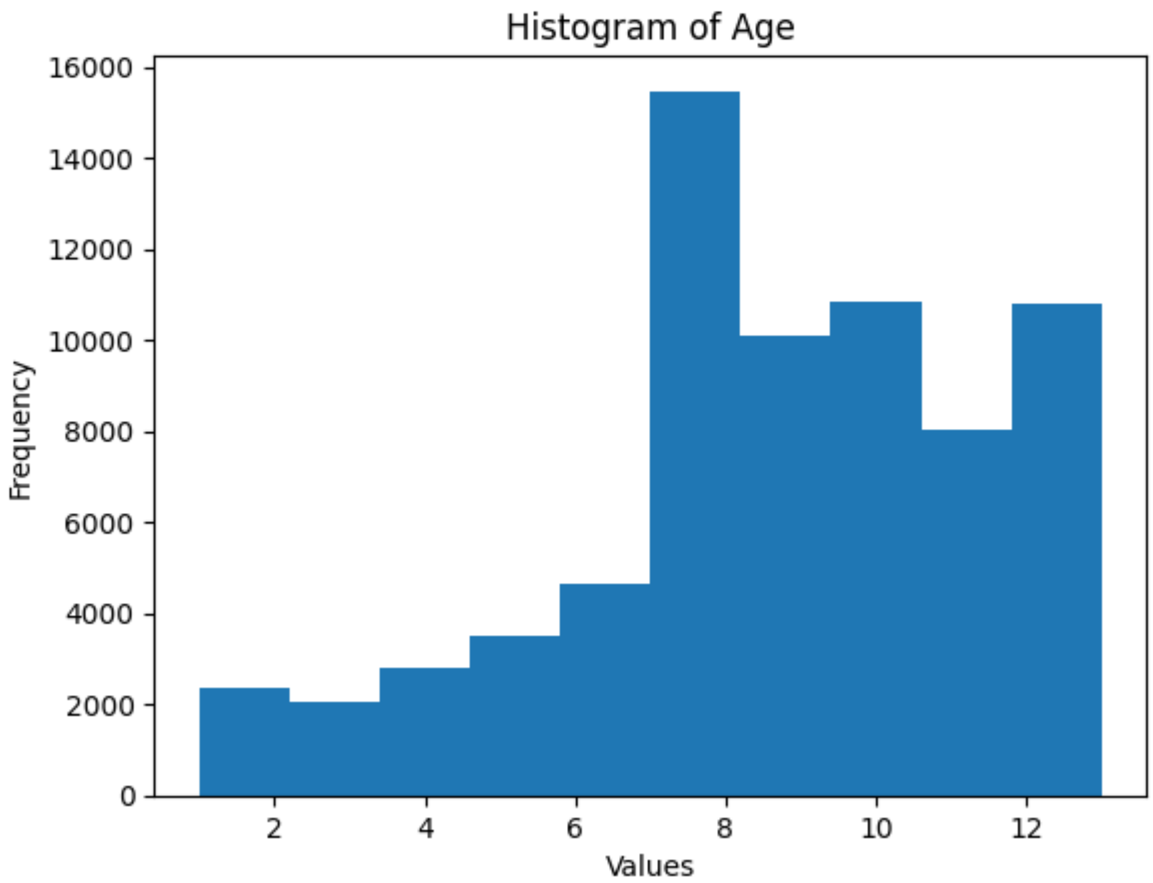
In [6]:

```
# generating histograms for each column in the dataframe
dist = diabetes_df.hist(figsize = (20,20), grid = False)
```

In order to visualize the age distribution better, we can generate a separate plot.

In [7]:
```
# histogram of age
plt.hist(diabetes_df['Age'])
plt.title('Histogram of Age')
plt.xlabel('Values')
plt.ylabel('Frequency')
```

Out[7]:
```
Text(0, 0.5, 'Frequency')
```

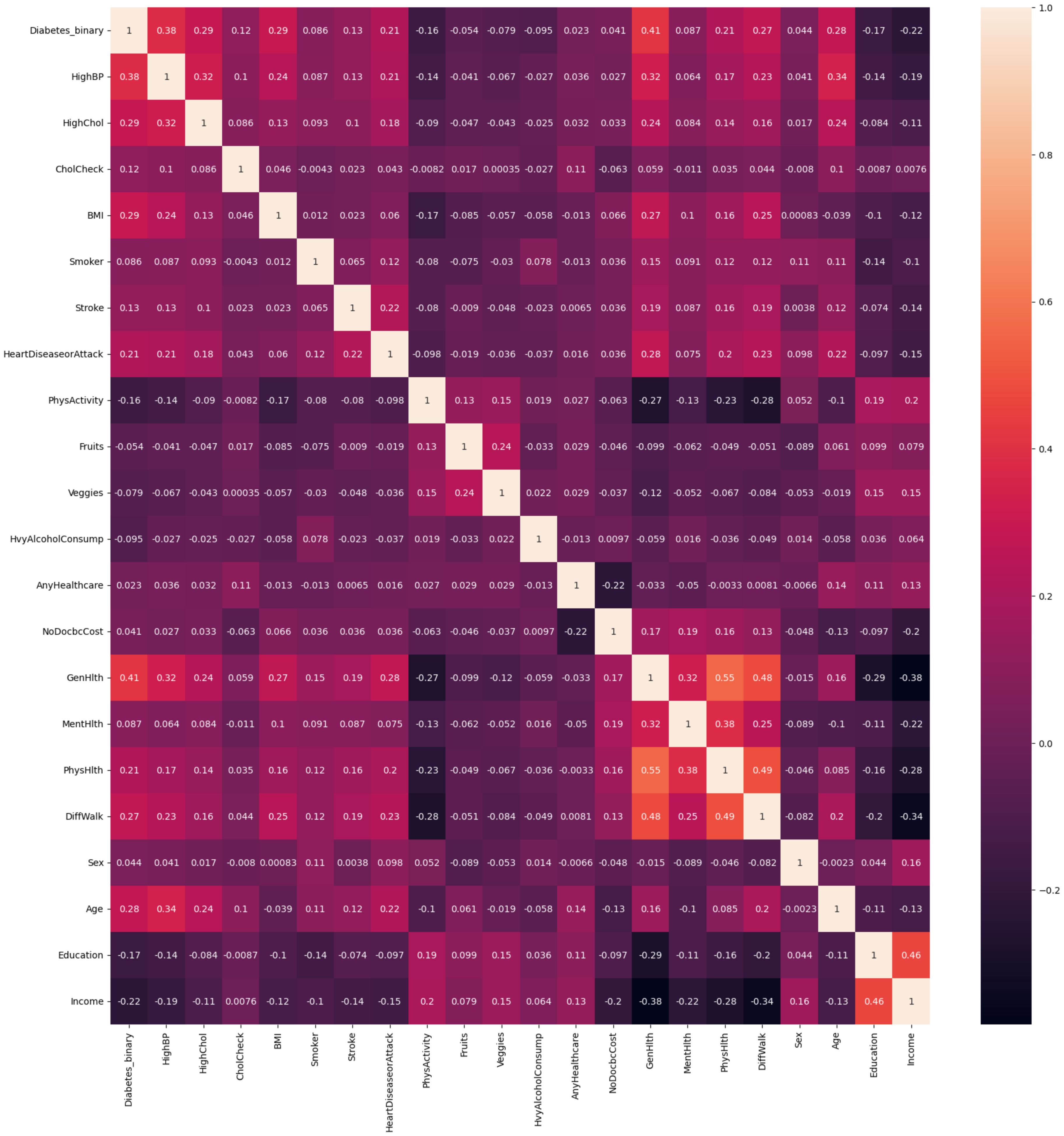


Based on the plots we can see that Diabetes, high blood pressure, smoking status, sex, and high cholesterol incidence are all roughly balanced in their distribution (are all binary variables), while other factors such as fruit/vegetable/alcohol consumption, heart disease, stroke, difficulty walking, and healthcare status are more imbalanced. The 50-54 age group was the most common in the dataset, with the data being more skewed toward people above age of 50. Higher income brackets were also more prevalent, with the >$75000 annual income range being the most frequent in the dataset. For this reason, higher education levels were also the most frequent in the dataset. The majority of people had access to some form of healthcare, and a minority did not have access to a doctor because of cost. BMI was distributed with the greatest incidence between the range of 20-40. Mental Health and Physical Health are distributed such that a larger number of people report little to no issues. GenHlth has a more normal distribution with the peak at 3 which indicates Good Health as compared to Excellent(1) and Poor (5) at the extremes

Bivariate Analysis

The relationships between feature variables was explored via correlation matrix while the relationship between the feature variables and target variables were explored through count plots and percentage plots.

In [8]:
```
# correlation matrix heatmap
# degree of correlation between each pair of factors
plt.figure(figsize = (20,20))
sns.heatmap(diabetes_df.corr(), annot = True)

# display heatmap
plt.show()
```



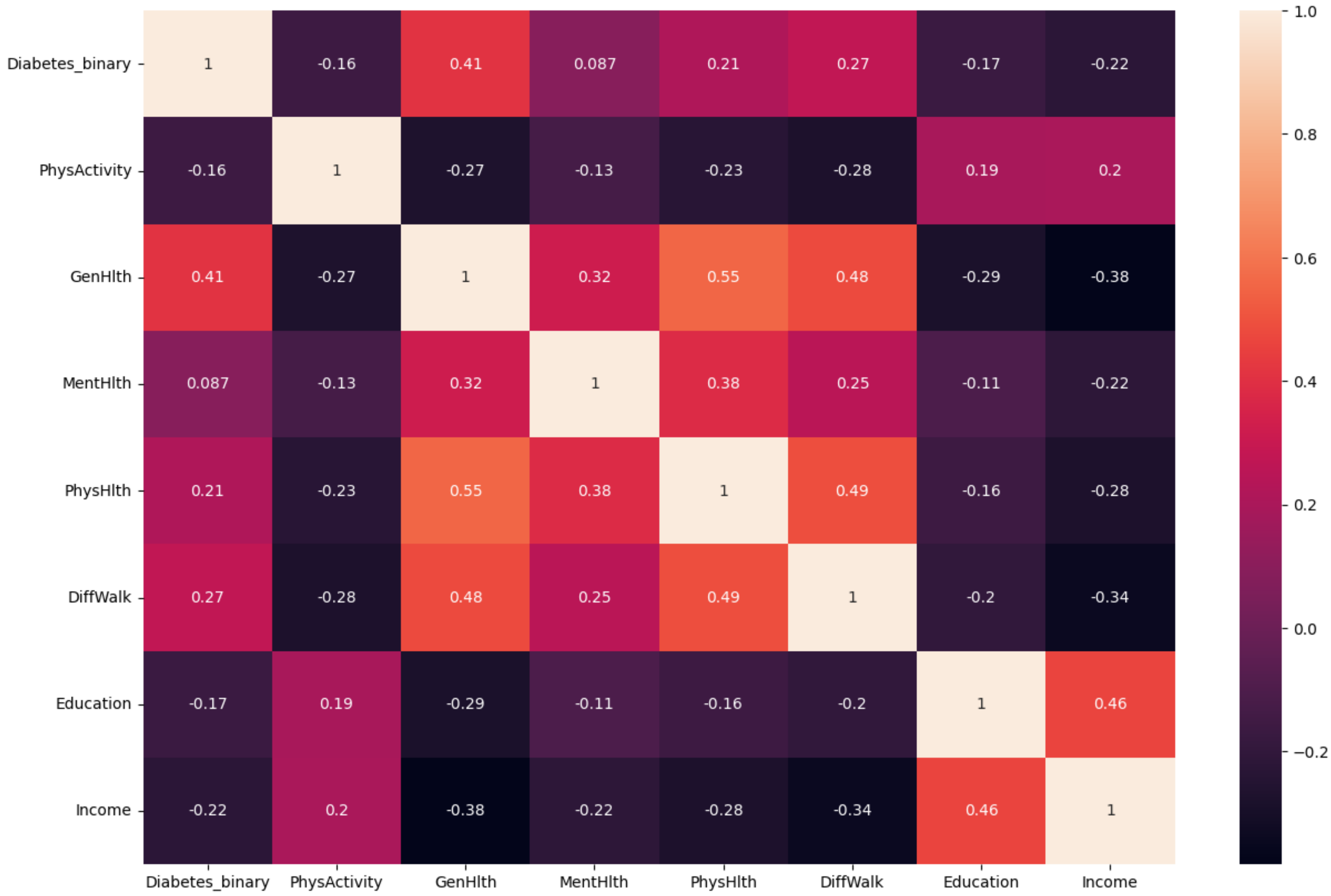
Based on the correlation matrix heatmap generated above it can be seen that the brighter the color the greater the positive correlation i.e. as one variable increases so does the other, and the darker the color the greater the negative correlation i.e. as one variable increases the other decreases.

Some of the variables that have the highest positive correlations are PhysHlth and GenHlth, DiffWalk and PhysHealth, DiffWalk and GenHlth, PhysHlth and MentHlth. Some variables that have the highest negative correlations are GenHlth and Income, DiffWalk and Income and Education and GenHlth

It is important to note that when interpreting these results the scales of the variables need to be considered.

Based on the above heatmap a more condensed heatmap was generated of the variables that have the strongest correlations (either positive or negative)

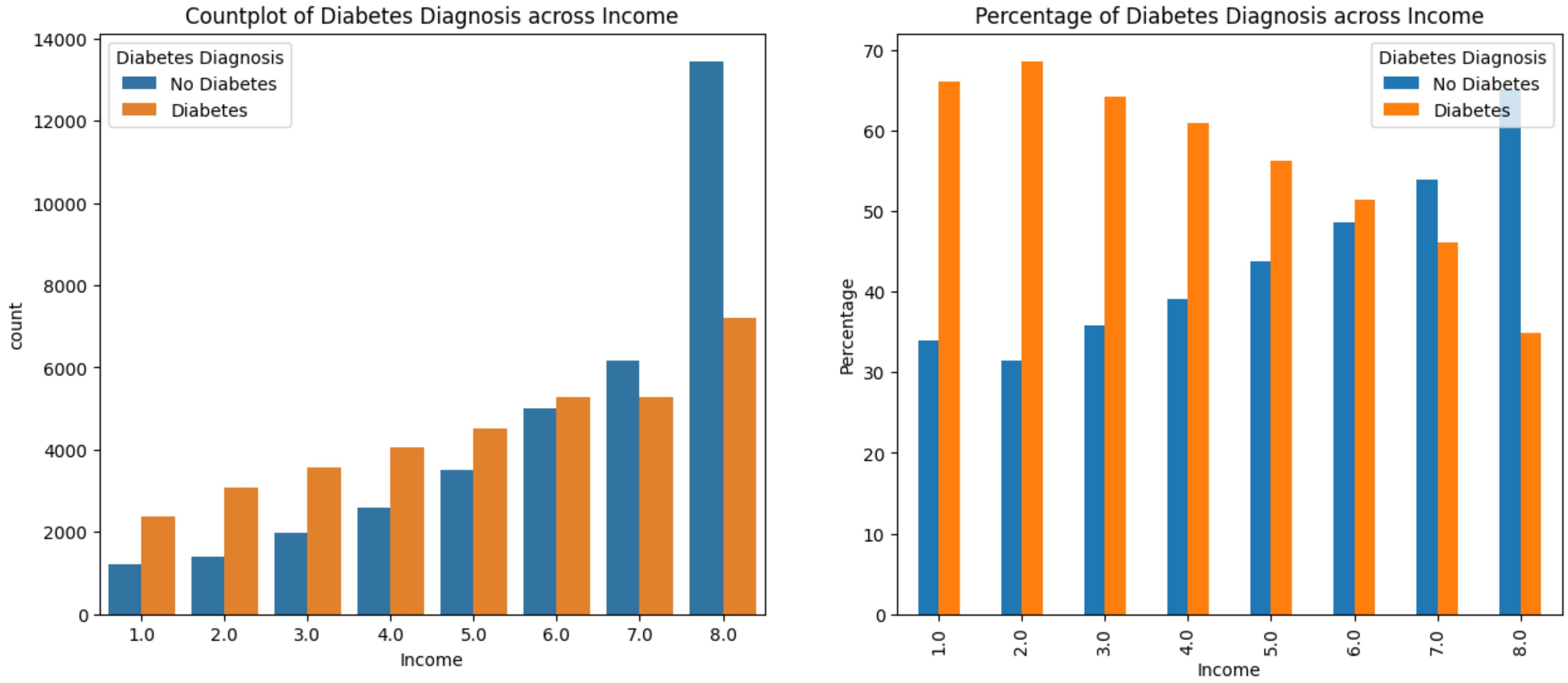
In [9]:
```
# condensed correlation matrix heatmap
# filter the dataframe to use only selected features in the correlation matrix
diabetes_filtered_df = diabetes_df.loc[:, ['Diabetes_binary', 'PhysActivity', 'GenHlth', 'MentHlth', 'PhysHlth', 'D

# generate heatmap
plt.figure(figsize = (15,10))
sns.heatmap(diabetes_filtered_df.corr(), annot = True)

# display heatmap
plt.show()
```



Based on preliminary analysis (wherein countplots and percent plots for diabetes diagnosis against all factors was generated) and PCA. Only a few factors were explored in greater detail.

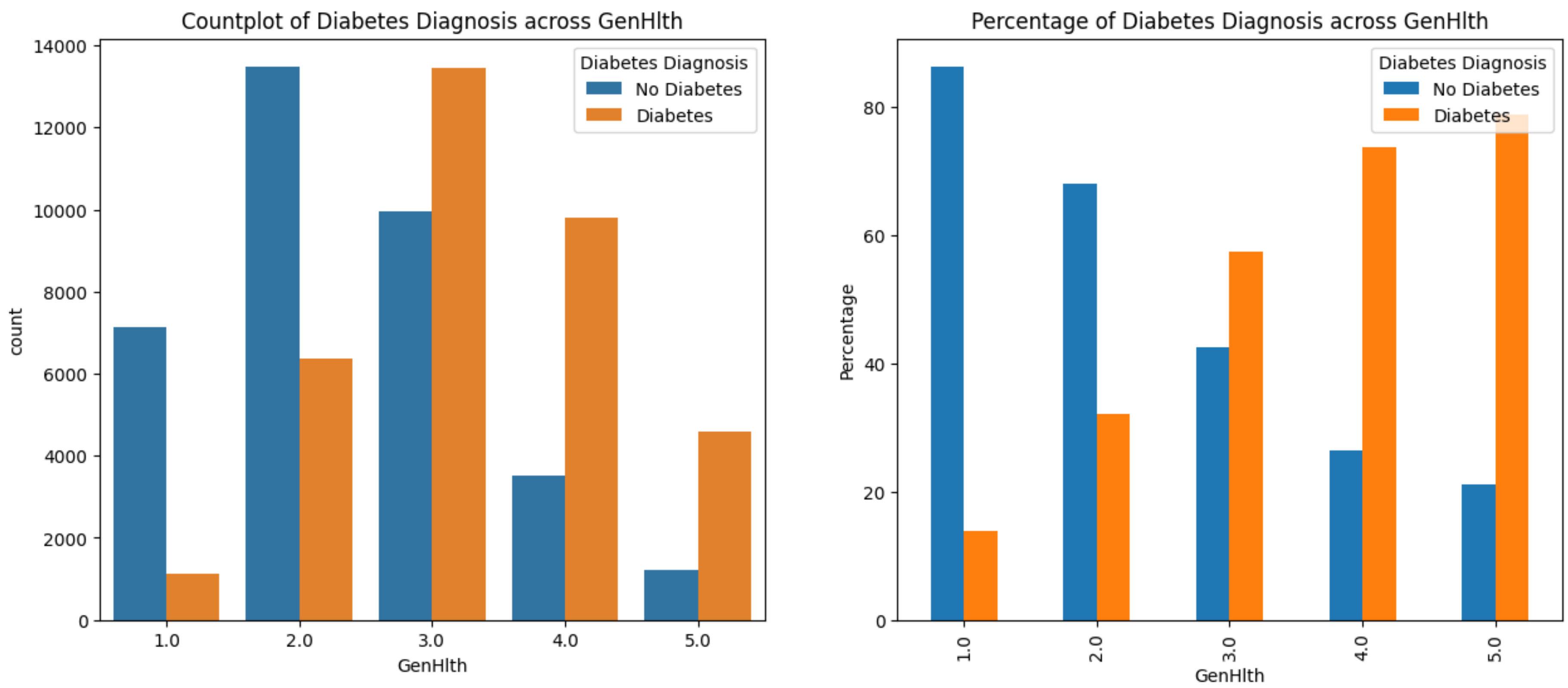
Generating countplot and percent plots for Distribution of Diabetes across Income

In [10]:
```
# generating figure with subplots
fig, axes = plt.subplots(1,2, figsize = (15,6))

# subplot 1
# countplot for diabetes diagnosis across income
sns.countplot(x = 'Income', hue = 'Diabetes_binary', data = diabetes_df, ax = axes[0])
# setting axis labels, plot title and legend
axes[0].set_xlabel('Income')
axes[0].set_title('Countplot of Diabetes Diagnosis across Income')
axes[0].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'])

# subplot 2
# converting the normalized value counts in income to percentages
income_percent = diabetes_df.groupby('Income')['Diabetes_binary'].value_counts(normalize=True).unstack() * 100
# percentage plot for Diabetes Diagnosis across Income
income_percent.plot(kind='bar', ax = axes[1])
# setting axis labels, plot title and legend
axes[1].set_title('Percentage of Diabetes Diagnosis across Income')
axes[1].set_ylabel('Percentage')
axes[1].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'], loc='upper right')

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



Generating countplot and percent plots for Distribution of Diabetes across General Health (GenHlth)

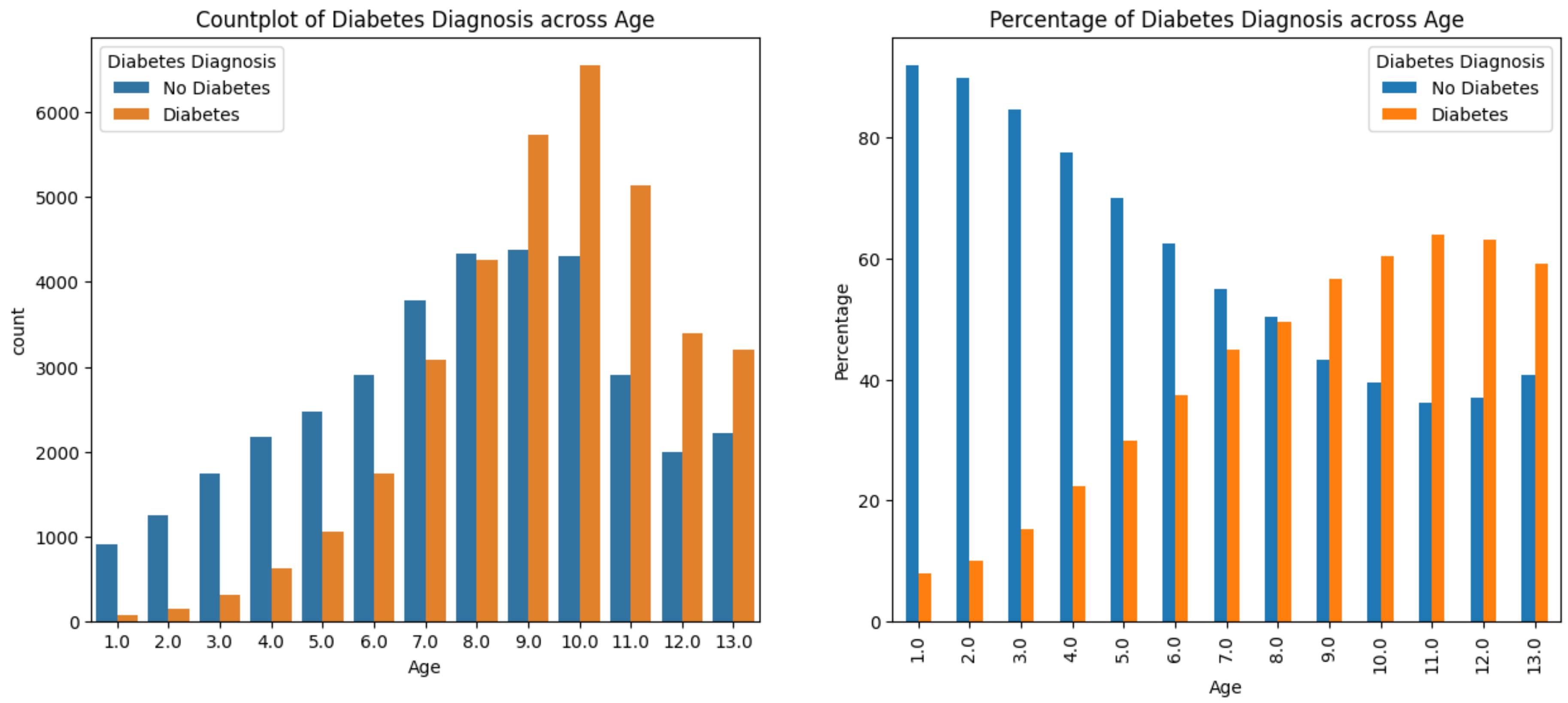
In [11]:
```
# generating figure with subplots
fig, axes = plt.subplots(1,2, figsize = (15,6))

# subplot 1
# countplot  for diabetes diagnosis across GenHlth
sns.countplot(x = 'GenHlth', hue = 'Diabetes_binary', data = diabetes_df, ax = axes[0])
# setting axis labels, plot title and legend
axes[0].set_xlabel('GenHlth')
axes[0].set_title('Countplot of Diabetes Diagnosis across GenHlth')
axes[0].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'])

# subplot 2
# converting the normalized value counts in GenHlth to percentages
income_percent = diabetes_df.groupby('GenHlth')['Diabetes_binary'].value_counts(normalize=True).unstack() * 100
# Percent Plot for diabetes diagnosis across General Health
income_percent.plot(kind='bar', ax = axes[1])
# setting axis labels, plot title and legend
axes[1].set_title('Percentage of Diabetes Diagnosis across GenHlth')
axes[1].set_ylabel('Percentage')
axes[1].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'], loc='upper right')

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



Generating countplot and percent plots for Distribution of Diabetes across Age

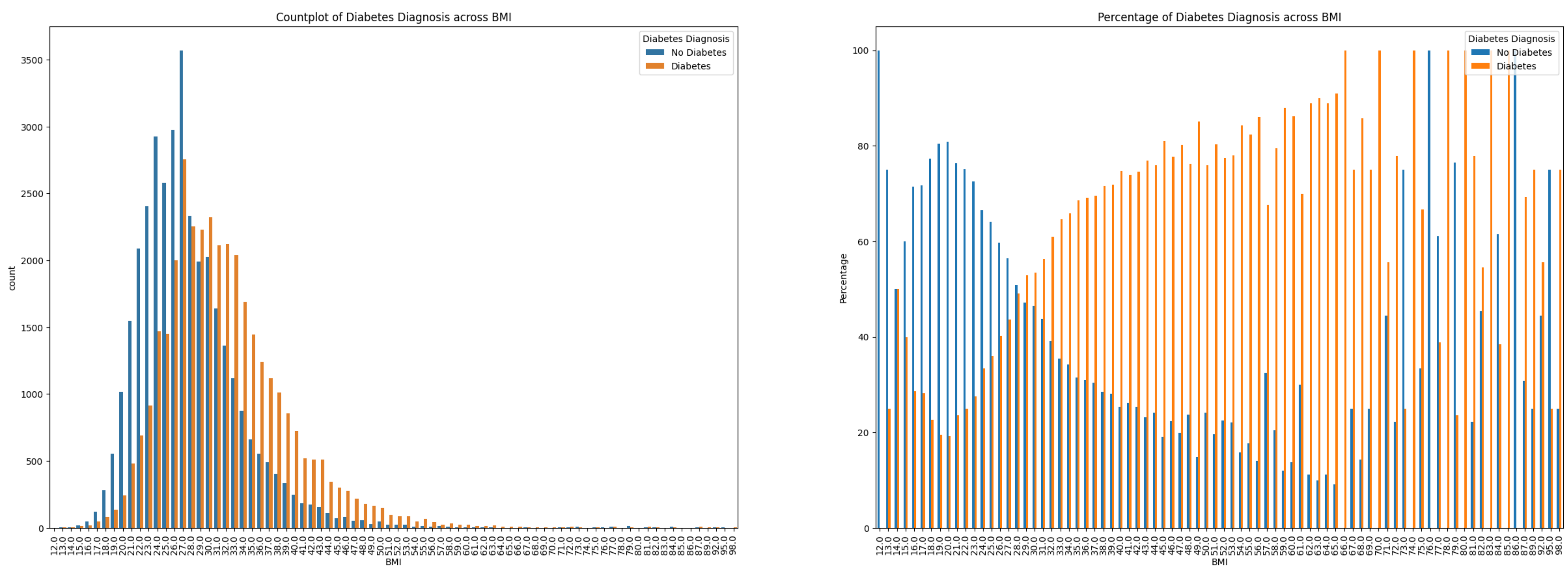
In [12]:
```
# generating figure with subplots
fig, axes = plt.subplots(1,2, figsize = (15,6))

# subplot 1
# countplot for diabetes diagnosis across Age
sns.countplot(x = 'Age', hue = 'Diabetes_binary', data = diabetes_df, ax = axes[0])
# setting axis labels, plot title and legend
axes[0].set_xlabel('Age')
axes[0].set_title('Countplot of Diabetes Diagnosis across Age')
```

```
axes[0].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'])

# subplot 2
# converting the normalized value counts in Age to percentages
income_percent = diabetes_df.groupby('Age')['Diabetes_binary'].value_counts(normalize=True).unstack() * 100
# Percent Plot for diabetes diagnosis across Age
income_percent.plot(kind='bar', ax = axes[1])
# setting axis labels, plot title and legend
axes[1].set_title('Percentage of Diabetes Diagnosis across Age')
axes[1].set_ylabel('Percentage')
axes[1].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'], loc='upper right')

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



Generating countplot and percent plots for Distribution of Diabetes across BMI

In [13]:
```
# generating figure with subplots
fig, axes = plt.subplots(1,2, figsize = (30,10))

# subplot 1
# countplot for diabetes diagnosis across BMI
sns.countplot(x = 'BMI', hue = 'Diabetes_binary', data = diabetes_df, ax = axes[0])
# setting axis labels, xticks, plot title and legend
axes[0].set_xlabel('BMI')
axes[0].set_xticklabels(axes[0].get_xticklabels(), rotation=90)
axes[0].set_title('Countplot of Diabetes Diagnosis across BMI')
axes[0].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'])

# subplot 2
# converting the normalized value counts in BMI to percentages
income_percent = diabetes_df.groupby('BMI')['Diabetes_binary'].value_counts(normalize=True).unstack() * 100
# Percent Plot for diabetes diagnosis across BMI
income_percent.plot(kind='bar', ax = axes[1])
# setting axis labels, plot title and legend
axes[1].set_title('Percentage of Diabetes Diagnosis across BMI')
axes[1].set_ylabel('Percentage')
axes[1].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'], loc='upper right')

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

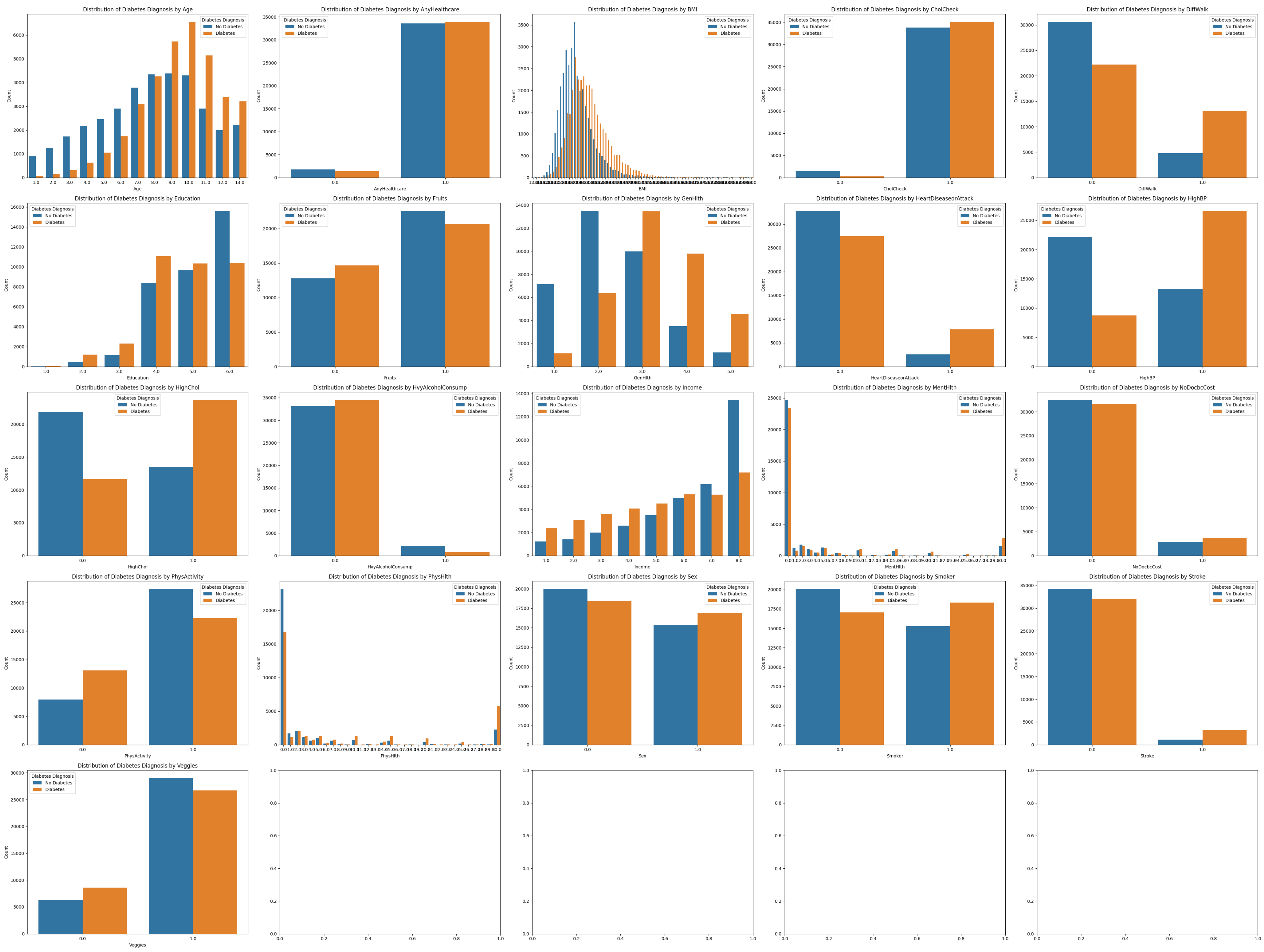


Additional Material

The count and percent plots for all factors from which a the most relevant were selected.

The Countplots for Diabetes Diagnosis across all factors

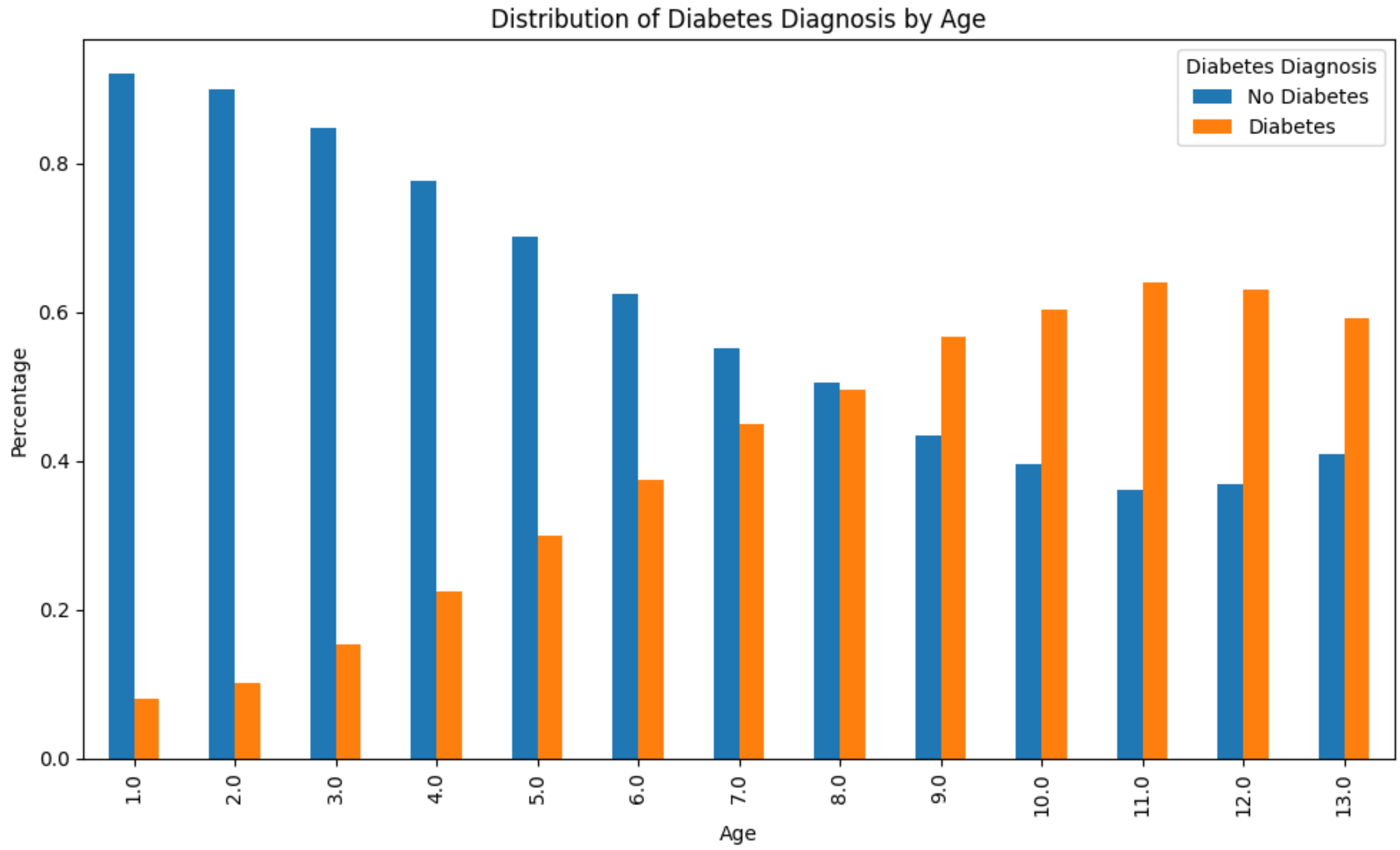
In [14]:
```
# selecting only the factors i.e. predictor columns
predictor_columns = diabetes_df.columns.difference(['Diabetes_binary'])

# generating figure with subplots for all factors (as a large image with columns and rows)
fig, axes = plt.subplots(nrows=5, ncols=5, figsize=(40,30))

# Loop through each predictor column and create a count plot
for i, column in enumerate(predictor_columns):
    ax = axes.flatten()[i]
    sns.countplot(x=column, hue='Diabetes_binary', data=diabetes_df, ax=ax)
    # setting axis labels, plot title and legend
    ax.set_title(f'Distribution of Diabetes Diagnosis by {column}')
    ax.set_xlabel(column)
    ax.set_ylabel('Count')
    ax.legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'])

# Adjust layout
plt.tight_layout()
# Show plot
plt.show()
```

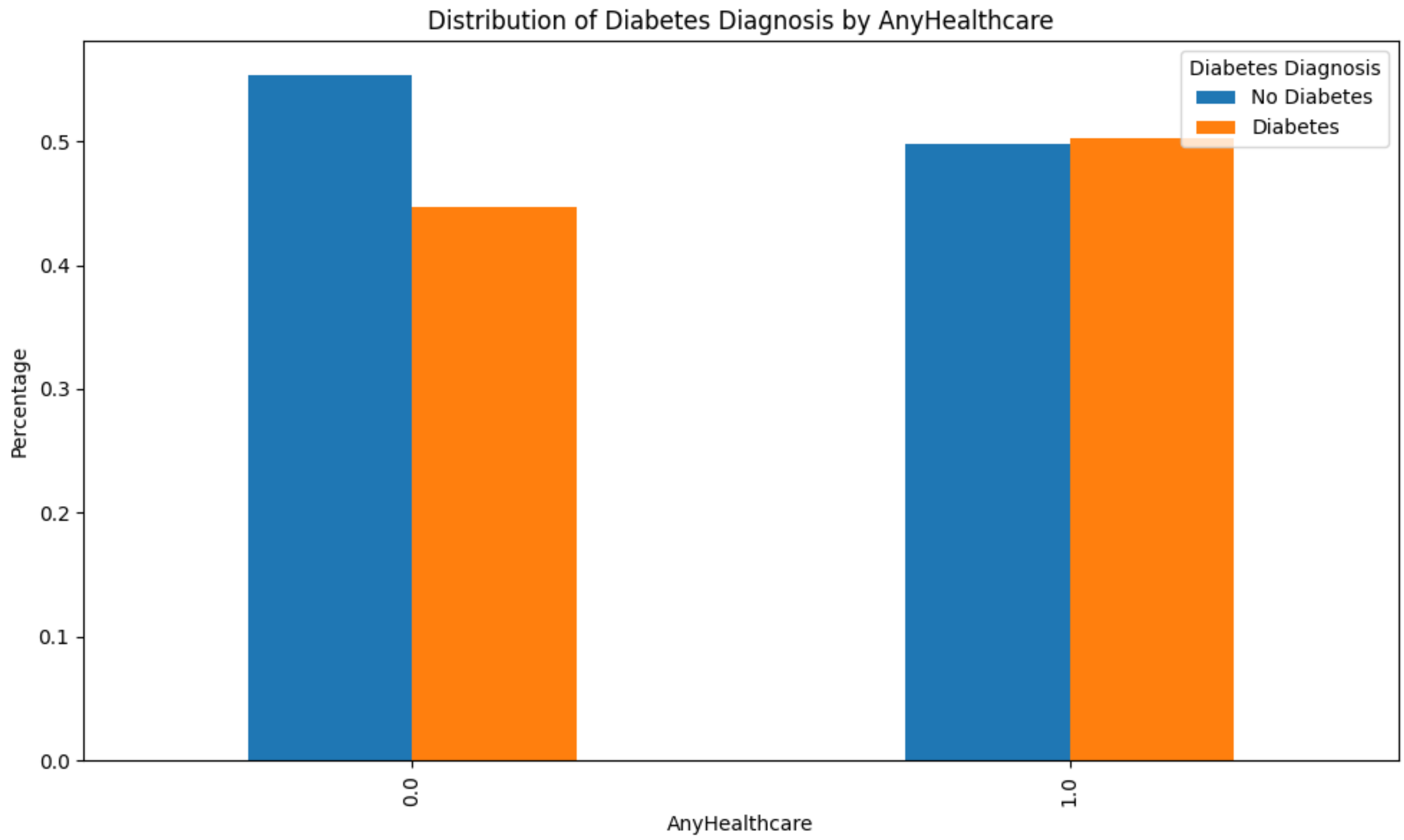


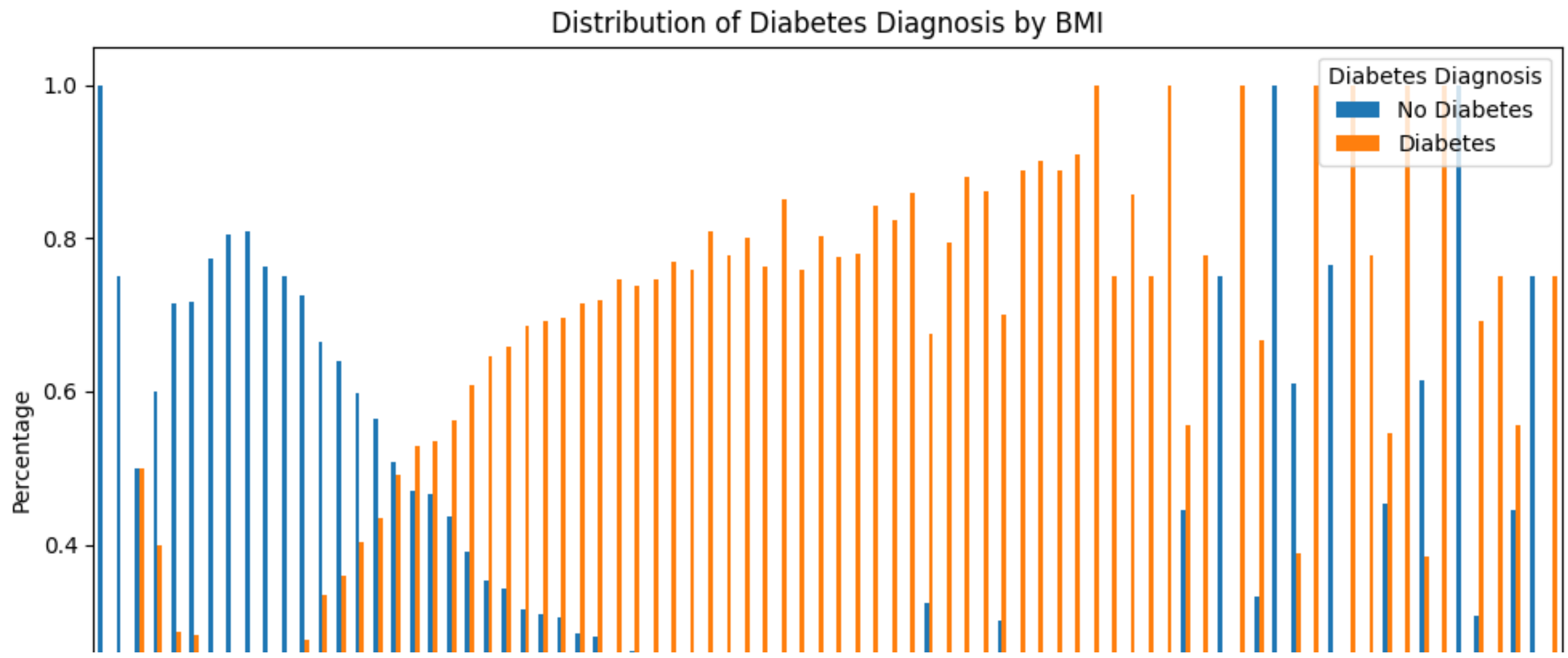
The percent plots for diabetes diagnosis across all factors

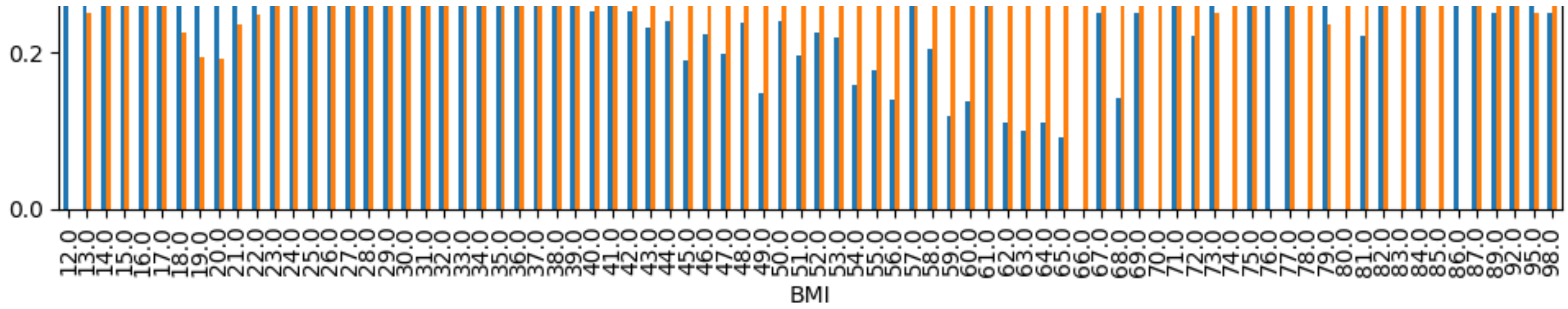
In [15]:
```
# generating percentage plots for diabetes diagnosis across all factors

# selecting only the factors i.e. predictor columns
predictor_columns = diabetes_df.columns.difference(['Diabetes_binary'])

# Set up subplots as one column
fig, axes = plt.subplots(nrows=len(predictor_columns), ncols=1, figsize=(10, 6 * len(predictor_columns)))

# Loop through each predictor column and create a percentage plot
for i, column in enumerate(predictor_columns):
    # Calculate percentage for each category
    percentage_data = diabetes_df.groupby(column)['Diabetes_binary'].value_counts(normalize=True).unstack()

    # Plot the percentage plot
    percentage_data.plot(kind='bar', ax=axes[i])
```

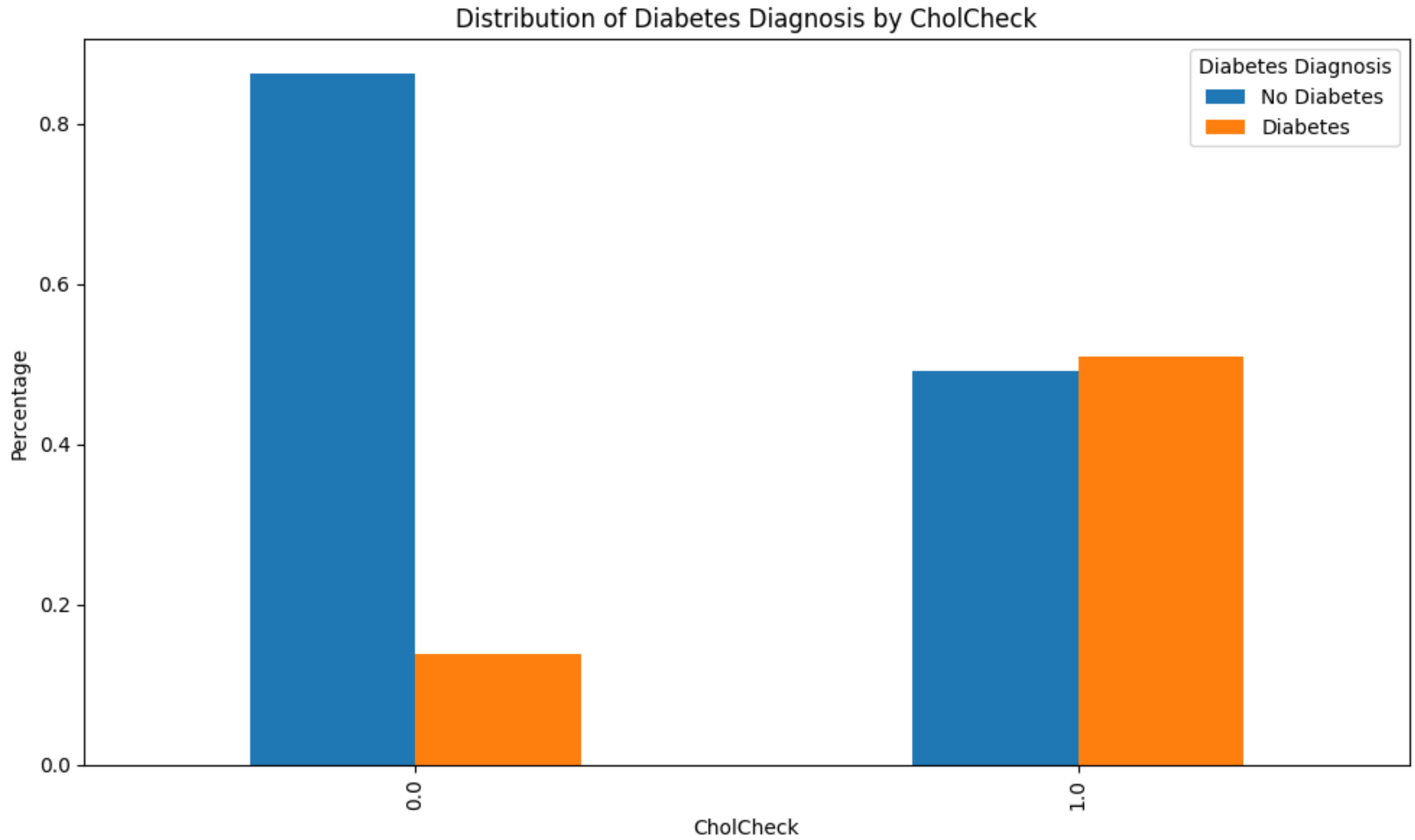
```
    # setting axis labels, plot title and legend
    axes[i].set_title(f'Distribution of Diabetes Diagnosis by {column}')
    axes[i].set_xlabel(column)
    axes[i].set_ylabel('Percentage')
    axes[i].legend(title='Diabetes Diagnosis', labels=['No Diabetes', 'Diabetes'], loc='upper right')

# Adjust layout
plt.tight_layout()
plt.show()
```

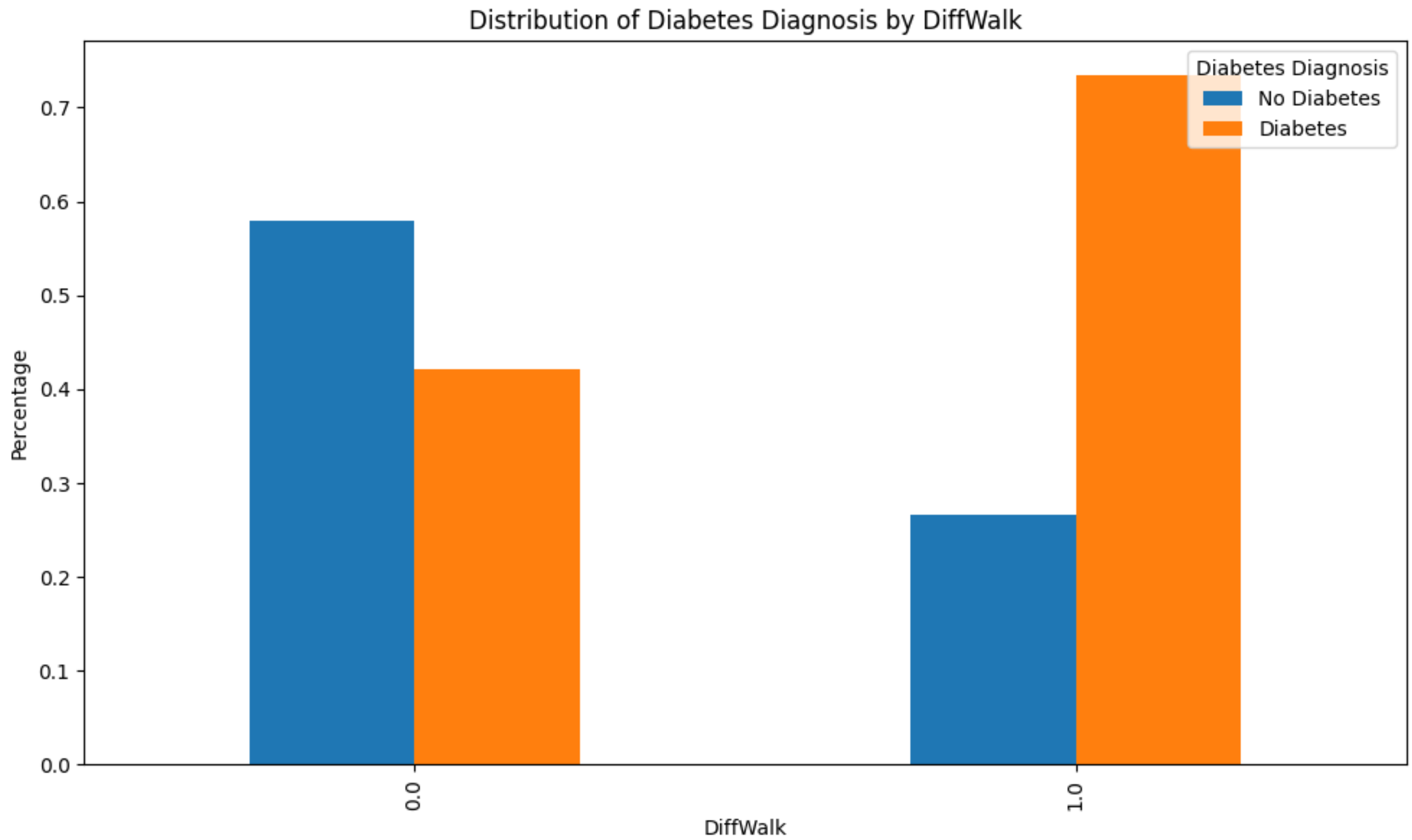

Distribution of Diabetes Diagnosis by Age
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.0
0.2
0.4
0.6
0.8
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0
13.0
Age

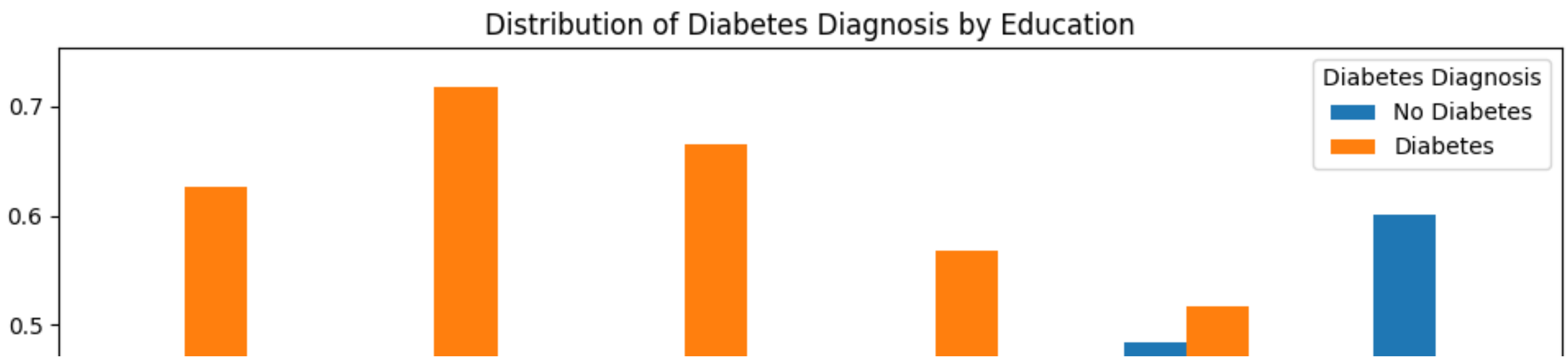

Distribution of Diabetes Diagnosis by AnyHealthcare
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.0
0.1
0.2
0.3
0.4
0.5
0.0
1.0
AnyHealthcare

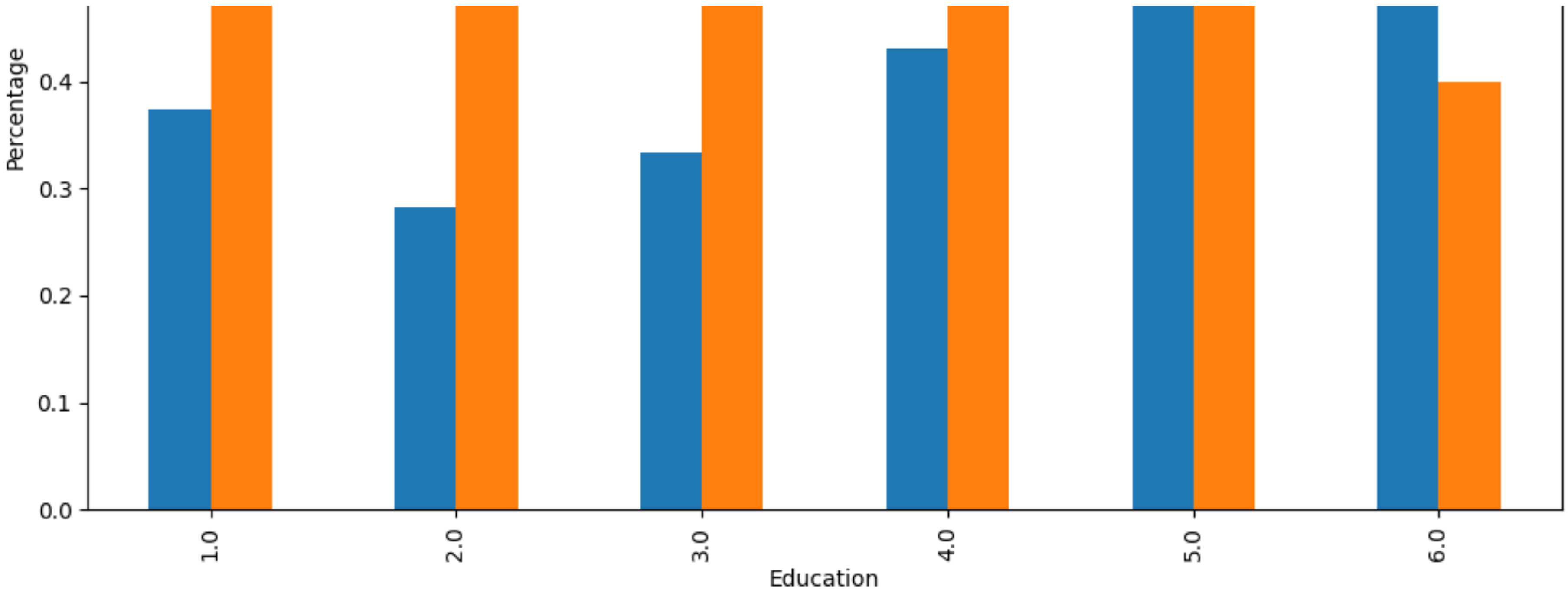

Distribution of Diabetes Diagnosis by BMI
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.4
0.6
0.8
1.0


0.2
0.0
BMI

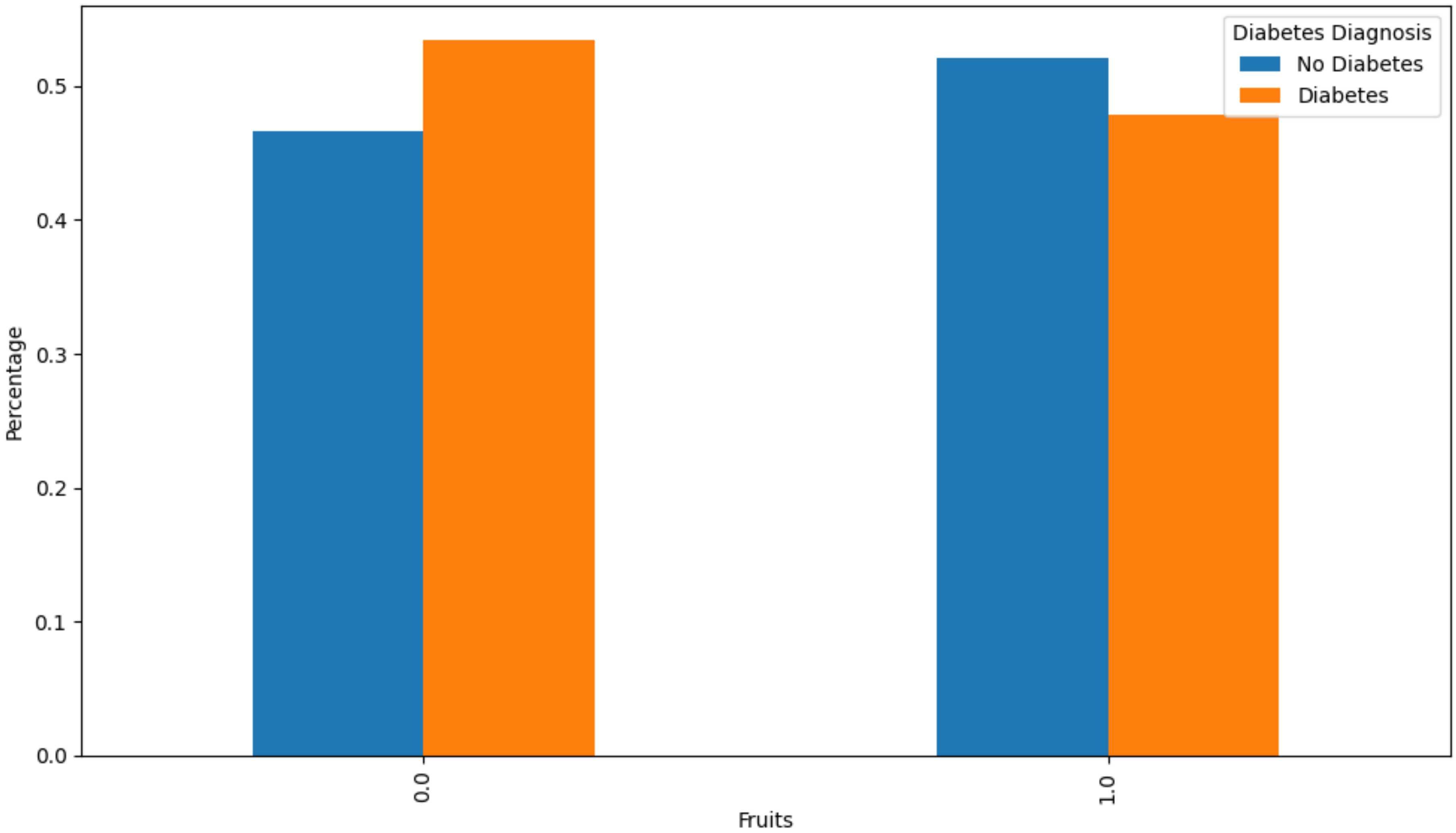

Distribution of Diabetes Diagnosis by CholCheck
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.0
0.2
0.4
0.6
0.8
0.0
1.0
CholCheck


Distribution of Diabetes Diagnosis by DiffWalk
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
1.0
DiffWalk

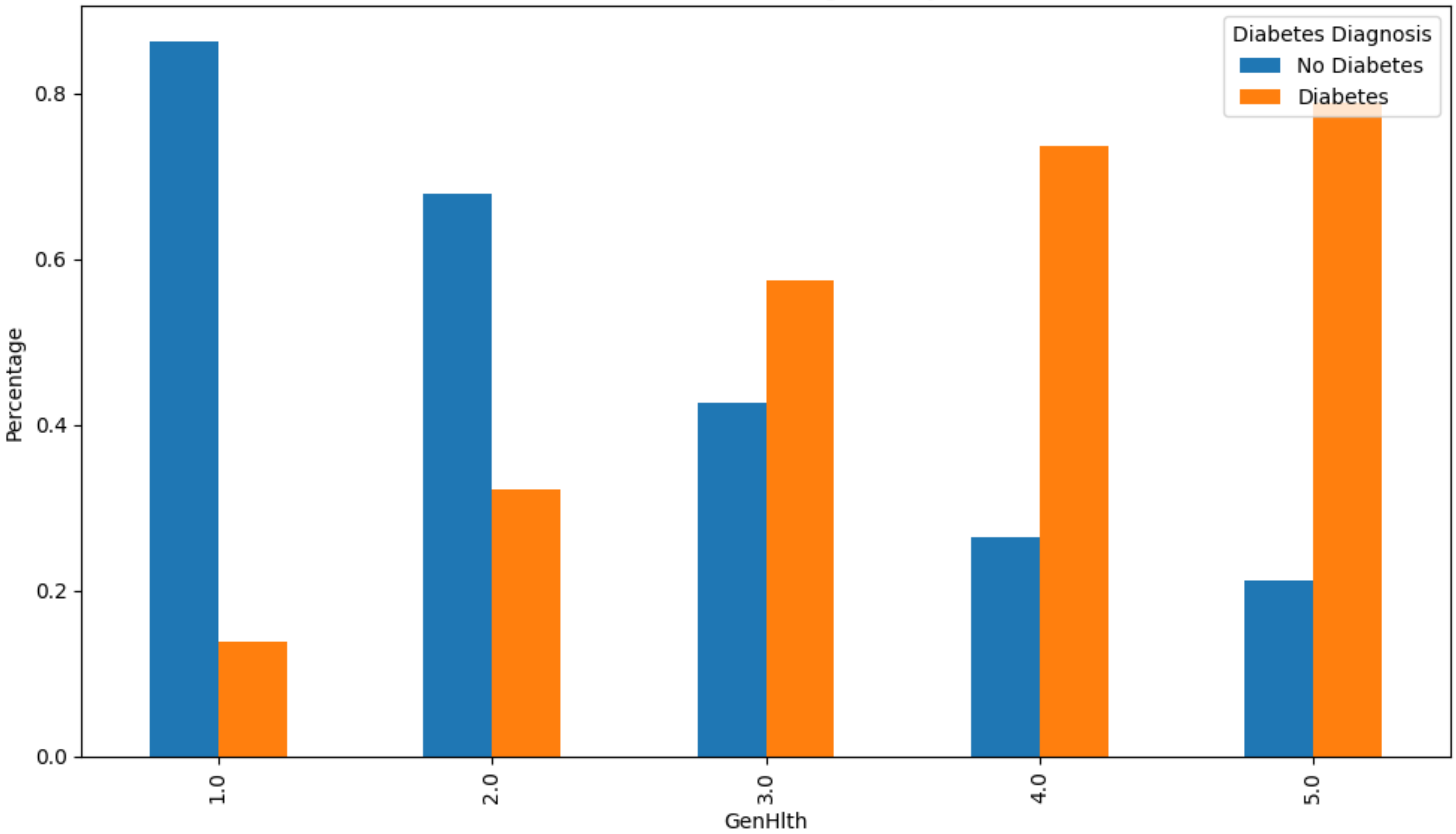

Distribution of Diabetes Diagnosis by Education
Diabetes Diagnosis
No Diabetes
Diabetes
0.5
0.6
0.7



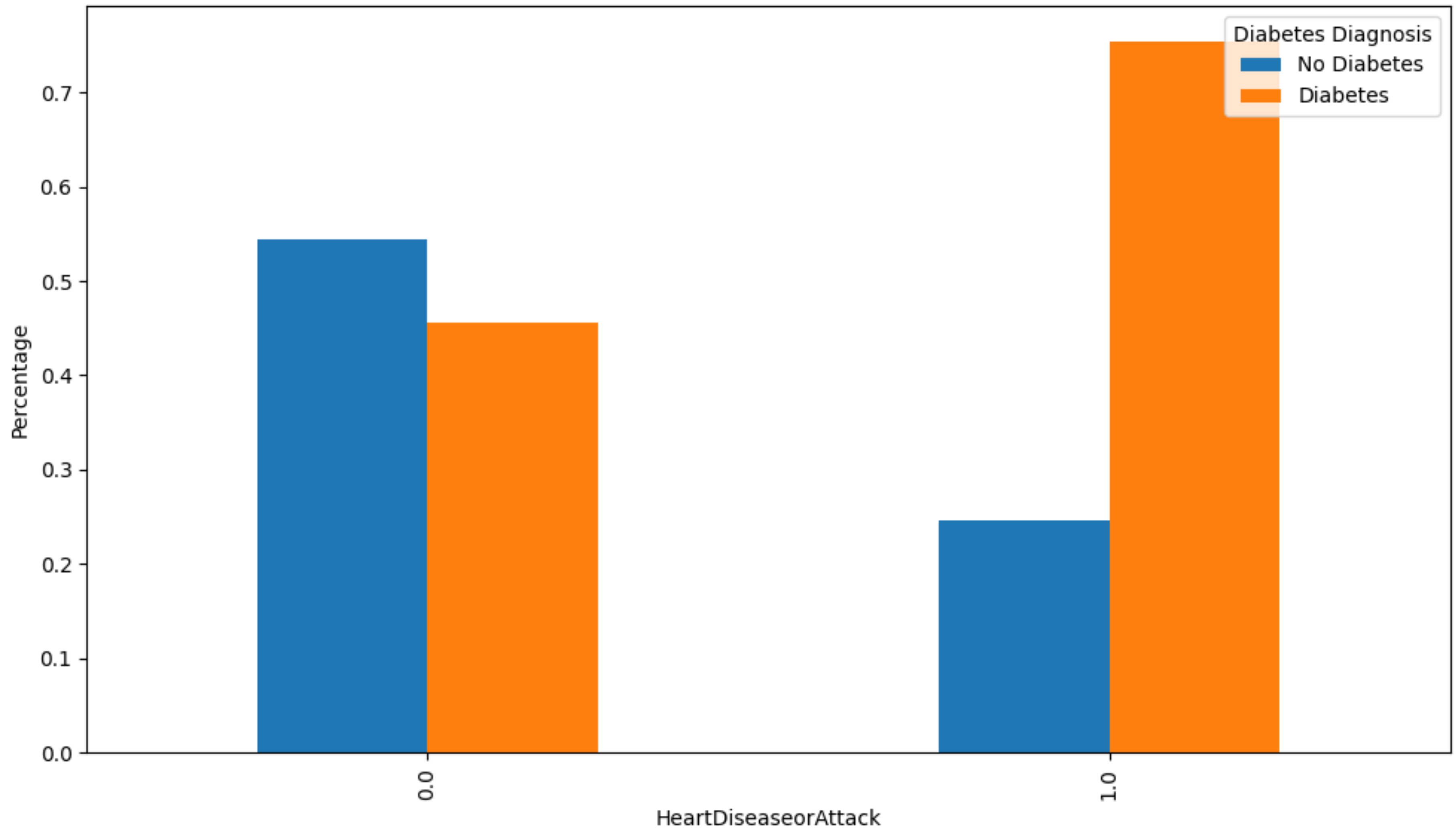
Distribution of Diabetes Diagnosis by Fruits



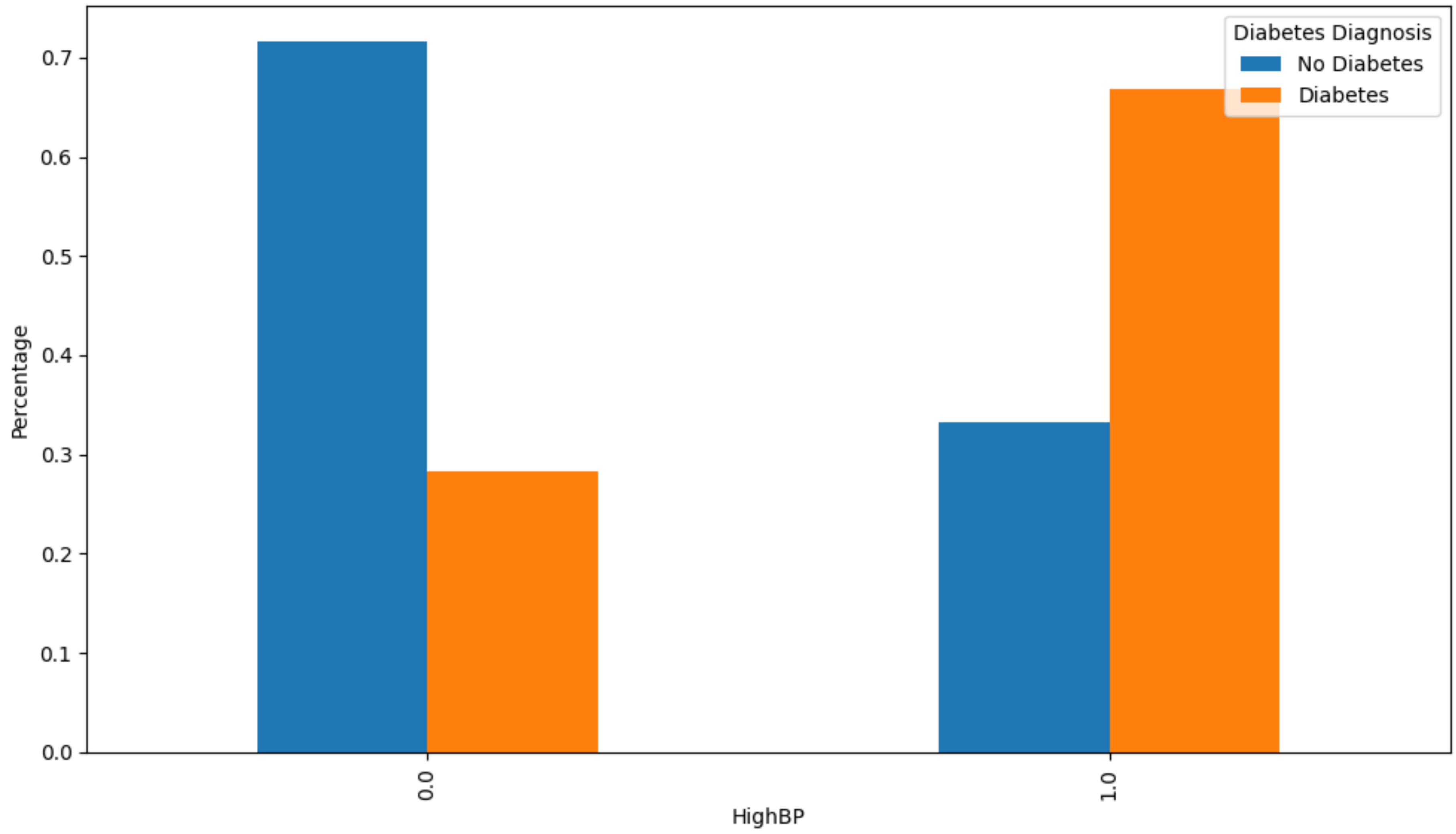
Distribution of Diabetes Diagnosis by GenHlth



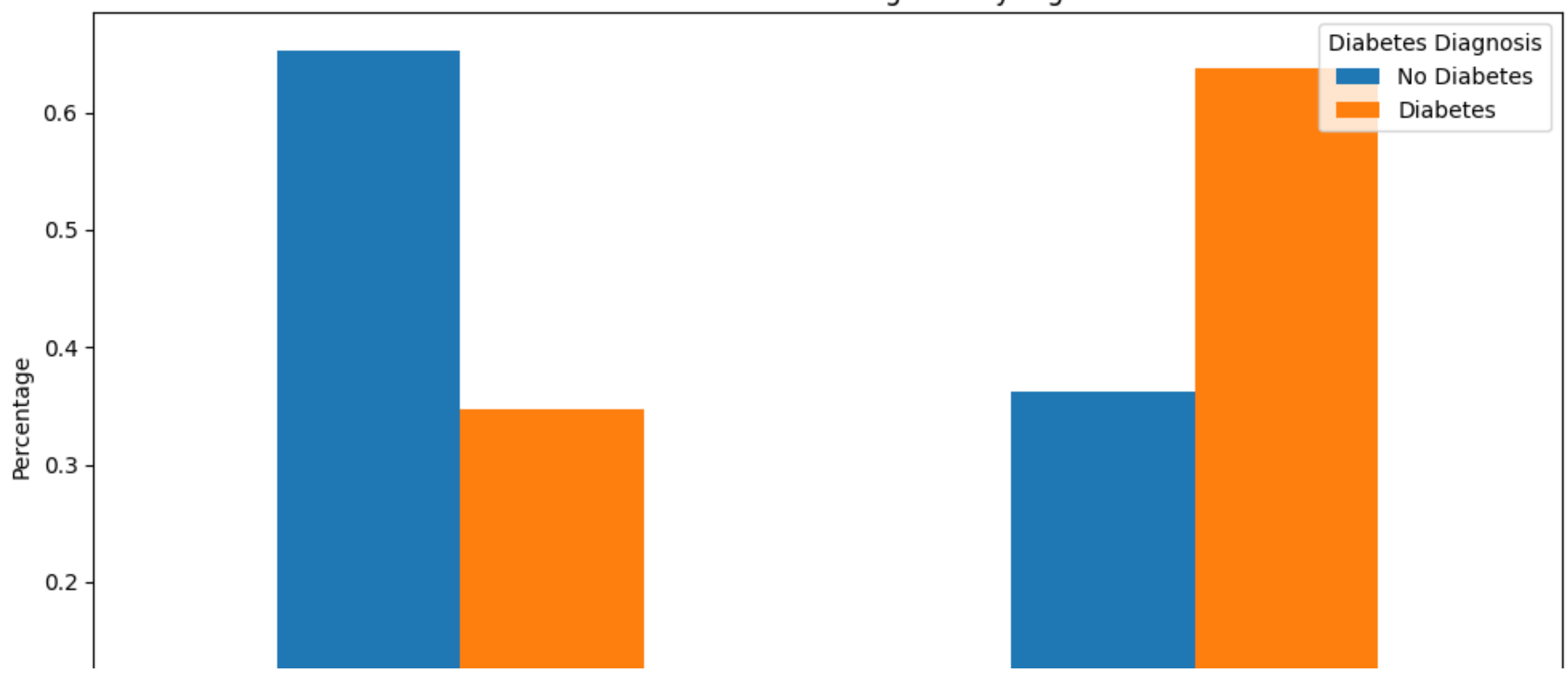
Distribution of Diabetes Diagnosis by HeartDiseaseorAttack

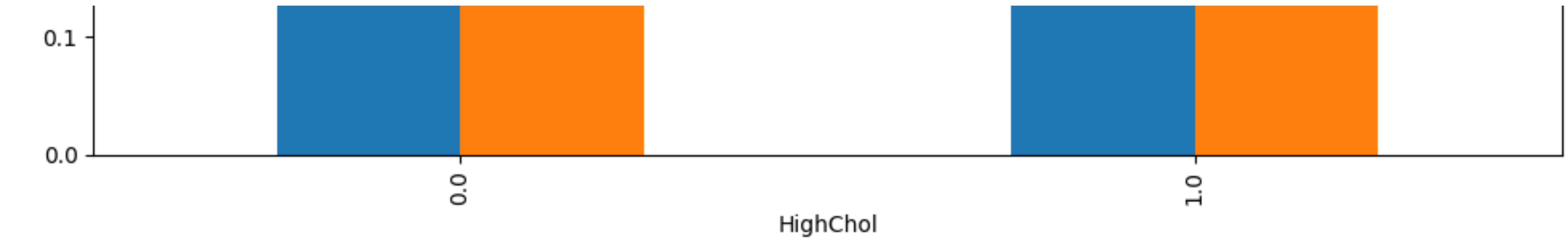


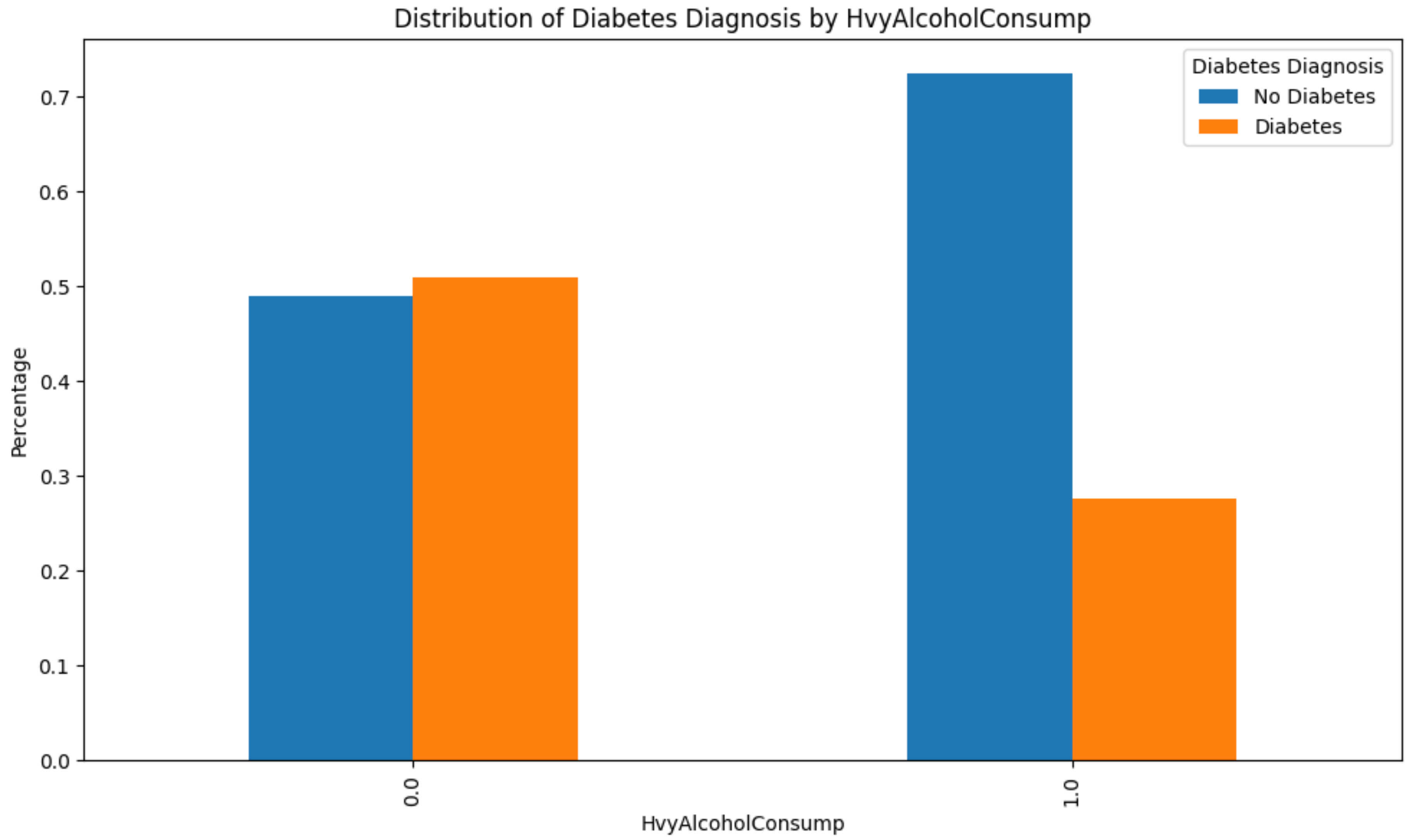
Distribution of Diabetes Diagnosis by HighBP

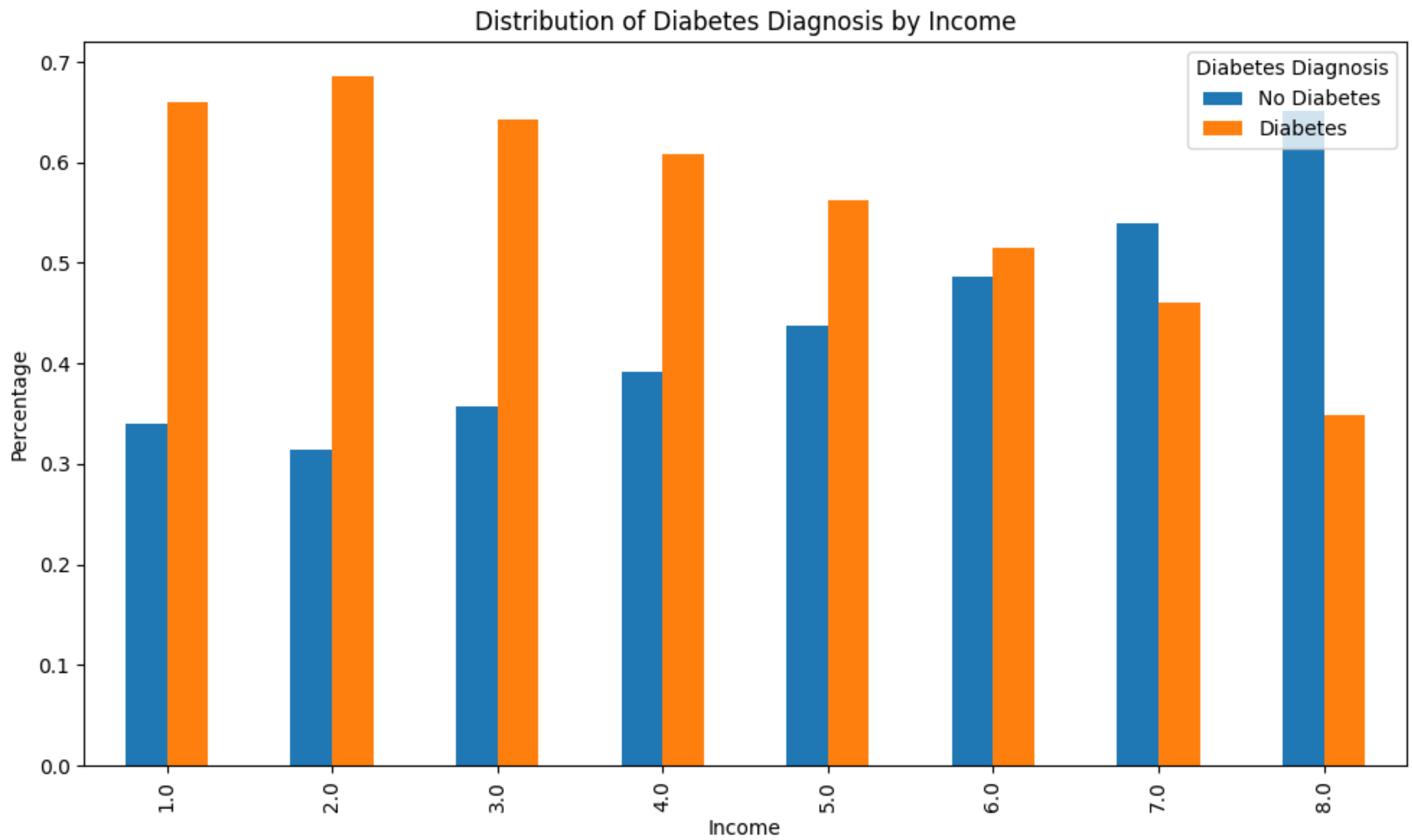


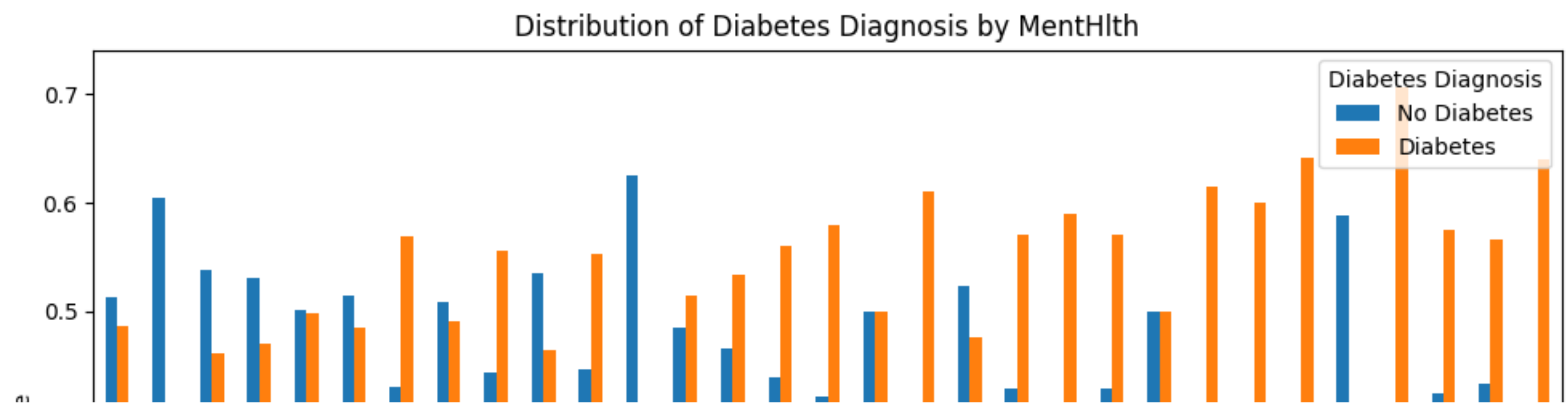
Distribution of Diabetes Diagnosis by HighChol

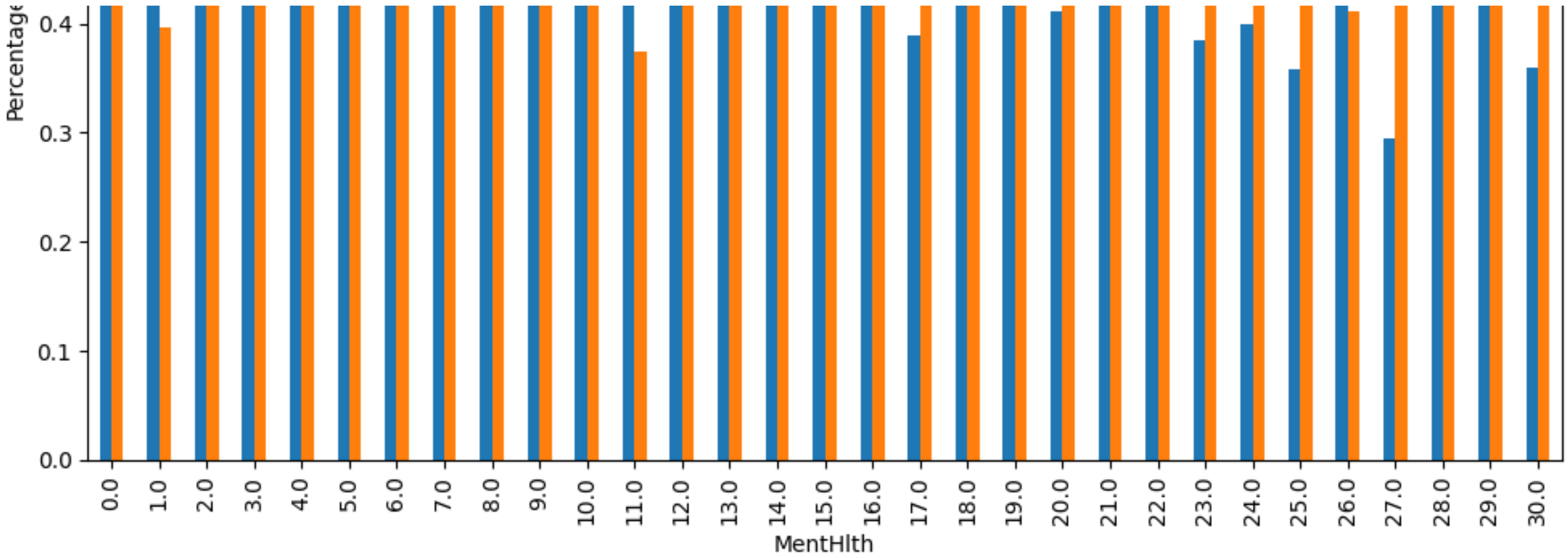



0.1
0.0
0.0
1.0
HighChol

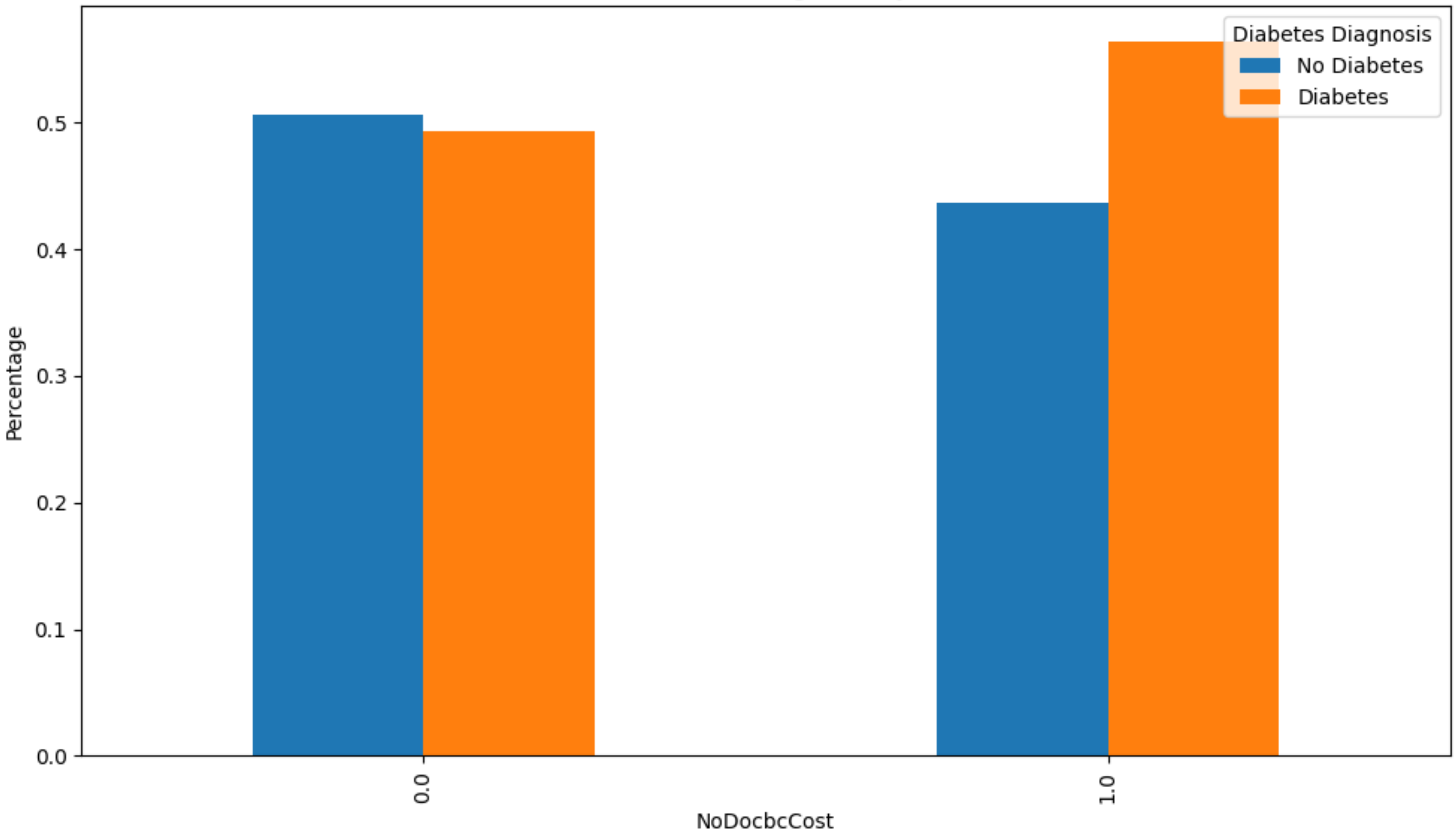

Distribution of Diabetes Diagnosis by HvyAlcoholConsump
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
1.0
HvyAlcoholConsump


Distribution of Diabetes Diagnosis by Income
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
Income

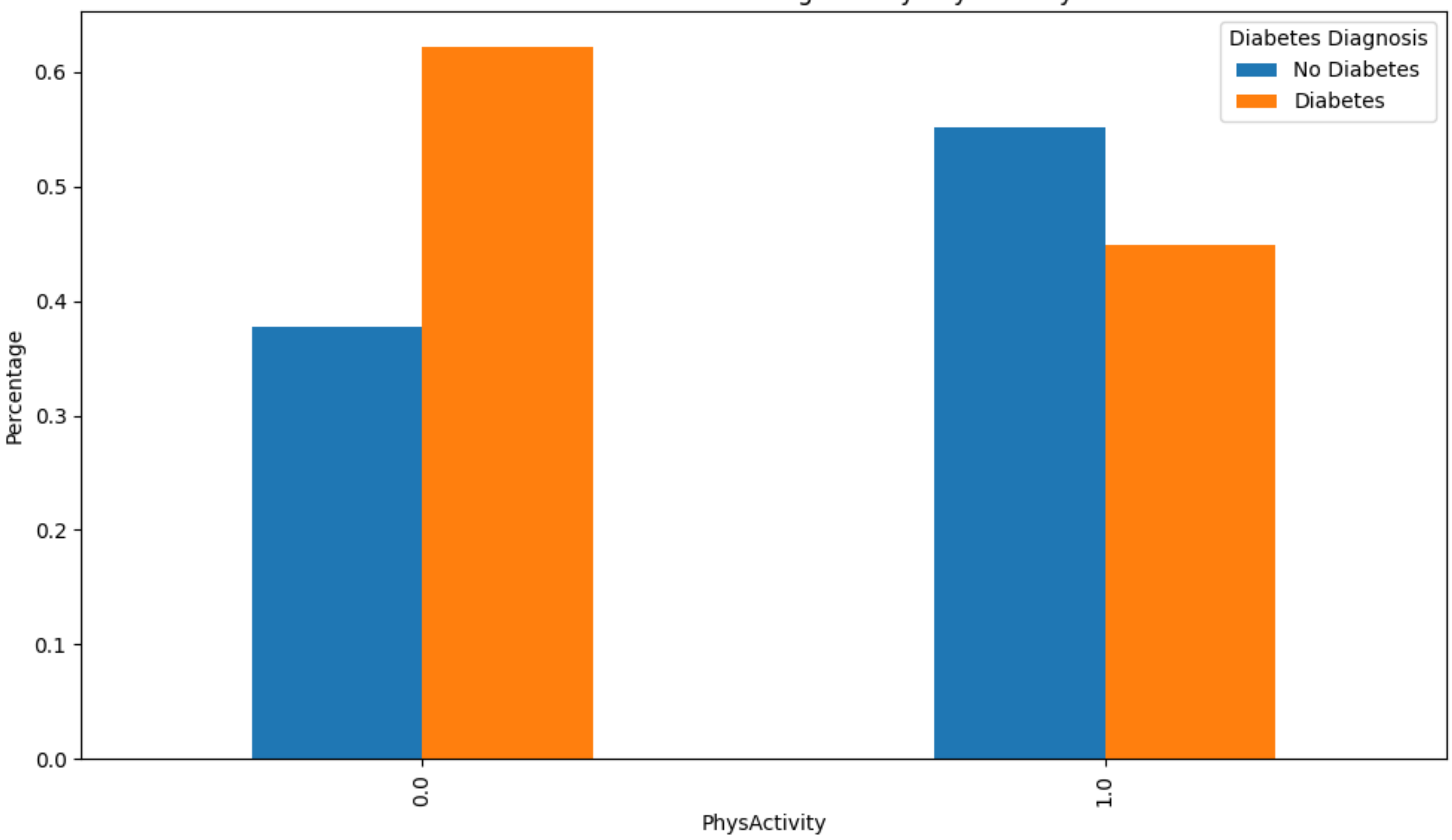

Distribution of Diabetes Diagnosis by MentHlth
Diabetes Diagnosis
No Diabetes
Diabetes
0.7
0.6
0.5



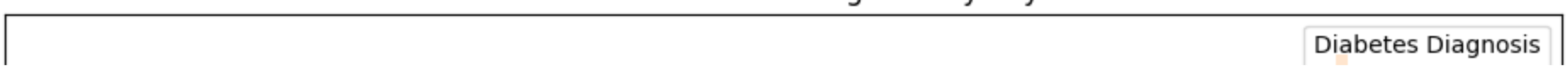
Distribution of Diabetes Diagnosis by NoDocbcCost

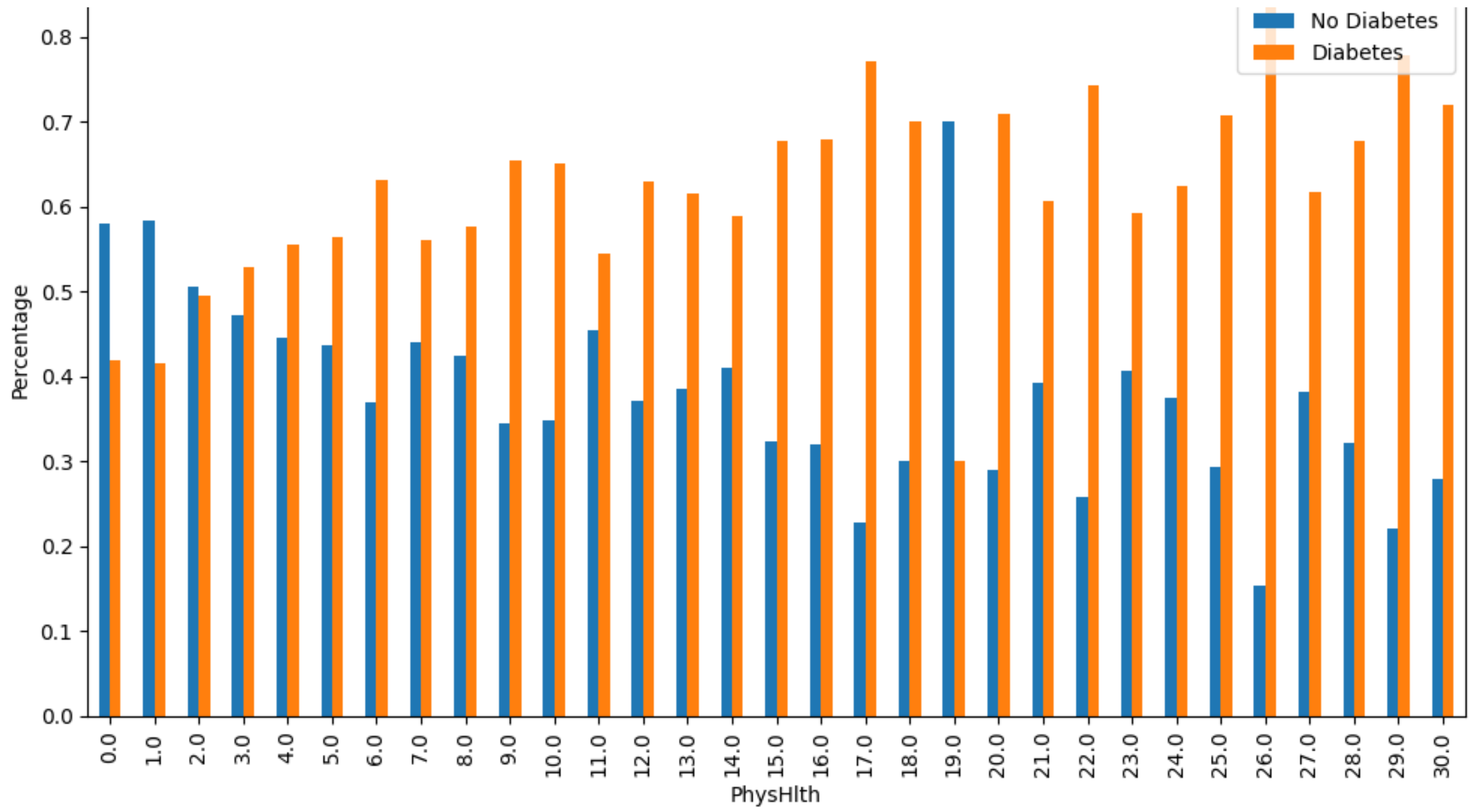


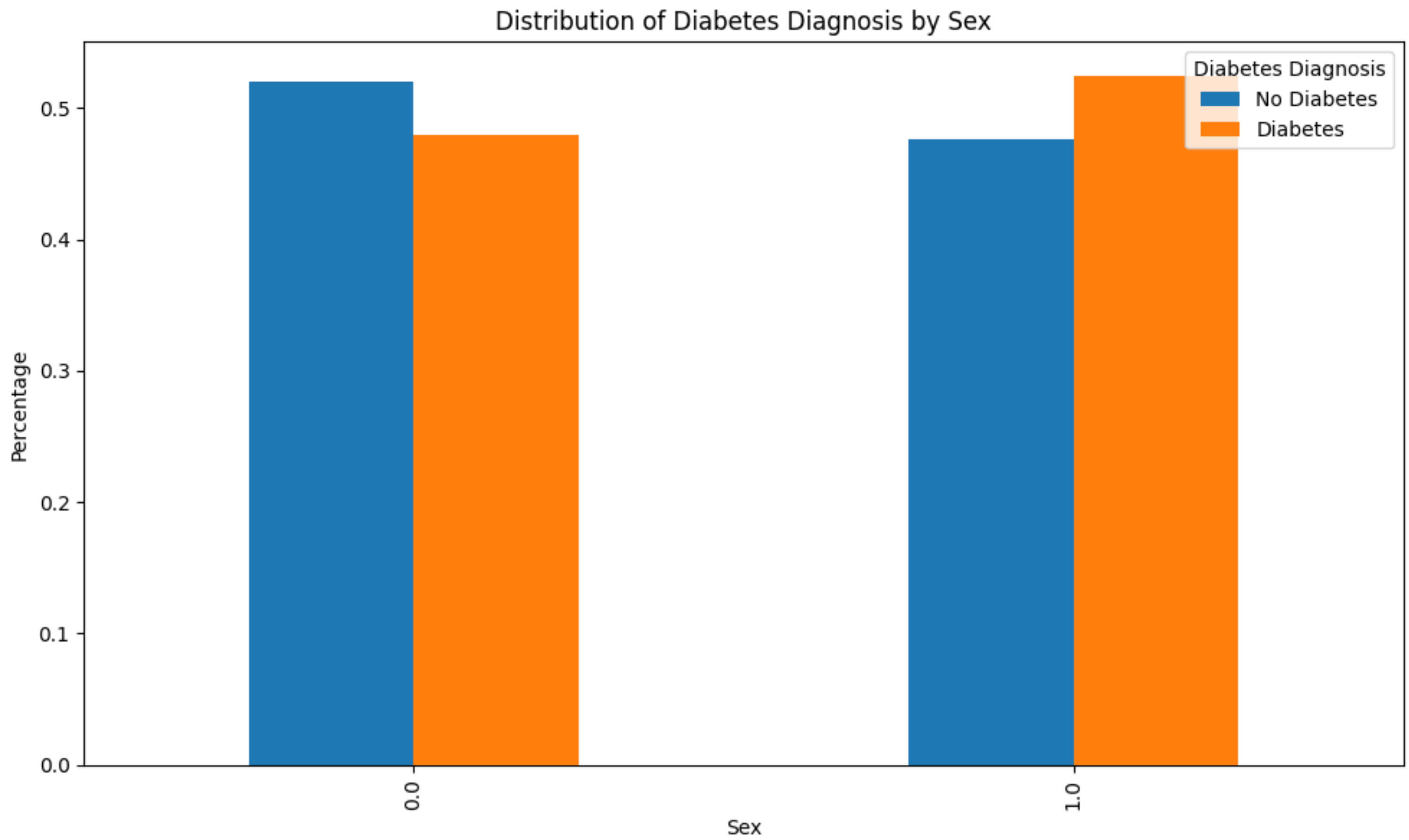
Distribution of Diabetes Diagnosis by PhysActivity

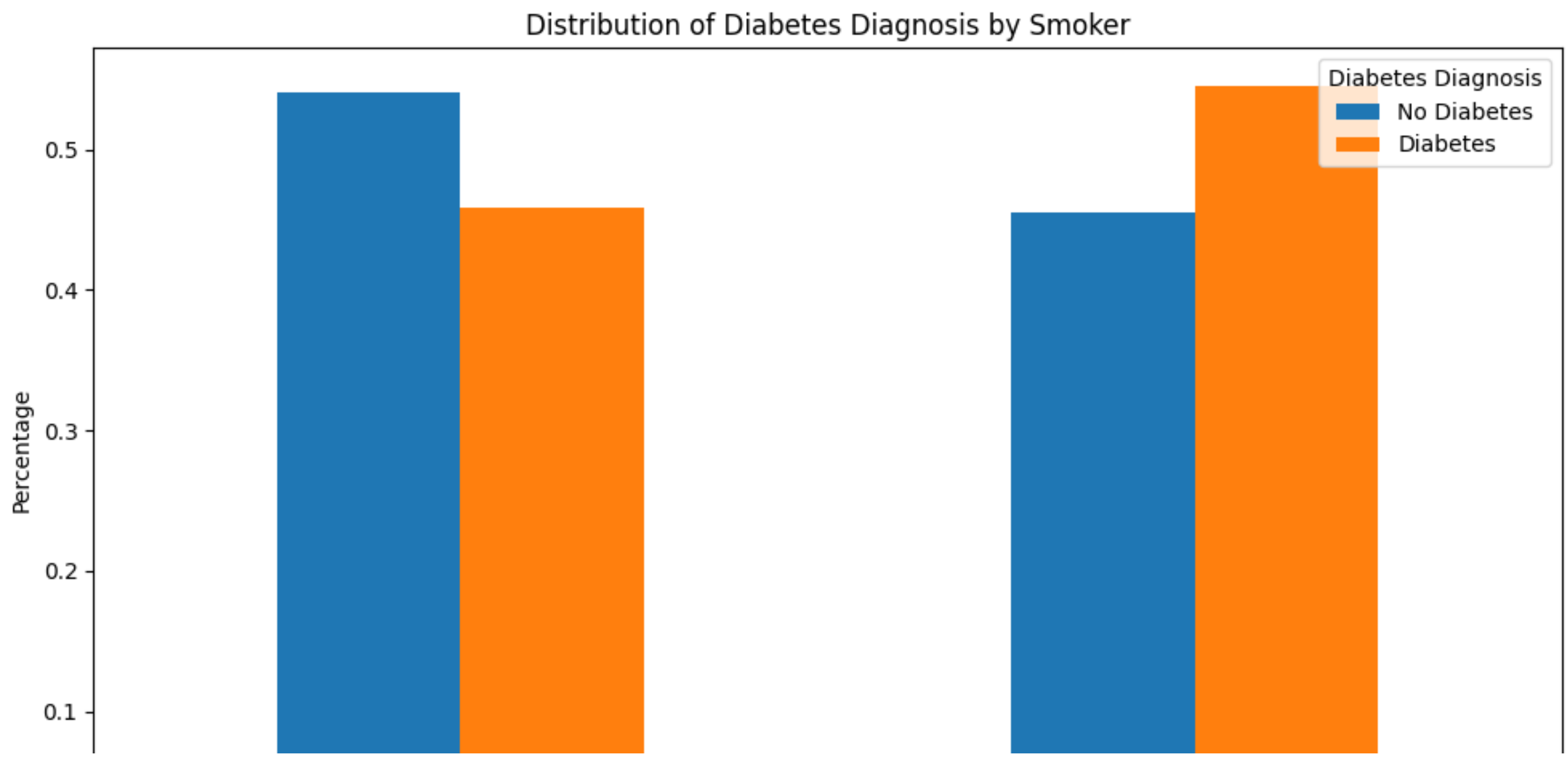


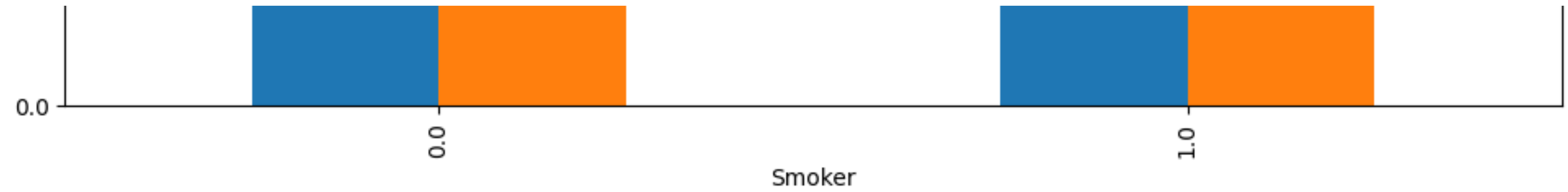
Distribution of Diabetes Diagnosis by PhysHlth

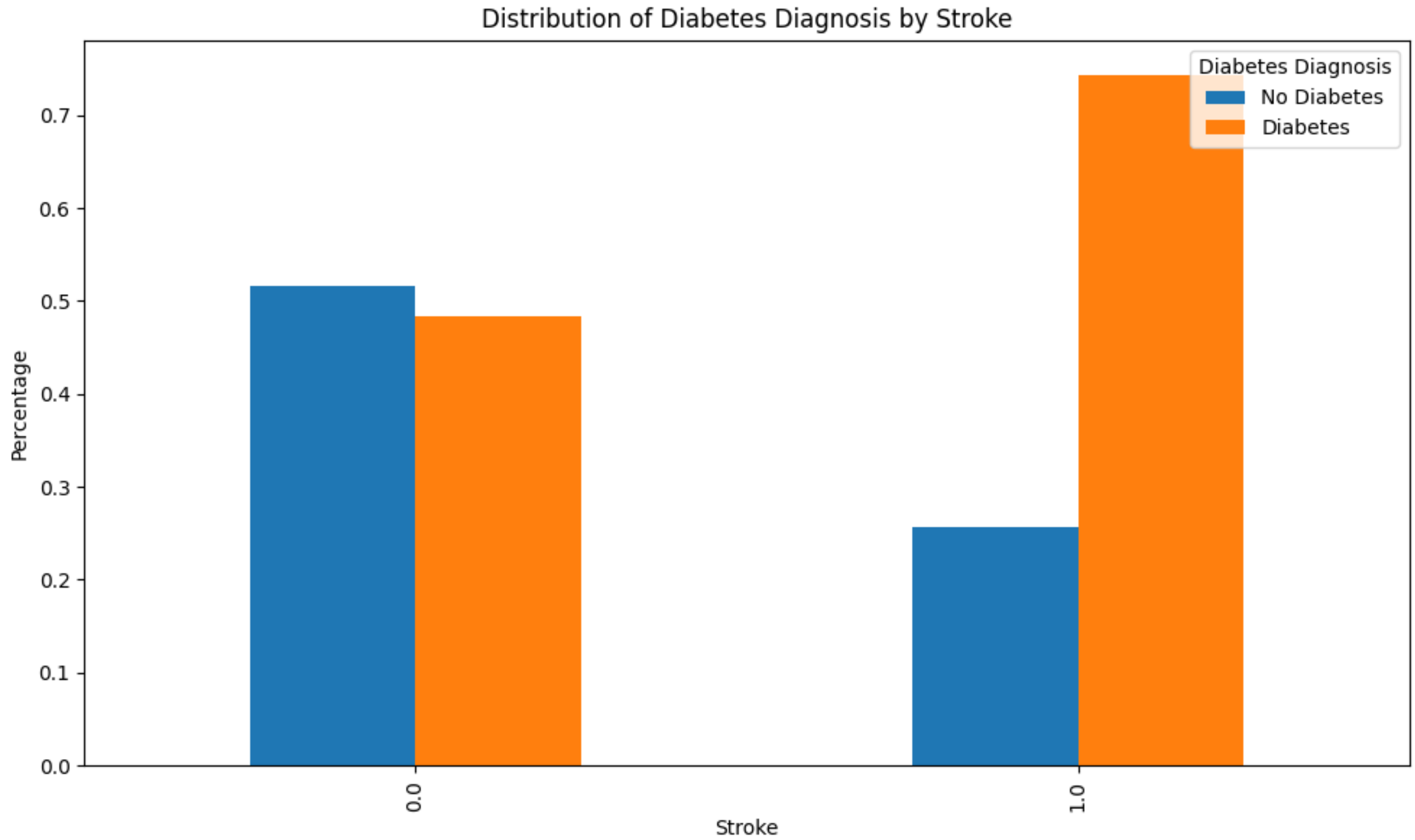


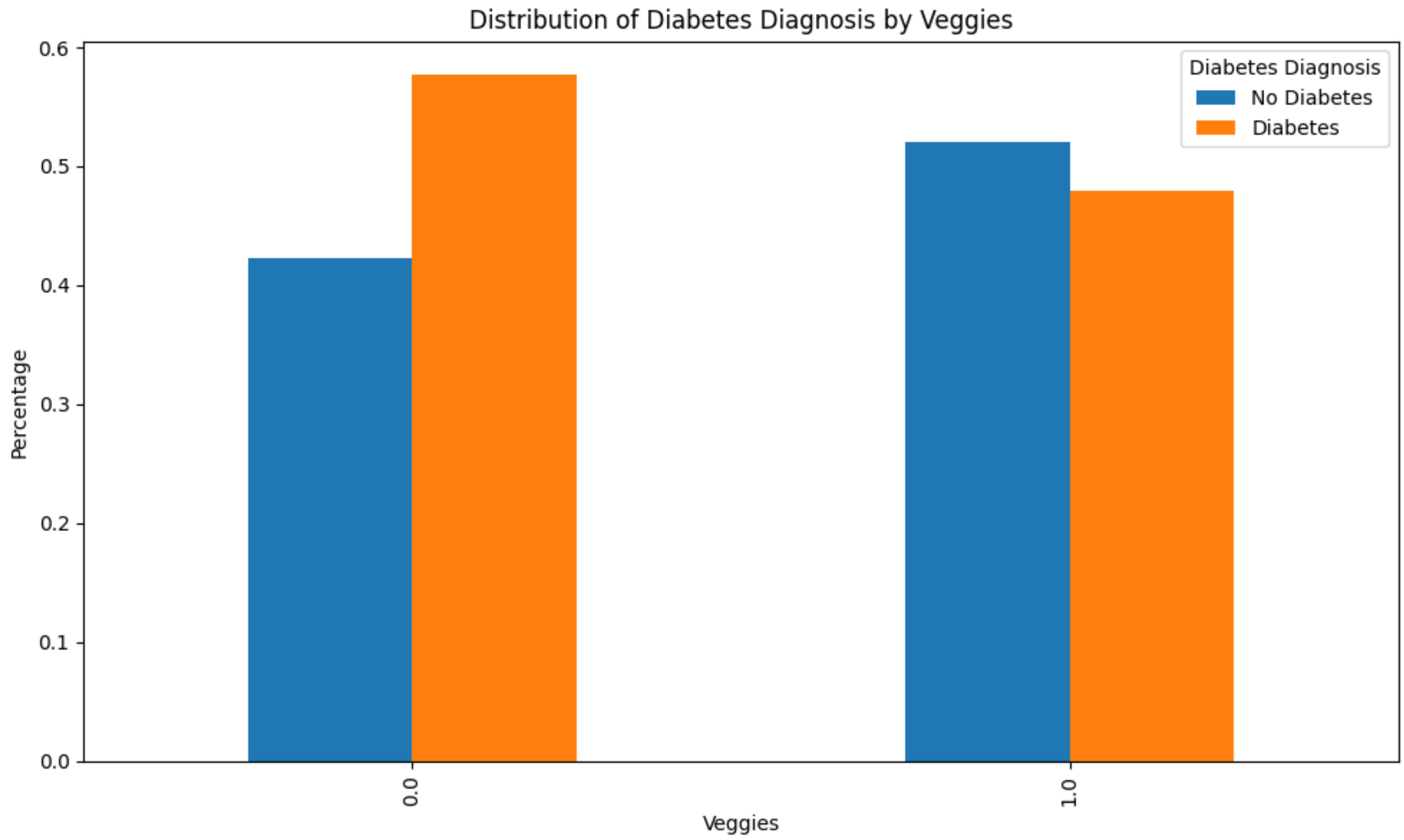

No Diabetes
Diabetes
Percentage
PhysHlth
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


Distribution of Diabetes Diagnosis by Sex
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
Sex
0.0
1.0


Distribution of Diabetes Diagnosis by Smoker
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage


0.0
0.0
1.0
Smoker


Distribution of Diabetes Diagnosis by Stroke
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.0
1.0
Stroke


Distribution of Diabetes Diagnosis by Veggies
Diabetes Diagnosis
No Diabetes
Diabetes
Percentage
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
1.0
Veggies