Final project:

Predictive model for Health Disease

Thuc Nhi Ly, Franck Vu

In [ ]:
```
#Import library
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
#Read the data to the csv file
df = pd.read_csv('CVD_cleaned.csv')
```

In [ ]:
```
#Display the first 5 row of data
df.head()
```

Out[ ]:

	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression
0	Poor	Within the past 2 years	No	No	No	No	No
1	Very Good	Within the past year	No	Yes	No	No	No
2	Very Good	Within the past year	Yes	No	No	No	No
3	Poor	Within the past year	Yes	Yes	No	No	No
4	Good	Within the past year	No	No	No	No	No

Data Exploration

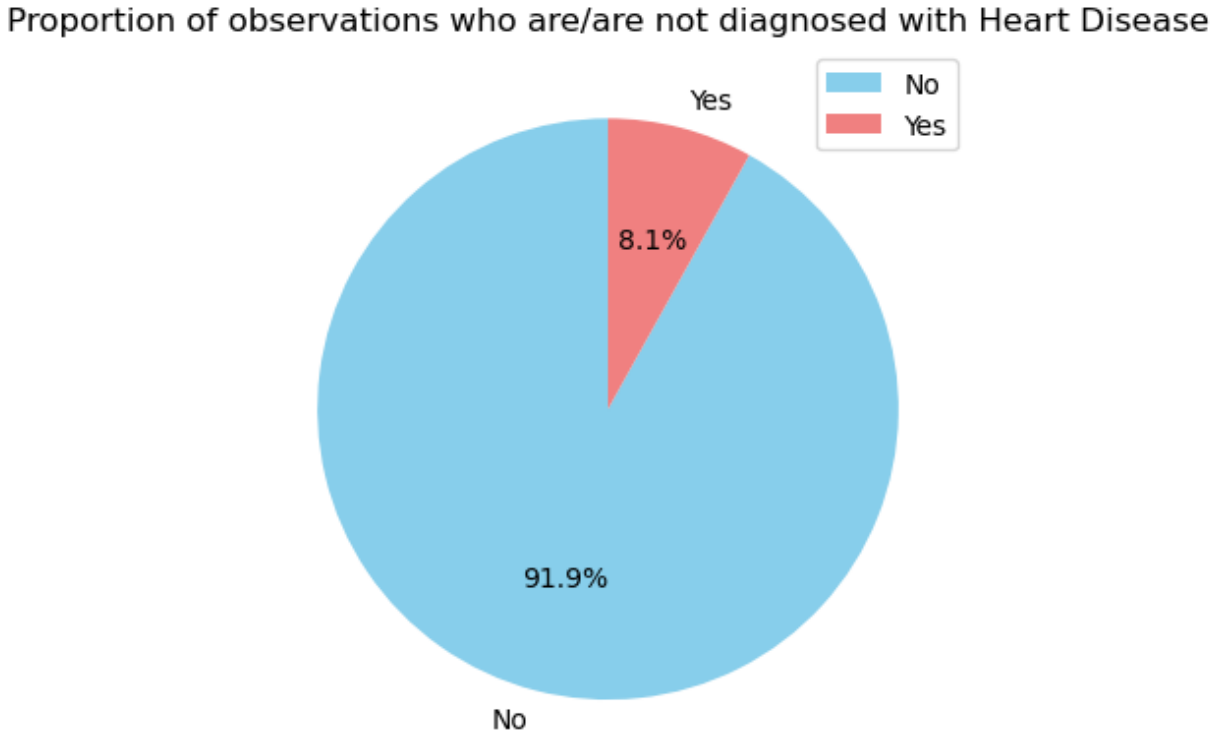
In [ ]:
```
#Count the distinct values of the target variable "Heart_Disease"
heartDisease_cnt = df['Heart_Disease'].value_counts().reset_index()
heartDisease_cnt
```

Out[ ]:

	Heart_Disease	count
0	No	283883
1	Yes	24971

In [ ]:
```
#Create a pie chart to display the portion of each distinct value in the targe
plt.pie(data = heartDisease_cnt, x = 'count',labels= heartDisease_cnt['Heart_D
```

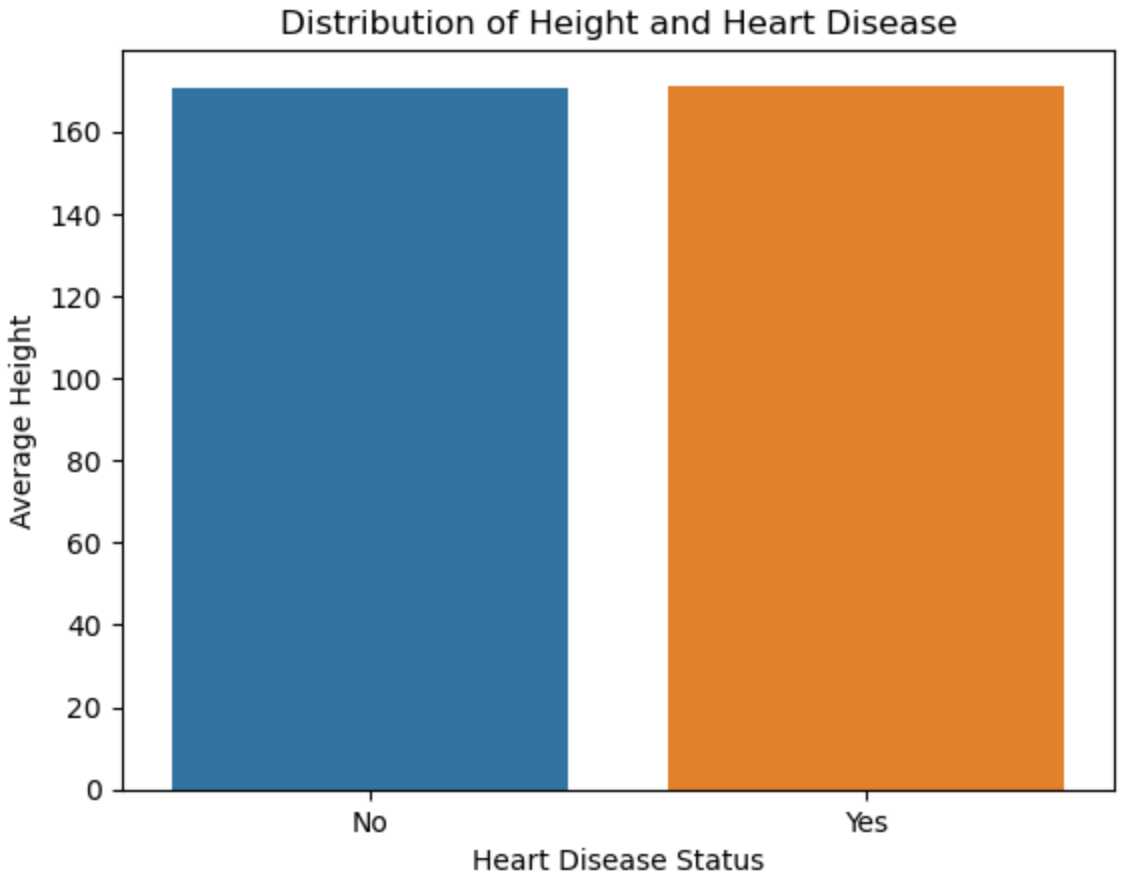
```
plt.title('Proportion of observations who are/are not diagnosed with Heart Dis
plt.legend()
plt.show()
```



Note:

As a result, 8.1% of the entire observations are diagnosed with heart disease, whereas 91.9% of the total observations are not diagnosed with heart disease, indicating that the data is highly unbalanced. Therefore, we should focus more on the portion diagnosed with heart disease while performing machine learning to avoid bias toward the majority of the observations.

In [ ]:
```
#Plot the distribution of Height and Heart Disease
heightDist = df.groupby('Heart_Disease')['Height_(cm)'].mean().reset_index()
sns.barplot(heightDist, x = 'Heart_Disease', y = 'Height_(cm)')
plt.title("Distribution of Height and Heart Disease")
plt.xlabel('Heart Disease Status')
plt.ylabel('Average Height')
plt.show()
```



In [ ]:
```
heightDist
```

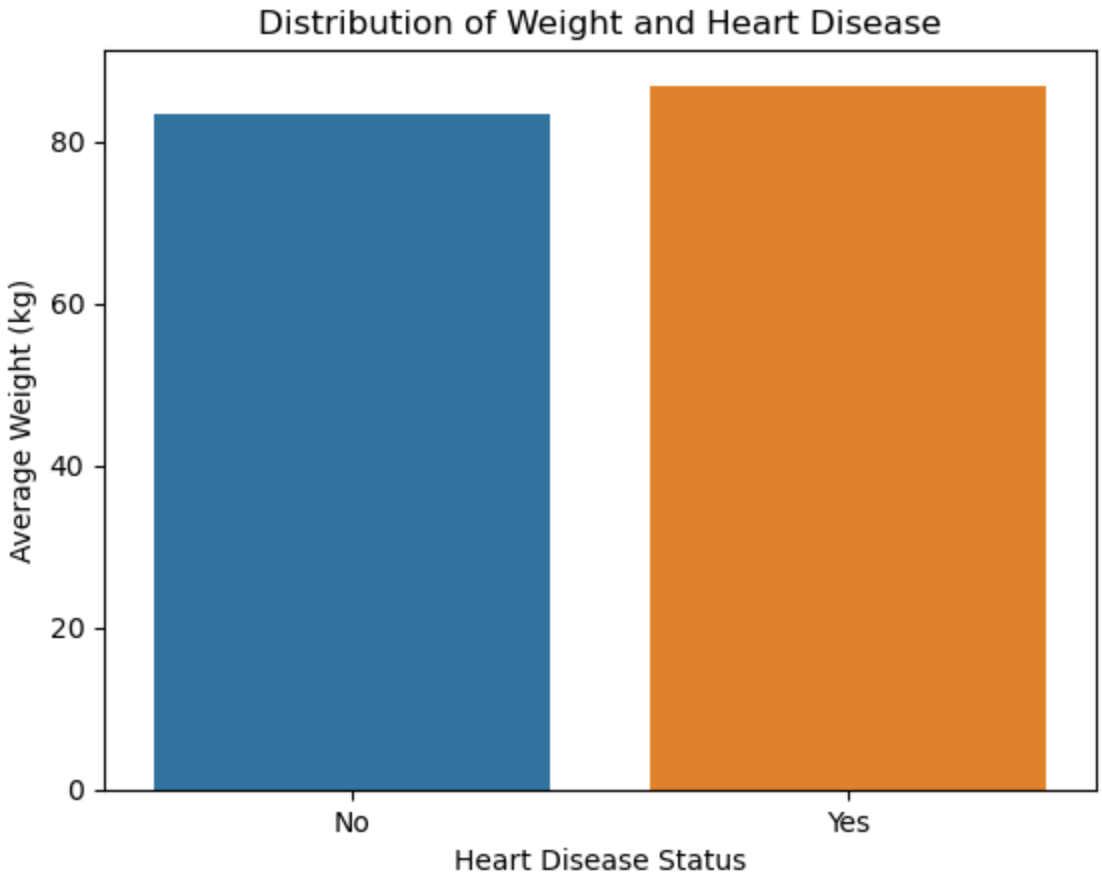
Out[ ]:

	Heart_Disease	Height_(cm)
0	No	170.565367
1	Yes	171.182332

Note:

There is a slightly different (~ 0.6 cm) in the average height between those who are diagnosed with heart disease and those who are not diagnosed with heart disease.

In [ ]:
```
#Plot the distribution of Height and Heart Disease
weightDist = df.groupby('Heart_Disease')['Weight_(kg)'].mean().reset_index()
sns.barplot(weightDist, x = 'Heart_Disease', y = 'Weight_(kg)')
plt.title("Distribution of Weight and Heart Disease")
plt.xlabel('Heart Disease Status')
plt.ylabel('Average Weight (kg)')
plt.show()
```



In [ ]:
```
weightDist
```

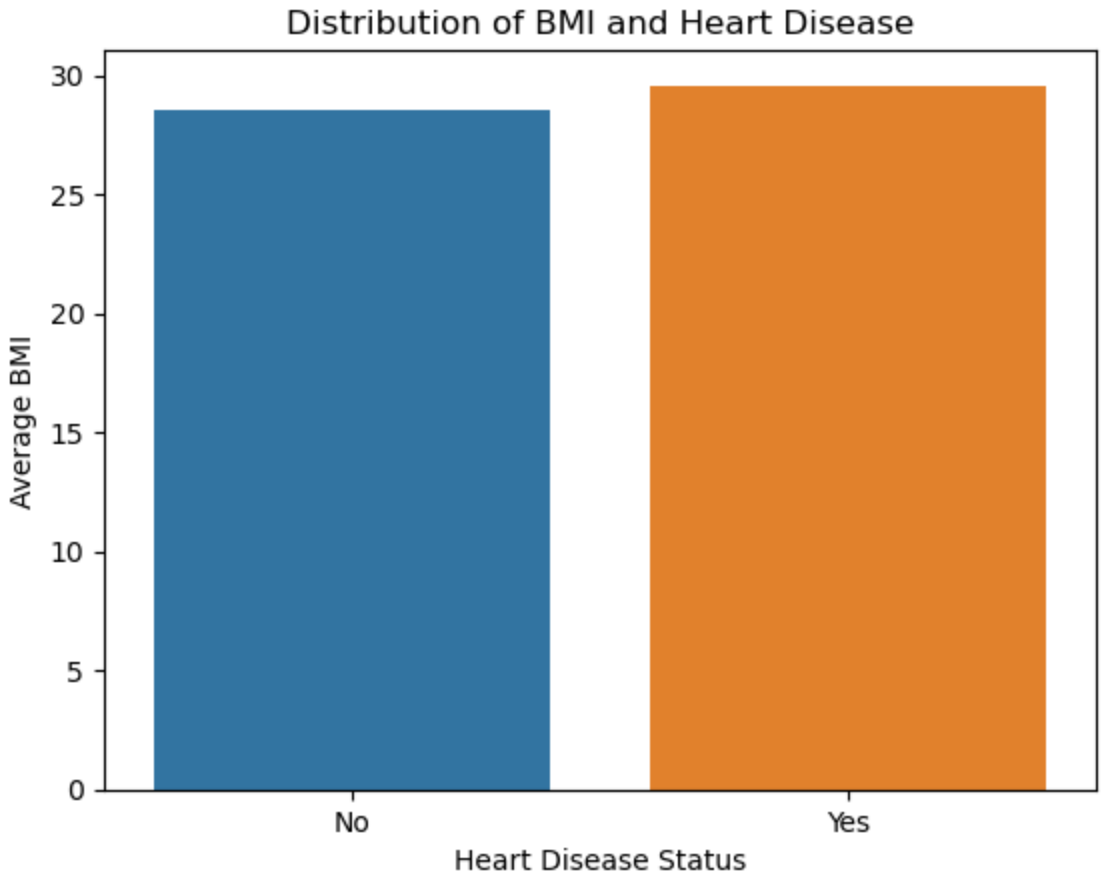
Out[ ]:

	Heart_Disease	Weight_(kg)
0	No	83.298262
1	Yes	86.889986

Note:

There is a slightly different (~ 3.5 kg) in the average weight between those who are diagnosed with heart disease and those who are not diagnosed with heart disease, but not as insignificant as Height. Therefore, Weight could be consider as an predictor variable for the predicted model.

In [ ]:
```
#Plot the distribution of Height and Heart Disease
BMIDist = df.groupby('Heart_Disease')['BMI'].mean().reset_index()
sns.barplot(BMIDist, x = 'Heart_Disease', y = 'BMI')
plt.title("Distribution of BMI and Heart Disease")
plt.xlabel('Heart Disease Status')
plt.ylabel('Average BMI')
plt.show()
```



In [ ]:
```
BMIDist
```

Out[ ]:

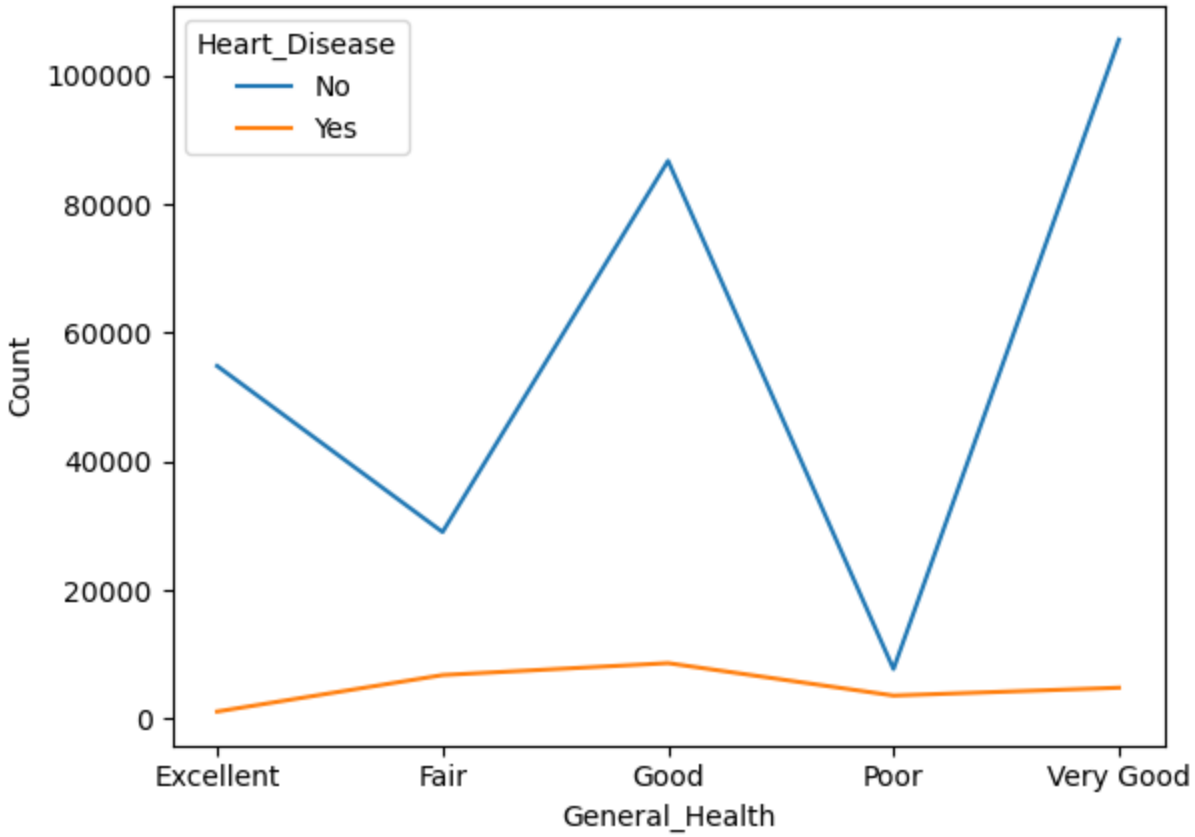
	Heart_Disease	BMI
0	No	28.543676
1	Yes	29.564505

Note:

There is a slightly different (~ 1) in the average BMI between those who are diagnosed with heart disease and those who are not diagnosed with heart disease.

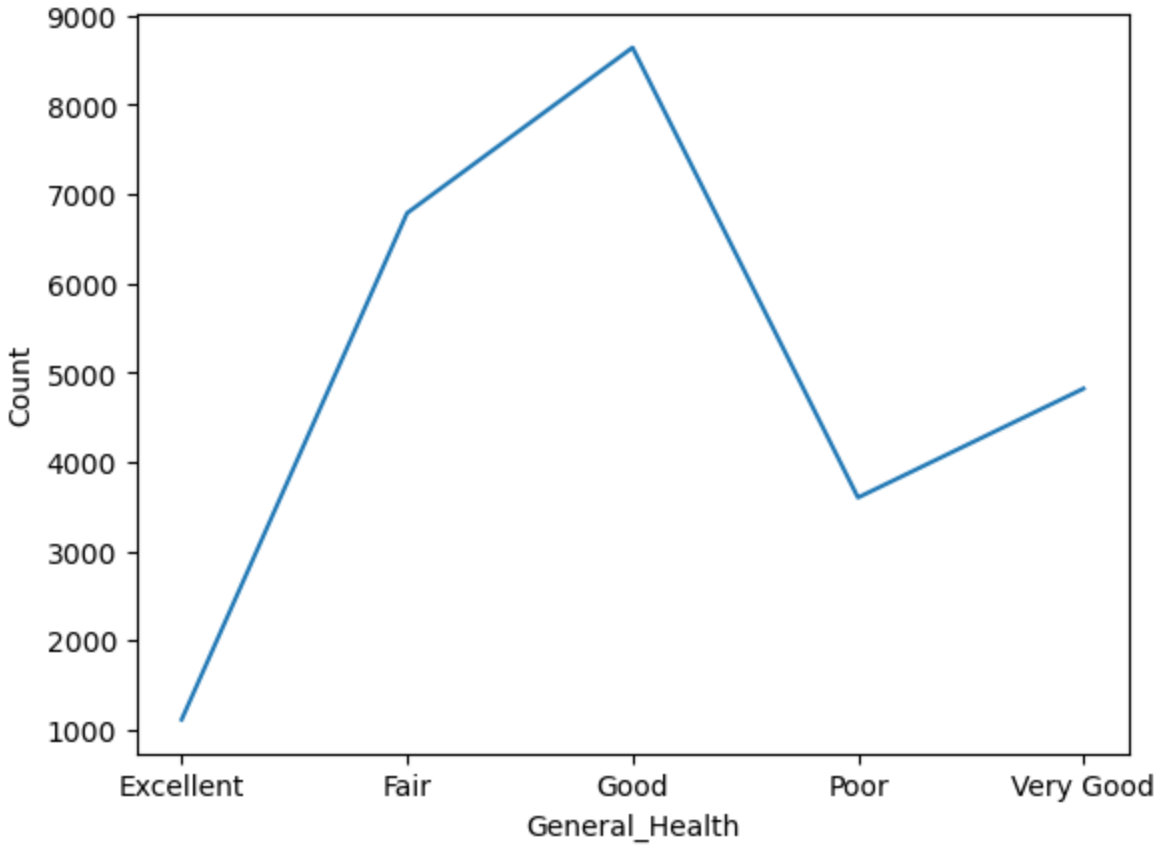
In [ ]:
```
#Plot the distribution between General Health and Heart Disease
generalHealth = df.groupby(['Heart_Disease', 'General_Health']).size().reset_i
sns.lineplot(generalHealth, x = 'General_Health', y = 'Count', hue = 'Heart_Di
```

Out[ ]:
```
<Axes: xlabel='General_Health', ylabel='Count'>
```



In [ ]:
```
#Plot the distribution between General Health and those who are diagnosed with
sns.lineplot(generalHealth[generalHealth['Heart_Disease']== 'Yes'], x= 'Genera
```

Out[ ]:
```
<Axes: xlabel='General_Health', ylabel='Count'>
```



Note:

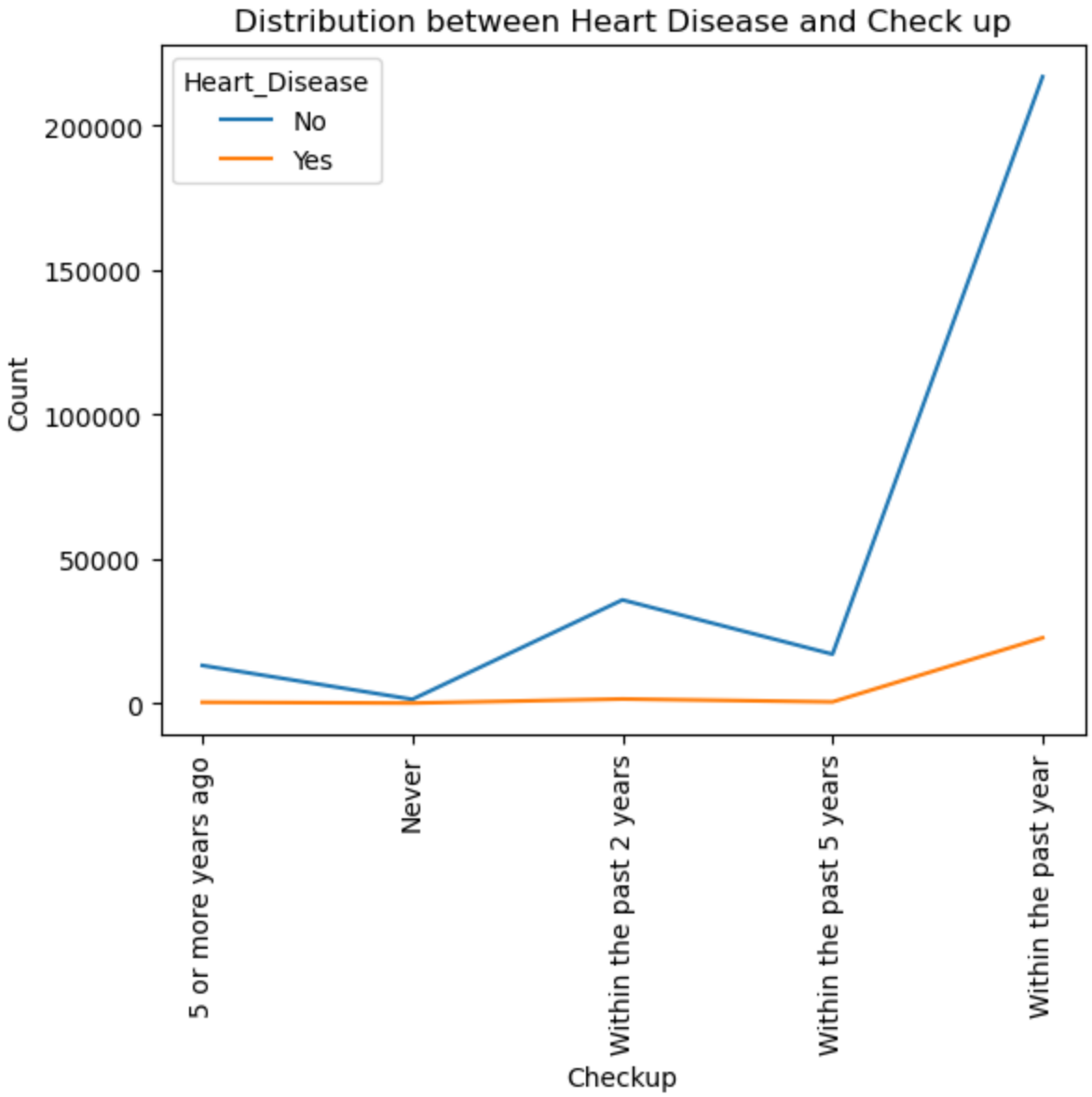
For this dataset, among those who are diagnosed with heart disease, most of them have good and fair general health.

In [ ]:
```
#Group Heart_Disease and Checkup by counting the number of observations
checkUp = df.groupby(['Heart_Disease', 'Checkup']).size().reset_index(name = '
checkUp
```

Out[ ]:

	Heart_Disease	Checkup	Count
0	No	5 or more years ago	13079
1	No	Never	1349
2	No	Within the past 2 years	35748
3	No	Within the past 5 years	16971
4	No	Within the past year	216736
5	Yes	5 or more years ago	342
6	Yes	Never	58
7	Yes	Within the past 2 years	1465
8	Yes	Within the past 5 years	471
9	Yes	Within the past year	22635

In [ ]:
```
#Plot the distribution between Heart_Disease and Checkup
sns.lineplot(checkUp, x = 'Checkup', y = 'Count', hue = 'Heart_Disease')
plt.xticks(rotation = 90)
plt.title('Distribution between Heart Disease and Check up')
plt.show()
```

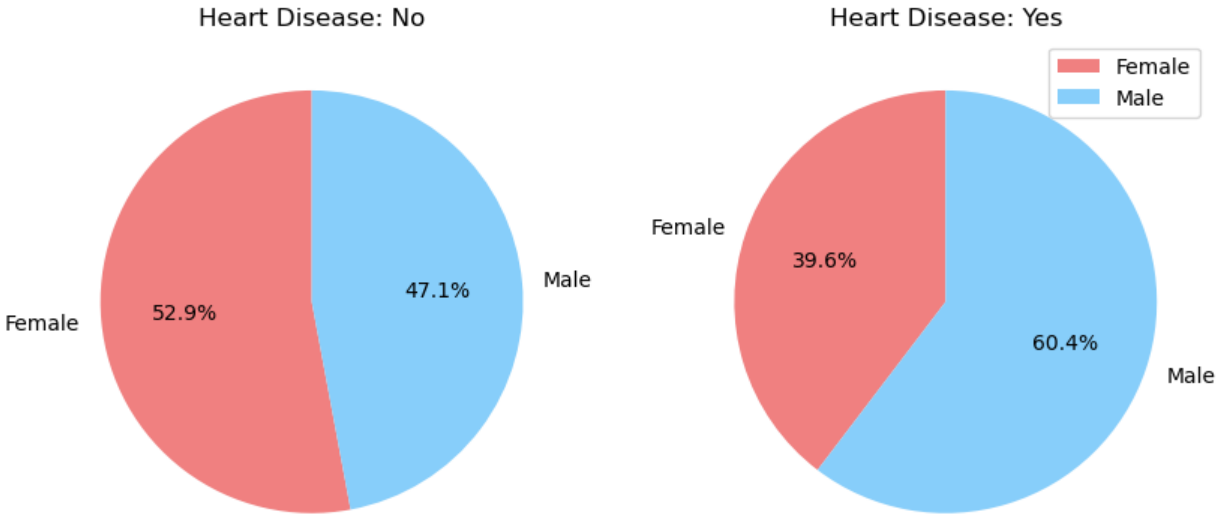


Note:

The amount of those who did check up within the past year and had no heart disease is exclusively high. This Checkup - Within the past year could be a significant variable for the predicted model.

In [ ]:
```
#Group the data between Heart Disease and Sex by counting the number of observa
sex = df.groupby(['Heart_Disease', 'Sex']).size().reset_index(name = 'Count')
# Separate data for 'No' and 'Yes' categories
no_data = sex[sex['Heart_Disease'] == 'No']
yes_data = sex[sex['Heart_Disease'] == 'Yes']

# Plotting pie charts
fig, axes = plt.subplots(1, 2, figsize=(10, 5))


# Pie chart for 'No' category
axes[0].pie(no_data['Count'], labels=no_data['Sex'], autopct='%1.1f%%', startar
axes[0].set_title('Heart Disease: No')
```

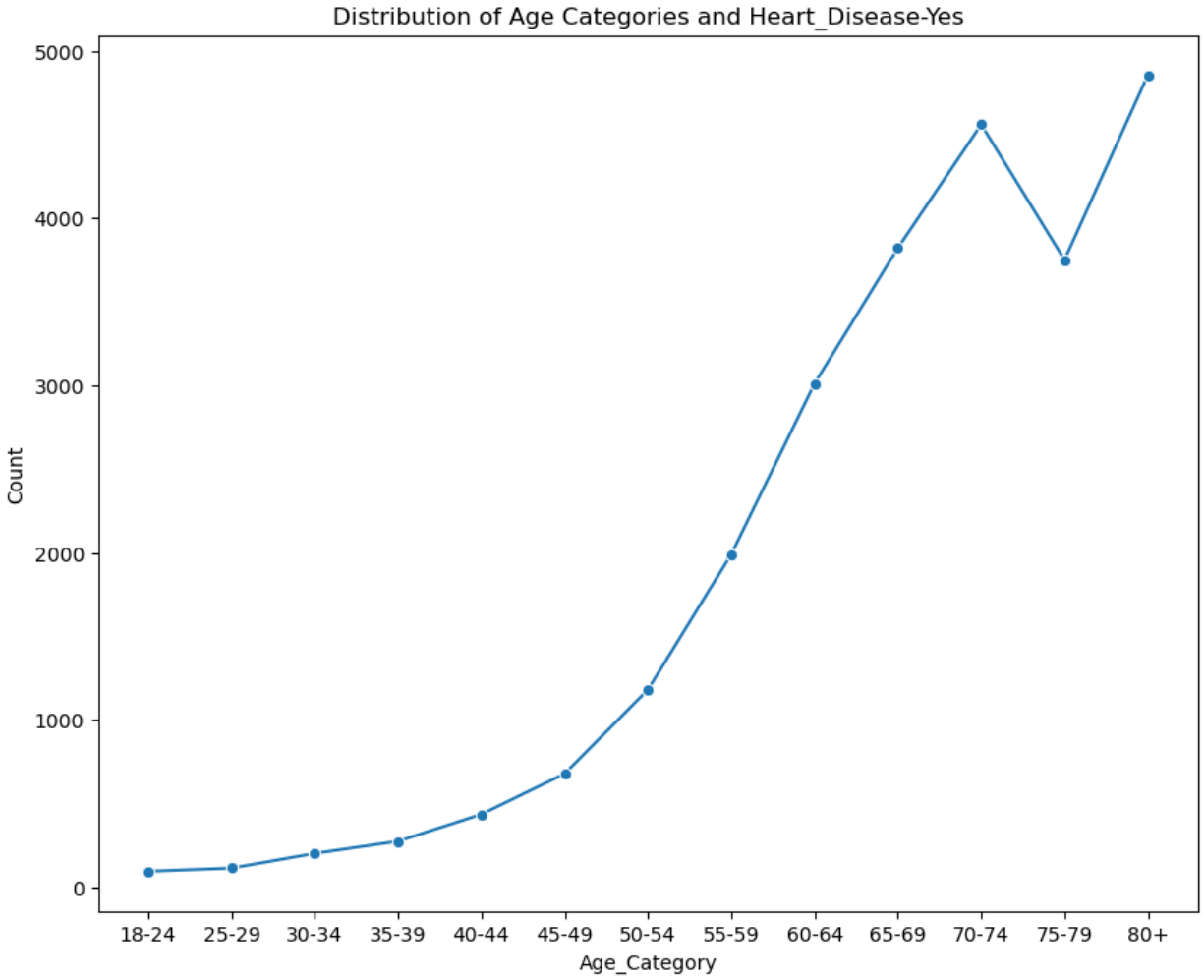
```
# Pie chart for 'Yes' category
axes[1].pie(yes_data['Count'], labels=yes_data['Sex'], autopct='%1.1f%%', star
axes[1].set_title('Heart Disease: Yes')

plt.legend()
plt.show()
```



Note:

Among those who are diagnosed with heart disease, 60.4% of them are males, whereas 39.6% of them are female. Since the difference is quite significant, this variable is useful to predict the target variable.

In [ ]:
```
#Group the data between Heart Disease and Age Category by counting the number
ageDist = df.groupby(['Heart_Disease', 'Age_Category']).size().reset_index(name
plt.figure(figsize=(10,8))
sns.lineplot(ageDist[ageDist['Heart_Disease']=='Yes'], x = 'Age_Category', y =
plt.title('Distribution of Age Categories and Heart_Disease-Yes')
plt.show()
```

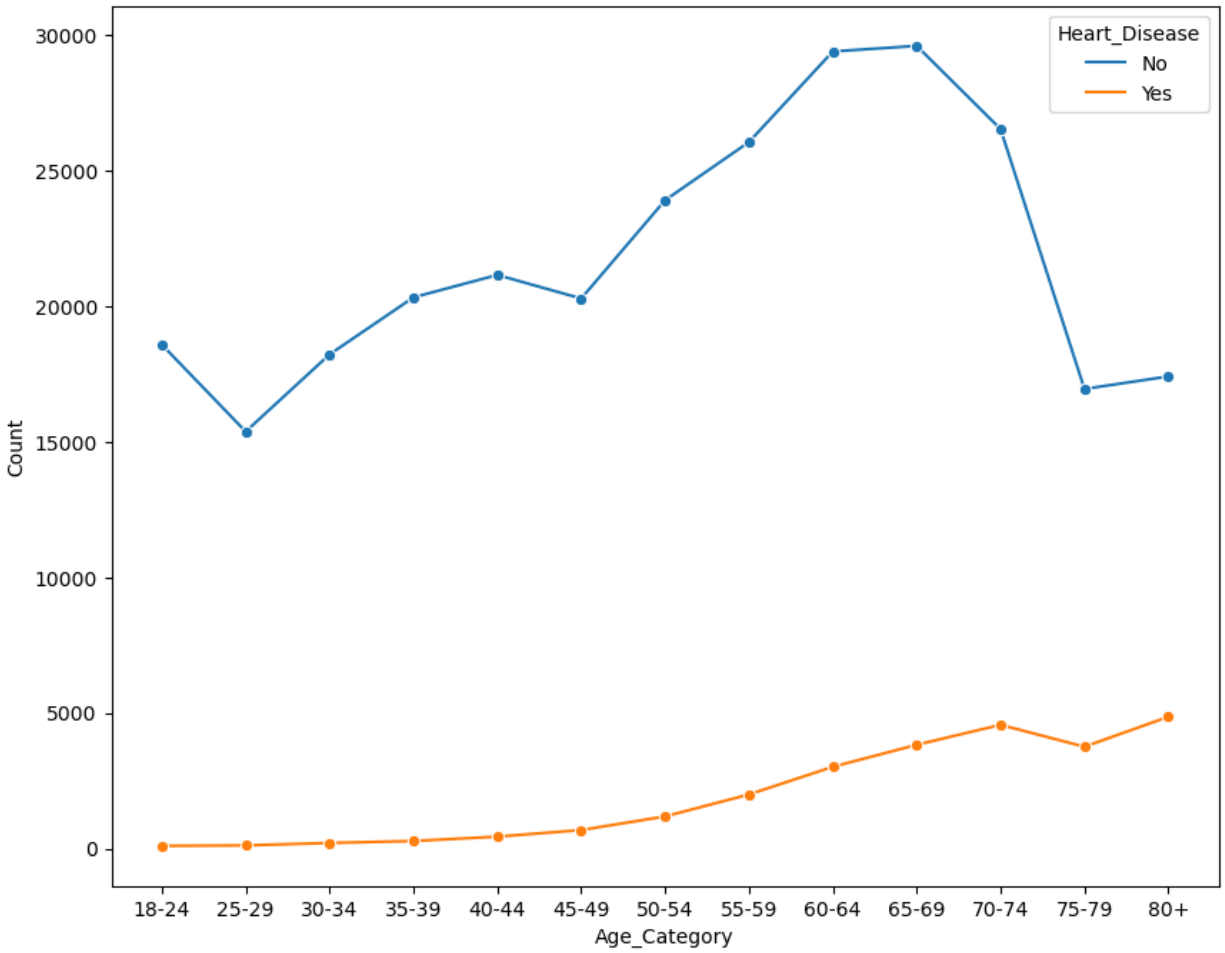


Note:

Age_Category: Based on the trend line between Age_Category and the count number of Heart_Disease, there is a positive linear correlation between the two variables. As the Age increases, the number of people who are diagnosed with heart disease would also increase.

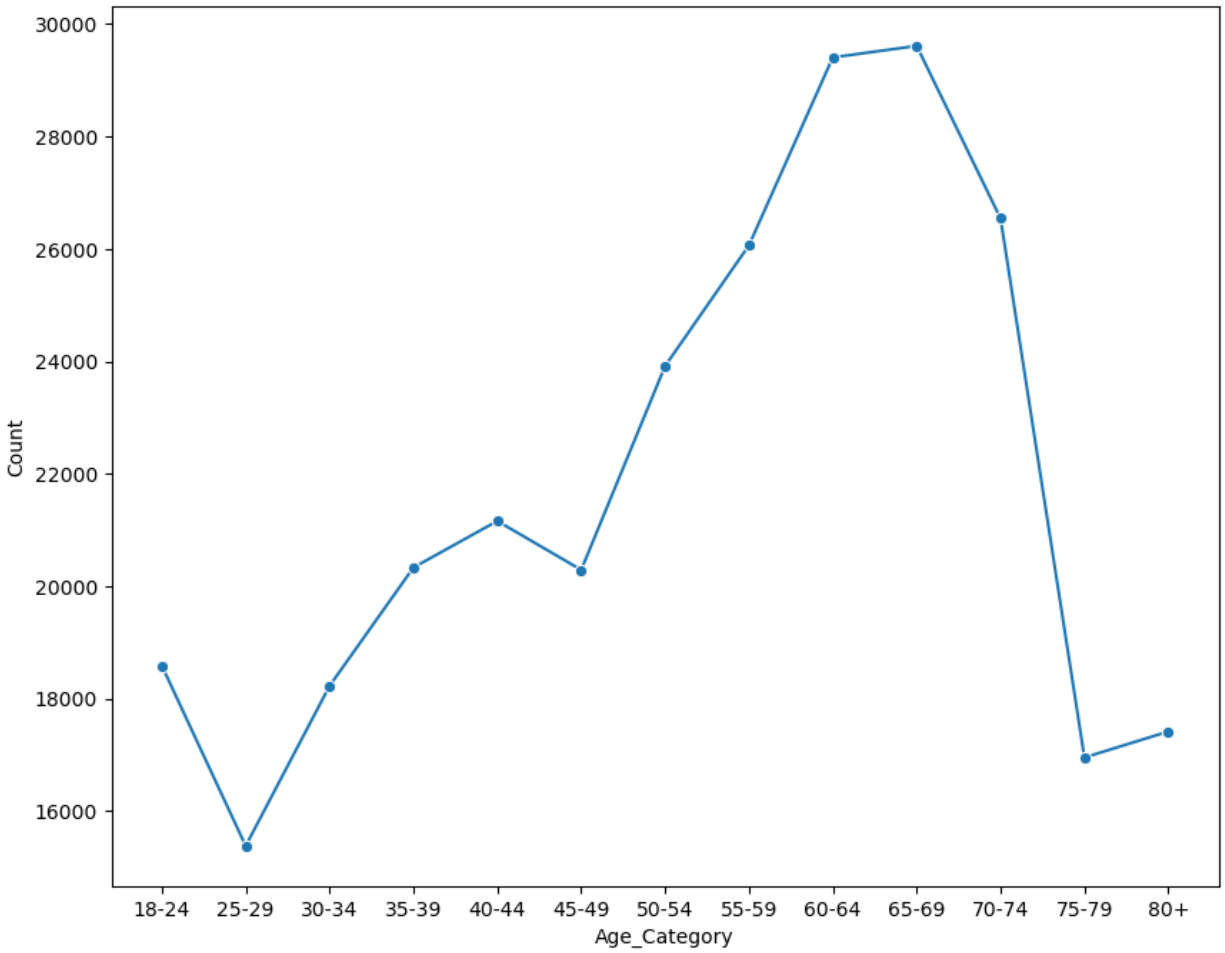
In [ ]:
```
#Plot the line chart for both group of Heart Disease
plt.figure(figsize=(10,8))
sns.lineplot(ageDist, x = 'Age_Category', y = 'Count', hue = 'Heart_Disease', r
```

Out[ ]:
```
<Axes: xlabel='Age_Category', ylabel='Count'>
```

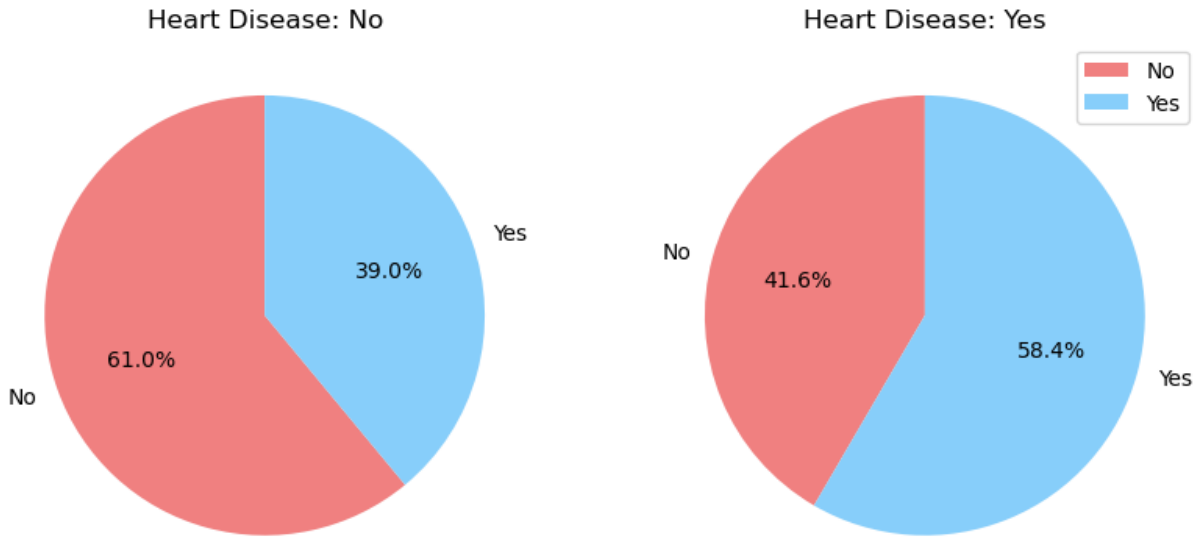


In [ ]:
```
#Plot the line chart for group of Heart Disease_Yes
plt.figure(figsize=(10,8))
sns.lineplot(ageDist[ageDist['Heart_Disease']=='No'], x = 'Age_Category', y =
```

Out[ ]:
```
<Axes: xlabel='Age_Category', ylabel='Count'>
```



In [ ]:
```
#Group the data between Heart Disease and Smoking History by counting the numbe
smokingHistory = df.groupby(['Heart_Disease', 'Smoking_History']).size().reset_
# Separate data for 'No' and 'Yes' categories
no_data = smokingHistory[smokingHistory['Heart_Disease'] == 'No']
yes_data = smokingHistory[smokingHistory['Heart_Disease'] == 'Yes']

# Plotting pie charts
fig, axes = plt.subplots(1, 2, figsize=(10, 5))


# Pie chart for 'No' category
axes[0].pie(no_data['Count'], labels=no_data['Smoking_History'], autopct='%1.1
axes[0].set_title('Heart Disease: No')

# Pie chart for 'Yes' category
axes[1].pie(yes_data['Count'], labels=yes_data['Smoking_History'], autopct='%1
axes[1].set_title('Heart Disease: Yes')

plt.legend()
plt.show()
```



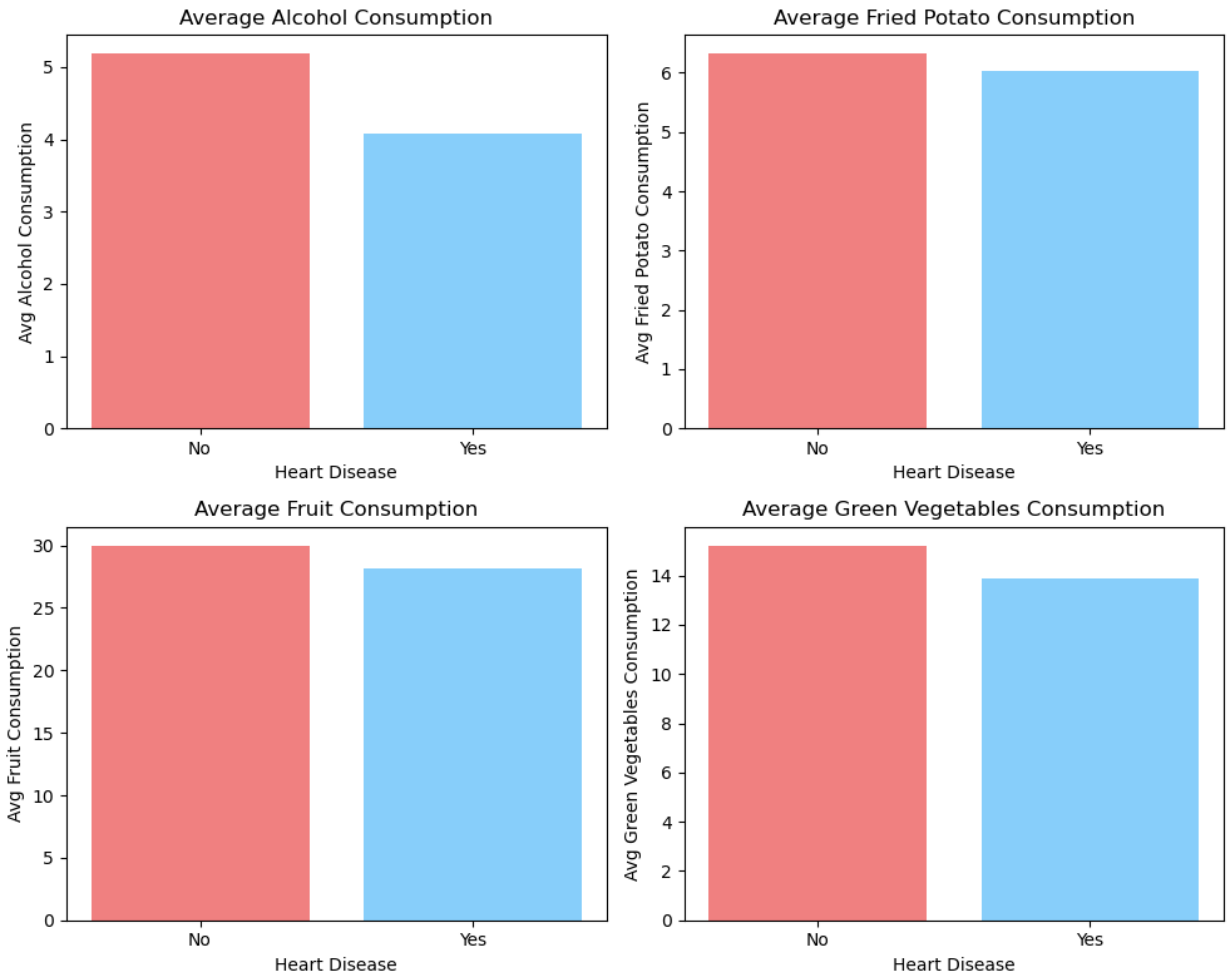
Note:

As the result, 58.4 % of those who are diagnosed with heart disease have smoking history, whereas 39% of those who are not diagnosed with heart disease have smoking history. Overall, those who have smoking history would tend to have risk of being diagnosed with heart disease.

In [ ]:
```
#Group the data between food & drink consumption and Heart Disease by counting
alcoholConsumption = df.groupby(['Heart_Disease'])['Alcohol_Consumption'].mean
friedPotatoConsumption = df.groupby('Heart_Disease')['FriedPotato_Consumption']
fruitConsumption = df.groupby('Heart_Disease')['Fruit_Consumption'].mean().res
greenVegetablesConsumption = df.groupby('Heart_Disease')['Green_Vegetables_Cons
```

In [ ]:
```
#Create a 2x2 subplot grid
fig, axes = plt.subplots(2,2, figsize = (10,8))


#Bar chart for Alcohol Consumption
bar1 = axes[0,0].bar(alcoholConsumption['Heart_Disease'], alcoholConsumption['A
axes[0,0].set_title('Average Alcohol Consumption')
axes[0,0].set_xlabel('Heart Disease')
axes[0,0].set_ylabel('Avg Alcohol Consumption')

#Bar chart for Fried Potatio Consumption
bar2 = axes[0,1].bar(friedPotatoConsumption['Heart_Disease'], friedPotatoConsum
axes[0,1].set_title('Average Fried Potato Consumption')
axes[0,1].set_xlabel('Heart Disease')
axes[0,1].set_ylabel('Avg Fried Potato Consumption')

#Bar chart for Fruit Consumption
bar3 = axes[1,0].bar(fruitConsumption['Heart_Disease'], fruitConsumption['Avg_C
axes[1,0].set_title('Average Fruit Consumption')
axes[1,0].set_xlabel('Heart Disease')
axes[1,0].set_ylabel('Avg Fruit Consumption')

#Bar chart for Green Vegetables Consumption
bar4 = axes[1,1].bar(greenVegetablesConsumption['Heart_Disease'], greenVegetab
```

```
axes[1,1].set_title('Average Green Vegetables Consumption')
axes[1,1].set_xlabel('Heart Disease')
axes[1,1].set_ylabel('Avg Green Vegetables Consumption')

plt.tight_layout()
plt.show()
```

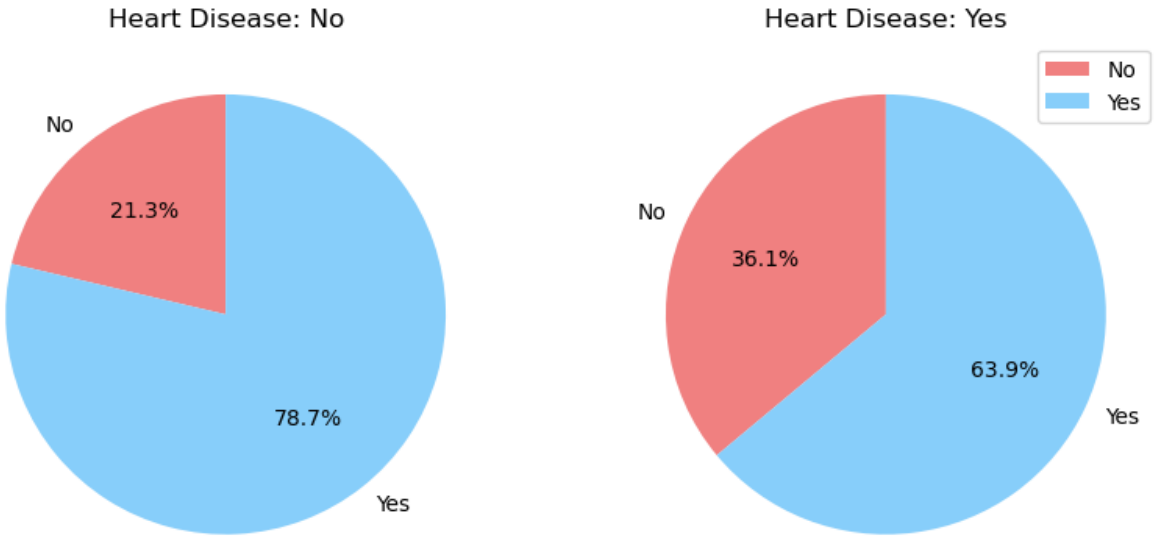


Note:

There is no significant difference between the Average Consumption and each group of Heart Disease (Yes/No). Based on the data distribution, these variables do not identify the causation of Heart Disease.

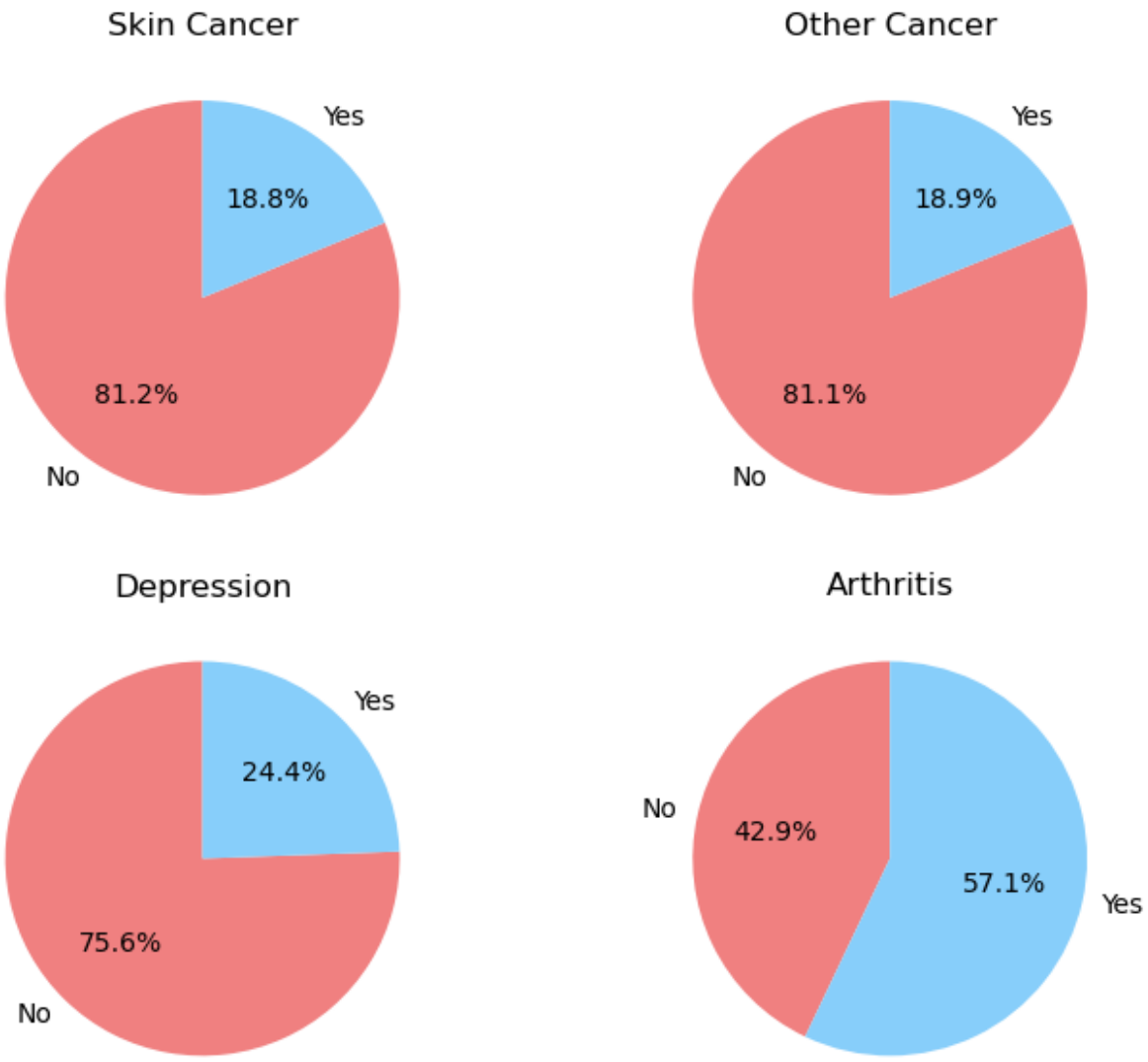
In [ ]:
```
exerciseDist = df.groupby(['Heart_Disease', 'Exercise']).size().reset_index(nar
# Separate data for 'No' and 'Yes' categories
no_data = exerciseDist[exerciseDist['Heart_Disease'] == 'No']
yes_data = exerciseDist[exerciseDist['Heart_Disease'] == 'Yes']

# Plotting pie charts
fig, axes = plt.subplots(1, 2, figsize=(10, 5))


# Pie chart for 'No' category
axes[0].pie(no_data['Count'], labels=no_data['Exercise'], autopct='%1.1f%%', s
axes[0].set_title('Heart Disease: No')
```

```
# Pie chart for 'Yes' category
axes[1].pie(yes_data['Count'], labels=yes_data['Exercise'], autopct='%1.1f%%',
axes[1].set_title('Heart Disease: Yes')

plt.legend()
plt.show()
```



In [ ]:
```
#Group the data between all of other diseases and Heart Disease by counting the
skinCancer = df.groupby(['Heart_Disease', 'Skin_Cancer']).size().reset_index(na
otherCancer = df.groupby(['Heart_Disease', 'Other_Cancer']).size().reset_index
depression = df.groupby(['Heart_Disease', 'Depression']).size().reset_index(nar
arthritis = df.groupby(['Heart_Disease', 'Arthritis']).size().reset_index(name
```

In [ ]:
```
#Create a 2x2 subplot grid
fig, axes = plt.subplots(2,2, figsize = (8,6))
skinCancer_y = skinCancer[skinCancer['Heart_Disease']=='Yes']
otherCancer_y = otherCancer[otherCancer['Heart_Disease']=='Yes']
depression_y = depression[depression['Heart_Disease']=='Yes']
arthritis_y = arthritis[arthritis['Heart_Disease']=='Yes']

#Pie chart for skin cancer
pie1 = axes[0,0].pie(skinCancer_y['Count'], labels=skinCancer_y['Skin_Cancer']
axes[0,0].set_title('Skin Cancer')

#Pie chart for other cancer
pie2 = axes[0,1].pie(otherCancer_y['Count'], labels=otherCancer_y['Other_Cance
axes[0,1].set_title('Other Cancer')

#Pie chart for depression
pie3 = axes[1,0].pie(depression_y['Count'], labels=depression_y['Depression'],
axes[1,0].set_title('Depression')

#Pie chart for arthritis
pie4 = axes[1,1].pie(arthritis_y['Count'], labels=arthritis_y['Arthritis'], au
axes[1,1].set_title('Arthritis')

plt.tight_layout()
plt.show()
```

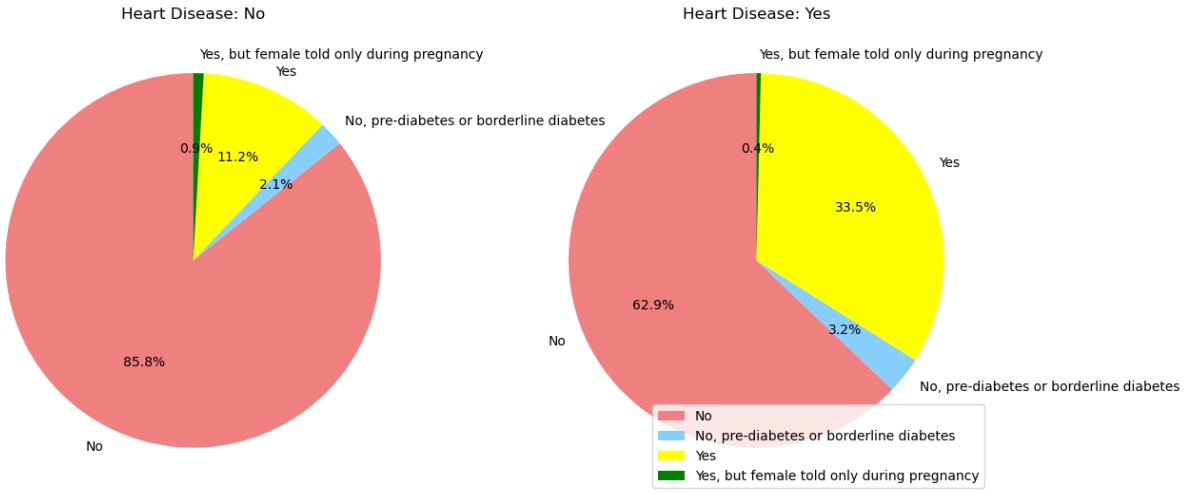


Note:

For those who are diagnosed with Heart Disease, 57.1 % of them are also diagnosed with Arthritis.

In [ ]:
```
diabetes = df.groupby(['Heart_Disease', 'Diabetes']).size().reset_index(name =
# Separate data for 'No' and 'Yes' categories
no_data = diabetes[diabetes['Heart_Disease'] == 'No']
yes_data = diabetes[diabetes['Heart_Disease'] == 'Yes']

# Plotting pie charts
fig, axes = plt.subplots(1, 2, figsize=(14,10))


# Pie chart for 'No' category
axes[0].pie(no_data['Count'], labels=no_data['Diabetes'], autopct='%1.1f%%', s
axes[0].set_title('Heart Disease: No')

# Pie chart for 'Yes' category
axes[1].pie(yes_data['Count'], labels=yes_data['Diabetes'], autopct='%1.1f%%',
axes[1].set_title('Heart Disease: Yes')
```

```
plt.legend(loc = 'lower right')
plt.show()
```



Model Selection

In [ ]:
```
#Import libraries
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.utils.class_weight import compute_sample_weight
from sklearn.metrics import accuracy_score, roc_auc_score ,confusion_matrix, c
from statsmodels.discrete.discrete_model import Logit
from statsmodels.tools import add_constant as add_constant
```

In [ ]:
```
#Get the dummies for categorical variables

df = pd.get_dummies(df, columns=['General_Health', 'Checkup', 'Sex','Diabetes'
pd.set_option('display.max_columns', None)
df.head()
```

Out[ ]:

	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Arthritis	Height_(cm)	We
0	No	No	No	No	No	Yes	150.0	
1	No	Yes	No	No	No	No	165.0	
2	Yes	No	No	No	No	No	163.0	
3	Yes	Yes	No	No	No	No	180.0	
4	No	No	No	No	No	No	191.0	

In [ ]:
```
#Replace the value Yes/No with 1/0
df.replace({'Yes': 1, 'No': 0}, inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 308854 entries, 0 to 308853
Data columns (total 43 columns):
 #   Column                                            Non-Null Count   Dty
pe
---  ------                                            --------------   ---
--
 0   Exercise                                          308854 non-null  int
64
 1   Heart_Disease                                     308854 non-null  int
64
 2   Skin_Cancer                                       308854 non-null  int
64
 3   Other_Cancer                                      308854 non-null  int
64
 4   Depression                                        308854 non-null  int
64
 5   Arthritis                                         308854 non-null  int
64
 6   Height_(cm)                                       308854 non-null  flo
at64
 7   Weight_(kg)                                       308854 non-null  flo
at64
 8   BMI                                               308854 non-null  flo
at64
 9   Smoking_History                                   308854 non-null  int
64
 10  Alcohol_Consumption                               308854 non-null  flo
at64
 11  Fruit_Consumption                                 308854 non-null  flo
at64
 12  Green_Vegetables_Consumption                      308854 non-null  flo
at64
 13  FriedPotato_Consumption                           308854 non-null  flo
at64
 14  General_Health_Excellent                          308854 non-null  boo
l
 15  General_Health_Fair                               308854 non-null  boo
l
 16  General_Health_Good                               308854 non-null  boo
l
 17  General_Health_Poor                               308854 non-null  boo
l
 18  General_Health_Very Good                          308854 non-null  boo
l
 19  Checkup_5 or more years ago                       308854 non-null  boo
l
 20  Checkup_Never                                     308854 non-null  boo
l
 21  Checkup_Within the past 2 years                   308854 non-null  boo
l
 22  Checkup_Within the past 5 years                   308854 non-null  boo
l
 23  Checkup_Within the past year                      308854 non-null  boo
l
 24  Sex_Female                                        308854 non-null  boo
l
 25  Sex_Male                                          308854 non-null  boo
l
 26  Diabetes_No                                       308854 non-null  boo
```

```
l
 27  Diabetes_No, pre-diabetes or borderline diabetes     308854 non-null  boo
l
 28  Diabetes_Yes                                         308854 non-null  boo
l
 29  Diabetes_Yes, but female told only during pregnancy  308854 non-null  boo
l
 30  Age_Category_18-24                                   308854 non-null  boo
l
 31  Age_Category_25-29                                   308854 non-null  boo
l
 32  Age_Category_30-34                                   308854 non-null  boo
l
 33  Age_Category_35-39                                   308854 non-null  boo
l
 34  Age_Category_40-44                                   308854 non-null  boo
l
 35  Age_Category_45-49                                   308854 non-null  boo
l
 36  Age_Category_50-54                                   308854 non-null  boo
l
 37  Age_Category_55-59                                   308854 non-null  boo
l
 38  Age_Category_60-64                                   308854 non-null  boo
l
 39  Age_Category_65-69                                   308854 non-null  boo
l
 40  Age_Category_70-74                                   308854 non-null  boo
l
 41  Age_Category_75-79                                   308854 non-null  boo
l
 42  Age_Category_80+                                     308854 non-null  boo
l
dtypes: bool(29), float64(7), int64(7)
memory usage: 41.5 MB
```

In [ ]:
```
#Drop one variable of each dummies to avoid dummies variables trap
df.drop(['General_Health_Poor', 'Checkup_Never', 'Sex_Female', 'Diabetes_No',
```

In [ ]:
```
#Replace the boolean value with 1 and 0
df = df.replace({True: 1, False: 0})
```

Hypothesis 1: Body Measurement might be a causation of Heart Disease

In [ ]:
```
#Filter the data for the column Height, Weight and BMT
df_measurement = df[['Height_(cm)', 'Weight_(kg)', 'BMI']]
df_measurement = add_constant(df_measurement)
```

In [ ]:
```
#Run the Logistic Regression for the target variable Heart Disease
#and the predictor variables Height, Weight, and BMT
model = Logit(df['Heart_Disease'], df_measurement)
result = model.fit()
result.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.279822
         Iterations 7
```

Out[ ]:

Model:	Logit	Method:	MLE
Dependent Variable:	Heart_Disease	Pseudo R-squared:	0.004
Date:	2023-12-06 16:52	AIC:	172856.5006
No. Observations:	308854	BIC:	172899.0631
Df Model:	3	Log-Likelihood:	-86424.
Df Residuals:	308850	LL-Null:	-86739.
Converged:	1.0000	LLR p-value:	3.7502e-136
No. Iterations:	7.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-3.0984	0.4189	-7.3958	0.0000	-3.9195	-2.2773
Height_(cm)	0.0000	0.0025	0.0109	0.9913	-0.0048	0.0049
Weight_(kg)	0.0057	0.0023	2.4697	0.0135	0.0012	0.0103
BMI	0.0060	0.0067	0.8909	0.3730	-0.0072	0.0192

Interpretation:

The intercept's (coef ~ -3.09) z-value of -7.3958 indicates that it is significantly different from zero. It represents the log-odds of the baseline category when all predictors are zero.

Height*(cm) (coef ~ 0) z-value is 0.0109, which is very close to zero. This suggests that the coefficient for Height*(cm) is not significantly different from zero, and this predictor may not have a significant impact on the log-odds of heart disease.

Weight*(kg) (coef ~ .0057) z-value is 2.4697, indicating that the coefficient for Weight*(kg) is significantly different from zero. A positive coefficient suggests that an increase in weight is associated with an increase in the log-odds of heart disease. BMI: 0.8909

BMI (coef ~ .0.006) z-value is 0.8909, suggesting that the coefficient for BMI is not significantly different from zero. This implies that BMI may not have a statistically significant impact on the log-odds of heart disease.

All predictors have associated p-values less than 0.05, indicating that they are statistically significant in predicting heart disease.

Pseudo R-squared: 0.004

The Pseudo R-squared provides a measure of how well the model explains the variability in the dependent variable. In this case, the model accounts for a small percentage of the variability in heart disease.

AIC (Akaike Information Criterion): 172856.5006

BIC (Bayesian Information Criterion): 172899.0631

AIC and BIC are information criteria that balance the goodness of fit with the complexity of the model. Lower values indicate a better-fitting model.

This model, though statistically significant, explains a small portion of the variability in heart disease. The predictors (Height, Weight, BMI) have statistically significant effects on the log-odds of having heart disease. However, the pseudo R-squared suggests that there may be other unaccounted factors influencing heart disease. Further investigation and model refinement may be needed.

Examining whether food consumption would affect on Heart Disease

In [ ]:
```
df_consumption = df[['Alcohol_Consumption', 'Fruit_Consumption', 'Green_Vegetal
df_consumption= add_constant(df_consumption)
model = Logit(df['Heart_Disease'], df_consumption)
result = model.fit()
result.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.279702
         Iterations 7
```

Out[ ]:

Model:	Logit	Method:	MLE
Dependent Variable:	Heart_Disease	Pseudo R-squared:	0.004
Date:	2023-12-06 16:52	AIC:	172784.3300
No. Observations:	308854	BIC:	172837.5331
Df Model:	4	Log-Likelihood:	-86387.
Df Residuals:	308849	LL-Null:	-86739.
Converged:	1.0000	LLR p-value:	5.1713e-151
No. Iterations:	7.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-2.1795	0.0131	-166.2930	0.0000	-2.2052	-2.1538
Alcohol_Consumption	-0.0179	0.0009	-19.5639	0.0000	-0.0197	-0.0161
Fruit_Consumption	-0.0025	0.0003	-8.6726	0.0000	-0.0030	-0.0019
Green_Vegetables_Consumption	-0.0047	0.0005	-9.4592	0.0000	-0.0057	-0.0038
FriedPotato_Consumption	-0.0045	0.0008	-5.3474	0.0000	-0.0062	-0.0029

Interpretation:

Based on the interpretation for the model above, we make the overall interpretation for this model as below:

- The model is statistically significant, and all predictors have associated p-values less than 0.05.
- The predictors (Alcohol Consumption, Fruit Consumption, Green Vegetables Consumption, Fried Potato Consumption) have statistically significant effects on the log-odds of having heart disease.
- The Pseudo R-squared suggests that the model explains a small percentage of the variability in heart disease.
- AIC and BIC values are relatively low, indicating a good balance between model fit and complexity.
- Despite statistical significance, the model explains a small portion of heart disease variability, and there may be other factors influencing heart disease not captured by the current predictors. Further investigation and model refinement are recommended.

Hypothesis 2: General Health Condition and Frequency Health Checkup will have effect on Heart Condition.

In [ ]:
```
predictors = df[['General_Health_Excellent','General_Health_Very Good', 'Genera
predictors = add_constant(predictors)
m = Logit(df['Heart_Disease'], predictors)
m = m.fit()
m.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.251391
         Iterations 8
```

Out[ ]:

Model:	Logit	Method:	MLE
Dependent Variable:	Heart_Disease	Pseudo R-squared:	0.105
Date:	2023-12-06 16:52	AIC:	155304.2071
No. Observations:	308854	BIC:	155399.9728
Df Model:	8	Log-Likelihood:	-77643.
Df Residuals:	308845	LL-Null:	-86739.
Converged:	1.0000	LLR p-value:	0.0000
No. Iterations:	8.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.5222	0.1380	-11.0299	0.0000	-1.7927	-1.2517
General_Health_Excellent	-3.0312	0.0365	-83.0388	0.0000	-3.1028	-2.9597
General_Health_Very Good	-2.2647	0.0251	-90.1014	0.0000	-2.3140	-2.2155
General_Health_Good	-1.4993	0.0233	-64.4058	0.0000	-1.5449	-1.4537
General_Health_Fair	-0.6710	0.0244	-27.4524	0.0000	-0.7189	-0.6231
Checkup_5 or more years ago	-0.3708	0.1476	-2.5126	0.0120	-0.6601	-0.0816
Checkup_Within the past 2 years	0.0885	0.1395	0.6347	0.5256	-0.1848	0.3618
Checkup_Within the past 5 years	-0.2908	0.1447	-2.0096	0.0445	-0.5745	-0.0072
Checkup_Within the past year	0.8581	0.1370	6.2646	0.0000	0.5897	1.1266

Interpretation:

- The model is statistically significant, and most predictors have highly significant effects on the log-odds of heart disease, except checkup_5 or more year ago, Checkup_Within the past 2 years, and Check_Within the past 5 years (0 <|z|< 2.6)
- The Pseudo R-squared suggests that the model explains a substantial percentage of the variability in heart disease.
- AIC and BIC values are relatively low, indicating a good balance between model fit and complexity.
- This model provides valuable insights into the relationships between general health status, checkup frequency, and the likelihood of heart disease.

In [ ]:
```
#Examine the distribution of sex on heart disease
sex = df['Sex_Male']
sex = add_constant(sex)
s = Logit(df['Heart_Disease'], sex)
s = s.fit()
s.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.278197
         Iterations 7
```

Out[ ]:

Model:	Logit	Method:	MLE
Dependent Variable:	Heart_Disease	Pseudo R-squared:	0.009
Date:	2023-12-06 16:52	AIC:	171848.4956
No. Observations:	308854	BIC:	171869.7768
Df Model:	1	Log-Likelihood:	-85922.
Df Residuals:	308852	LL-Null:	-86739.
Converged:	1.0000	LLR p-value:	0.0000
No. Iterations:	7.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-2.7203	0.0104	-262.1436	0.0000	-2.7406	-2.6999
Sex_Male	0.5385	0.0135	39.9663	0.0000	0.5120	0.5649

Hypothesis 3: Age Categories are correlated with Heart Disease

In [ ]:

```
age = df[['Age_Category_25-29', 'Age_Category_30-34', 'Age_Category_35-39', 'A
age = add_constant(age)
a = Logit(df['Heart_Disease'], age)
a = a.fit()
a.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.250110
         Iterations 9
```

Out[ ]:

Logit Regression Results

Dep. Variable:	Heart_Disease	**No. Observations:**	308854
Model:	Logit	**Df Residuals:**	308841
Method:	MLE	**Df Model:**	12
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	0.1094
Time:	16:52:18	**Log-Likelihood:**	-77247.
converged:	True	**LL-Null:**	-86739.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-5.2869	0.103	-51.129	0.000	-5.490	-5.084
Age_Category_25-29	0.3734	0.140	2.667	0.008	0.099	0.648
Age_Category_30-34	0.7796	0.125	6.217	0.000	0.534	1.025
Age_Category_35-39	0.9801	0.120	8.170	0.000	0.745	1.215
Age_Category_40-44	1.4024	0.114	12.282	0.000	1.179	1.626
Age_Category_45-49	1.8882	0.111	17.083	0.000	1.672	2.105
Age_Category_50-54	2.2787	0.108	21.175	0.000	2.068	2.490
Age_Category_55-59	2.7150	0.106	25.617	0.000	2.507	2.923
Age_Category_60-64	3.0083	0.105	28.608	0.000	2.802	3.214
Age_Category_65-69	3.2398	0.105	30.908	0.000	3.034	3.445
Age_Category_70-74	3.5257	0.105	33.695	0.000	3.321	3.731
Age_Category_75-79	3.7788	0.105	36.000	0.000	3.573	3.984
Age_Category_80+	4.0098	0.105	38.310	0.000	3.805	4.215

Interpretation:

- The model is statistically significant, and age categories are strongly associated with the likelihood of heart disease.
- The Pseudo R-squared suggests that the model explains a reasonable proportion of the variability in heart disease.
- Age is a significant predictor, with older age categories having higher log-odds of heart disease.
- The LLR p-value indicates that the model as a whole is a better fit than a null model.
- This model provides insights into the relationship between age and the likelihood of heart disease. Older age categories are associated with a higher likelihood of heart disease, as reflected in the positive and significant coefficients for each age category.

Hypothesis 4: Lifestyle could affect the heart condition

In [ ]:
```
lifestyle = df[['Exercise', 'Smoking_History']]
lifestyle = add_constant(lifestyle)
ls = Logit(df['Heart_Disease'], lifestyle)
ls = ls.fit()
ls.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.271799
         Iterations 7
```

Out[ ]:

Logit Regression Results

Dep. Variable:	Heart_Disease	**No. Observations:**	308854
Model:	Logit	**Df Residuals:**	308851
Method:	MLE	**Df Model:**	2
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	0.03220
Time:	16:52:19	**Log-Likelihood:**	-83946.
converged:	True	**LL-Null:**	-86739.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.3137	0.014	-163.346	0.000	-2.342	-2.286
Exercise	-0.6606	0.014	-46.878	0.000	-0.688	-0.633
Smoking_History	0.7333	0.014	54.309	0.000	0.707	0.760

Interpretation:

- The model is statistically significant, suggesting that exercise and smoking history are associated with the likelihood of heart disease.
- The Pseudo R-squared indicates that the model explains a modest proportion of the variability in heart disease.
- Exercise has a negative coefficient, indicating a negative association with heart disease.
- Smoking history has a positive coefficient, indicating a positive association with heart disease.
- Both predictors are highly statistically significant.
- This model provides insights into the relationship between exercise, smoking history, and the likelihood of heart disease, highlighting the significance of these factors in predicting heart disease.

Hypothesis 5: Diagnosed with other diseases would relate with heart disease

In [ ]:
```
otherDisease = df[['Skin_Cancer', 'Other_Cancer', 'Depression', 'Diabetes_No,
otherDisease = add_constant(otherDisease)
od = Logit(df['Heart_Disease'], otherDisease)
od = od.fit()
od.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.256713
         Iterations 7
```

Out[ ]:

Logit Regression Results

Dep. Variable:	Heart_Disease	**No. Observations:**	308854
Model:	Logit	**Df Residuals:**	308846
Method:	MLE	**Df Model:**	7
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	0.08592
Time:	16:52:20	**Log-Likelihood:**	-79287.
converged:	True	**LL-Null:**	-86739.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-3.2351	0.011	-281.640	0.000	-3.258	-3.213
Skin_Cancer	0.5761	0.019	31.074	0.000	0.540	0.612
Other_Cancer	0.5291	0.019	28.584	0.000	0.493	0.565
Depression	0.0923	0.016	5.701	0.000	0.061	0.124
Diabetes_No, pre-diabetes or borderline diabetes	0.5483	0.039	13.944	0.000	0.471	0.625
Diabetes_Yes	1.2165	0.015	79.834	0.000	1.187	1.246
Diabetes_Yes, but female told only during pregnancy	-0.4863	0.105	-4.633	0.000	-0.692	-0.281
Arthritis	0.8553	0.014	60.827	0.000	0.828	0.883

Interpretation:

- The model is statistically significant, suggesting that a history of skin cancer, other cancers, depression, diabetes, and arthritis are associated with the likelihood of heart disease.
- The Pseudo R-squared indicates that the model explains a moderate proportion of the variability in heart disease.
- All predictors are highly statistically significant, implying their strong association with heart disease.

In [ ]:
```
X = df[['Weight_(kg)','Alcohol_Consumption', 'Fruit_Consumption', 'Green_Vegeta
Y = df['Heart_Disease']
```

In [ ]:
```
X_const = add_constant(X)
model = Logit(Y, X_const)
model = model.fit()
model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.221805
         Iterations 9
```

Out[ ]:

Logit Regression Results

Dep. Variable:	Heart_Disease	**No. Observations:**	308854
Model:	Logit	**Df Residuals:**	308822
Method:	MLE	**Df Model:**	31
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	0.2102
Time:	16:52:22	**Log-Likelihood:**	-68505.
converged:	True	**LL-Null:**	-86739.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.4859	0.083	-54.147	0.000	-4.648	-4.323
Weight_(kg)	-0.0002	0.000	-0.403	0.687	-0.001	0.001
Alcohol_Consumption	-0.0101	0.001	-11.009	0.000	-0.012	-0.008
Fruit_Consumption	1.603e-05	0.000	0.052	0.959	-0.001	0.001
Green_Vegetables_Consumption	0.0008	0.001	1.478	0.140	-0.000	0.002
FriedPotato_Consumption	-0.0007	0.001	-0.829	0.407	-0.002	0.001
Checkup_Within the past year	0.3534	0.024	14.720	0.000	0.306	0.400
General_Health_Excellent	-2.2593	0.040	-56.595	0.000	-2.338	-2.181
General_Health_Fair	-0.5420	0.026	-20.495	0.000	-0.594	-0.490
General_Health_Good	-1.1814	0.026	-45.517	0.000	-1.232	-1.131
General_Health_Very Good	-1.7425	0.029	-60.902	0.000	-1.799	-1.686
Exercise	-0.0226	0.016	-1.377	0.169	-0.055	0.010
Sex_Male	0.7611	0.016	46.225	0.000	0.729	0.793
Age_Category_30-34	0.4834	0.100	4.836	0.000	0.288	0.679
Age_Category_35-39	0.6043	0.093	6.487	0.000	0.422	0.787
Age_Category_40-44	0.9366	0.086	10.924	0.000	0.769	1.105
Age_Category_45-49	1.3162	0.081	16.268	0.000	1.158	1.475
Age_Category_50-54	1.6315	0.077	21.216	0.000	1.481	1.782
Age_Category_55-59	1.9704	0.075	26.379	0.000	1.824	2.117
Age_Category_60-64	2.2095	0.074	30.019	0.000	2.065	2.354
Age_Category_65-69	2.4531	0.073	33.492	0.000	2.310	2.597
Age_Category_70-74	2.7152	0.073	37.099	0.000	2.572	2.859
Age_Category_75-79	2.9557	0.074	39.969	0.000	2.811	3.101
Age_Category_80+	3.2326	0.074	43.858	0.000	3.088	3.377
Smoking_History	0.3926	0.015	26.540	0.000	0.364	0.422
Skin_Cancer	0.1176	0.020	5.961	0.000	0.079	0.156

Other_Cancer	0.0468	0.019	2.408	0.016	0.009	0.085
Depression	0.2472	0.018	13.645	0.000	0.212	0.283
Diabetes_No, pre-diabetes or borderline diabetes	0.1562	0.041	3.794	0.000	0.076	0.237
Diabetes_Yes	0.5410	0.017	31.991	0.000	0.508	0.574
Diabetes_Yes, but female told only during pregnancy	0.1357	0.109	1.243	0.214	-0.078	0.350
Arthritis	0.2651	0.015	17.299	0.000	0.235	0.295

Note:

Some predictor variables's z-value is closed to zero (0<|z|<2). Hence, we eliminate these variables and re-run the model afterward

In [ ]:
```
X = df[['Checkup_Within the past year','General_Health_Excellent', 'General_Hea
Y = df['Heart_Disease']
```

In [ ]:
```
X_const = add_constant(X)
model = Logit(Y, X_const)
model = model.fit()
model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.222017
         Iterations 9
```

Out[ ]:

Logit Regression Results

Dep. Variable:	Heart_Disease	**No. Observations:**	308854
Model:	Logit	**Df Residuals:**	308829
Method:	MLE	**Df Model:**	24
Date:	Wed, 06 Dec 2023	**Pseudo R-squ.:**	0.2095
Time:	16:52:25	**Log-Likelihood:**	-68571.
converged:	True	**LL-Null:**	-86739.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-4.5123	0.077	-58.775	0.000	-4.663	-4.362
Checkup_Within the past year	0.3587	0.024	14.961	0.000	0.312	0.406
General_Health_Excellent	-2.2979	0.039	-58.813	0.000	-2.374	-2.221
General_Health_Fair	-0.5515	0.026	-20.985	0.000	-0.603	-0.500
General_Health_Good	-1.2030	0.025	-47.288	0.000	-1.253	-1.153
General_Health_Very Good	-1.7777	0.028	-64.118	0.000	-1.832	-1.723
Sex_Male	0.7301	0.015	48.681	0.000	0.701	0.760
Age_Category_30-34	0.4795	0.100	4.799	0.000	0.284	0.675
Age_Category_35-39	0.6024	0.093	6.471	0.000	0.420	0.785
Age_Category_40-44	0.9348	0.086	10.916	0.000	0.767	1.103
Age_Category_45-49	1.3172	0.081	16.302	0.000	1.159	1.476
Age_Category_50-54	1.6302	0.077	21.224	0.000	1.480	1.781
Age_Category_55-59	1.9694	0.075	26.397	0.000	1.823	2.116
Age_Category_60-64	2.2075	0.074	30.029	0.000	2.063	2.352
Age_Category_65-69	2.4504	0.073	33.501	0.000	2.307	2.594
Age_Category_70-74	2.7127	0.073	37.123	0.000	2.569	2.856
Age_Category_75-79	2.9550	0.074	40.040	0.000	2.810	3.100
Age_Category_80+	3.2366	0.073	44.082	0.000	3.093	3.381
Smoking_History	0.3767	0.015	25.716	0.000	0.348	0.405
Skin_Cancer	0.1090	0.020	5.537	0.000	0.070	0.148
Other_Cancer	0.0446	0.019	2.297	0.022	0.007	0.083
Depression	0.2472	0.018	13.673	0.000	0.212	0.283
Diabetes_No, pre-diabetes or borderline diabetes	0.1663	0.041	4.047	0.000	0.086	0.247
Diabetes_Yes	0.5624	0.016	34.126	0.000	0.530	0.595
Arthritis	0.2631	0.015	17.294	0.000	0.233	0.293

Overall interpretation

The model is statistically significant and explains a substantial proportion of the variability in heart disease.

Several factors, including age, sex, alcohol consumption, general health, and various health conditions, are associated with the likelihood of heart disease.

Smoking History, Skin Cancer, Other Cancer, Depression, Diabetes (No, pre-diabetes or borderline diabetes, Yes), Arthritis:

- Positive coefficients, indicating an association with higher log-odds of heart disease.
- All are highly statistically significant.

Coefficients provide insights into the direction and strength of these associations.

The high statistical significance of coefficients suggests robust relationships between predictors and heart diseas

Pseudo R-squared: 0.2102 - Indicates the model explains a substantial percentage of the variability in heart disease. So far, this model has the highest Pseudo R-Squared Score

Model Performance using Scikit-learn

Data Preprocesing

In [ ]:
```
#Standardize the predictor variables
scaler = StandardScaler()
X = scaler.fit_transform(X)
```

In [ ]:
```
#Split the data into train set and test set
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size= 0.3, rand
print('Train set: ', X_train.shape, Y_train.shape)
print('Test set: ', X_test.shape, Y_test.shape)
```

```
Train set:  (216197, 24) (216197,)
Test set:  (92657, 24) (92657,)
```

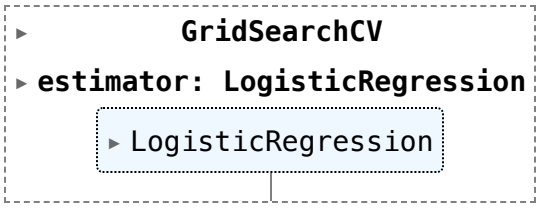
In [ ]:
```
#Compute the sample weight since the data is imbalace
W_train = compute_sample_weight('balanced',Y_train)
```

Logistic Regression classification

In [ ]:
```
#Create a Logistic Regression model
lg = LogisticRegression(random_state=42)
#Define the hyperparameter and their positive values
param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 0, 1, 10, 100]
}
```

```
#Create a grid search object
grid_search = GridSearchCV(lg, param_grid, cv = 5, scoring= 'accuracy')
grid_search.fit(X_train, Y_train, sample_weight = W_train)
```

```
/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_
validation.py:425: FitFailedWarning:
40 fits failed out of a total of 70.
The score on these train-test partitions for these parameters will be set to n
an.
If these failures are not expected, you can try to debug them by setting error
_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
--
30 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/model_sel
ection/_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/base.py",
line 1151, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/linear_mo
del/_logistic.py", line 1168, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/linear_mo
del/_logistic.py", line 56, in _check_solver
    raise ValueError(
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1 penalt
y.

--------------------------------------------------------------------------------
--
10 fits failed with the following error:
Traceback (most recent call last):
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/model_sel
ection/_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/base.py",
line 1144, in wrapper
    estimator._validate_params()
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/base.py",
line 637, in _validate_params
    validate_parameter_constraints(
  File "/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/utils/_pa
ram_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'C' parameter of Lo
gisticRegression must be a float in the range (0.0, inf]. Got 0 instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/Users/nalini/anaconda3/lib/python3.11/site-packages/sklearn/model_selection/_
search.py:976: UserWarning: One or more of the test scores are non-finite: [
nan 0.73633305        nan 0.73598617        nan 0.73595842
        nan        nan        nan 0.73592141        nan 0.73592604
        nan 0.73592604]
  warnings.warn(
```

Out[ ]:


In [ ]:
```
# Get the best param from the grid search
best_param = grid_search.best_params_
best_param
```

Out[ ]:
```
{'C': 0.001, 'penalty': 'l2'}
```

In [ ]:
```
#Get the best model from the grid search
best_model = grid_search.best_estimator_
best_model
```

Out[ ]:
```
LogisticRegression(C=0.001, random_state=42)
```

In [ ]:
```
#Predict the value of target variable
yhat_lg = best_model.predict(X_test)
```

In [ ]:
```
#Compute the accuracy of the model by comparing the predicted value with the a
accuracy_lg = accuracy_score(Y_test, yhat_lg)
print('Accuracy score:', accuracy_lg)
```

```
Accuracy score: 0.7378287663101547
```

In [ ]:
```
#Predict the probability that the target variable will be 1 (the observation is
yhat_prob = best_model.predict_proba(X_test)[:,1]
```

In [ ]:
```
#Compute the Roc-auc score
roc_auc = roc_auc_score(Y_test, yhat_prob)
print('The ROC_AUC score:', roc_auc)
```

```
The ROC_AUC score: 0.8352373261966946
```

Appendix: Decision Tree classification for comparing model performance

In [ ]:
```
#Compute the sample weight since the data is imbalace

W_train = compute_sample_weight('balanced',Y_train)

tree = DecisionTreeClassifier()
#Define the hyperparameter and their positive vaues
param_grid = {
    'criterion': ['gini', 'entropy'],   # Split criterion
    'max_depth': [None, 5, 10, 15],     # Maximum depth of the tree
    'min_samples_split': [2, 5, 10],    # Minimum number of samples required t
    'min_samples_leaf': [1, 2, 4]       # Minimum number of samples required t
}
#Create a grid search object
grid_search = GridSearchCV(tree, param_grid, cv = 5, scoring='accuracy')
grid_search.fit(X_train, Y_train, sample_weight= W_train)
```

Out[ ]:
```
▸ GridSearchCV
▸ estimator: DecisionTreeClassifier
    ▸ DecisionTreeClassifier
```

In [ ]:
```
#Find the best parameter
best_param = grid_search.best_params_
best_param
```

Out[ ]:
```
{'criterion': 'gini',
 'max_depth': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

In [ ]:
```
#Find the best model
best_model = grid_search.best_estimator_
best_model
```

Out[ ]:
```
▾ DecisionTreeClassifier

DecisionTreeClassifier()
```

In [ ]:
```
yhat = best_model.predict(X_test)
```

In [ ]:
```
accuracy = accuracy_score(Y_test, yhat)
print('Accuracy Score',accuracy)
```

```
Accuracy Score 0.7304143237963673
```

In [ ]:
```
yhat_prob = best_model.predict_proba(X_test)[:,1]
```

In [ ]:
```
roc_auc = roc_auc_score(Y_test, yhat_prob)
print('ROC-AUC score:',roc_auc)
```

```
ROC-AUC score: 0.7739396161030319
```

Note:

As the result, the decision tree classfication model have lower ROC-AUC score, therefore logistic regression is still the best model that we chose

In [ ]:
```
```