Shashi Bhushan
05/14/2023
DSC630 – Predictive Analytics
Spring 2023

Project Milestone 5 – Final Project Paper

Introduction

CDC reports that heart disease is one of the leading causes of death for people in the United States with one person dying from it every 34 seconds. In 2020 alone 697,000 people died from heart disease in the United States that was one in every five deaths accounted that year. From financial perspective, the average cost associated with this disease per year was about $229 billion between 2017 and 2019 (Heart Disease Facts, 2022). Therefore, we see that heart disease has significant impact on society in terms of overall wellbeing of the families together with requiring a lot of financial investment to manage the disease if detected. This represents a problem that requires to be solved or a solution being made where its impacts are minimized. To present a solution where the impact is minimized, I propose to propose a model to detect heart disease using parameters collected during regular checkups such as annual or quarterly check-ups with general practitioners.

Data Selection

Personal Key Indicators of Heart Disease, Personal Key Indicators of Heart Disease | Kaggle. The dataset has 18 variables with HeartDisease being the target variable with answers noted as binary (Yes and No). The data was collected in 2020 by Center for Disease Control and Prevention (CDC) by telephone surveys and included 300 variables. The dataset was further cleaned to select 17 factors that directly or indirectly influenced heart disease.

My data set has features such as BMI, smoking, alcohol drinking, history of stroke, physical health score, mental health score, difficulty in walking, gender, age category, race, existence of diabetes, whether pa8ent is physically active, general health condition, usual sleep duration, existence of asthma, kidney disease, and skin cancer. These features are typically more than what is noted during a physical health

check and therefore should be sufficient to train a model for the binary classification problem of existence or not of heart disease in a patient.

Modeling and Methods

Visualizations

I used the following visualizations to explain my data:

- Develop a heatmap showing correlation between features, in particular checking correlation with heart disease.
- Bar chart showing count of males and females having heart disease.
- Bar chart showing counts by races having heart disease.
- Bar chart showing counts by age group having heart disease.
- Bar chart showing count by general health having heart disease.
- Associated pie charts for the above cases showing percentages of people having heart disease in the above categories like males, females, races, age group, general health condition.

Data Preparation

Data was primarily consisting of objects and therefore converted into dummy variables. I checked for missing data, but no missing data was observed. I removed the duplicates.

I removed the redundant variables resulting from the creation of dummy variables for binary features from the data frame.

I checked observations per race and gender categories for assuring normalcy. These categories had over 30 observations each and therefore, assumed to be normally distributed.

ANOVA was used for feature selection. However, all p-values were noted to be less than 0.05 and therefore, no feature was dropped. I also used the standard scaler function to scale the values into a common range.

Shashi Bhushan
05/14/2023
DSC630 – Predictive Analytics
Spring 2023

Modeling

Our target variable, outcome (Yes or No) is binary. I built and evaluated two models namely Logistic Regression and K Nearest Neighbor. Used K as three for KN Classifier model as the model produced optimum results.

Logistic regression is helpful in the prediction of classification problems and involving continuous or discrete predictor variables. It also provides probabilities associated with new data. It also identifies variables that are effective in making predictions.

The nearest neighbor algorithm uses proximity to predict the grouping of an individual data point. Here we have to predict the heart disease based on multiple variables; therefore, nearest neighbor is an appropriate algorithm to use.

Interpretation of Analysis / Model Results

Visualizations

Following are my observations from the visualizations:

- Heart disease is more prevalent in males than females.
- White people had the maximum number in positive heart disease cases. However, this is reflective of the population proportion in the U.S.
- Age does play a role in heart disease as the bar chart showed that people in higher age ranges had more positive heart disease cases.
- The surprise finding was that a lot the highest number of people having heart disease were in good general health.
- Heat map did not show significant correlation of other features with heart disease.

Shashi Bhushan
05/14/2023
DSC630 – Predictive Analytics
Spring 2023

Model Results Interpretation

Three models produced high accuracy percentages with Logistic Regression and Knn producing 91%, and 89% respectively. However, upon generation of the classification reports for the two models, it was observed that though the model precision, recall, and f-1 scores are very high for predicting "No heart disease;" however, these scores are not good for predicting "heart disease." Logistic Regression Classifier has a decent 57% as precision score for prediction “heart disease.”

Conclusion

Since the maximum number of people in the data set having heart disease were in good general health, a prediction model becomes important.

Precision is the proportion of every observation predicted to be positive that is actually positive. Logistic Regression had higher accuracy together with highest precision scores for both predicting “heart disease” and “no heart disease” between the two models created (refer to the classification report below). Therefore, I recommend the use of Logistic Regression classifier model for predicting heart disease.

	precision	recall	f1-score	support
0	0.92	0.99	0.95	54876
1	0.57	0.11	0.19	5468
accuracy			0.91	60344
macro avg	0.74	0.55	0.57	60344
weighted avg	0.89	0.91	0.88	60344

In addition, the Logistic Regression model showed only slight overfitting as the training accuracy was observed to be slightly better than test accuracy.

The dataset I am using is a cleaned version where 17 variables were selected from about 300 variables. I do not have access to the basis on which it was selected and therefore, I am not sure if the basis was

ethically correct, or the data was cleaned to serve any desired outcome. Therefore, for a future study, selection of raw data collected from a reliable source is suggested to be used for model building purposes.

Please refer to the end of file for visualizations, codes, and complete analysis.

Bibliography

Heart Disease Facts. (2022, October 14). Retrieved from CDC: https://www.cdc.gov/heartdisease/facts.htm

In [57]:
```
# Shashi Bhushan
# MSDS DSC 630, Spring 2023
# Project Milestone 4
```

In [58]:
```
# Importing libraries
import pandas as pd
import numpy as np
```

In [59]:
```
# Reading CSV File
df = pd.read_csv('heart_2020_cleaned.csv')
df.head()
```

Out[59]:

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	A
0	No	16.60	Yes	No	No	3.0	30.0	No	Female	
1	No	20.34	No	No	Yes	0.0	0.0	No	Female	
2	No	26.58	Yes	No	No	20.0	30.0	No	Male	
3	No	24.21	No	No	No	0.0	0.0	No	Female	
4	No	23.71	No	No	No	28.0	0.0	Yes	Female	

In [60]:
```
# Exploring data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 319795 entries, 0 to 319794
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   HeartDisease      319795 non-null  object 
 1   BMI               319795 non-null  float64
 2   Smoking           319795 non-null  object 
 3   AlcoholDrinking   319795 non-null  object 
 4   Stroke            319795 non-null  object 
 5   PhysicalHealth    319795 non-null  float64
 6   MentalHealth      319795 non-null  float64
 7   DiffWalking       319795 non-null  object 
 8   Sex               319795 non-null  object 
 9   AgeCategory       319795 non-null  object 
 10  Race              319795 non-null  object 
 11  Diabetic          319795 non-null  object 
 12  PhysicalActivity  319795 non-null  object 
 13  GenHealth         319795 non-null  object 
 14  SleepTime         319795 non-null  float64
 15  Asthma            319795 non-null  object 
 16  KidneyDisease     319795 non-null  object 
 17  SkinCancer        319795 non-null  object 
dtypes: float64(4), object(14)
memory usage: 43.9+ MB
```

In [61]:
```
# finding percentage of missing values in each feature
import numpy as np
for column in df.columns:
    print('{} has {} % missing values'.format(column,np.round(df[column].isnull().sum()/
```

```
HeartDisease has 0.0 % missing values
BMI has 0.0 % missing values
Smoking has 0.0 % missing values
AlcoholDrinking has 0.0 % missing values
Stroke has 0.0 % missing values
```

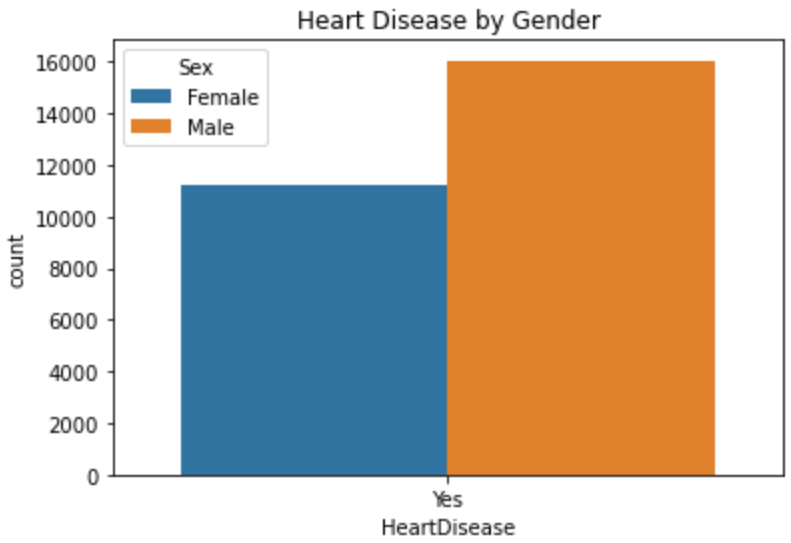
```
PhysicalHealth has 0.0 % missing values
MentalHealth has 0.0 % missing values
DiffWalking has 0.0 % missing values
Sex has 0.0 % missing values
AgeCategory has 0.0 % missing values
Race has 0.0 % missing values
Diabetic has 0.0 % missing values
PhysicalActivity has 0.0 % missing values
GenHealth has 0.0 % missing values
SleepTime has 0.0 % missing values
Asthma has 0.0 % missing values
KidneyDisease has 0.0 % missing values
SkinCancer has 0.0 % missing values
```

In [62]:
```
# dropping duplicate rows if any
df.drop_duplicates(inplace=True)
df.describe()
```

Out[62]:

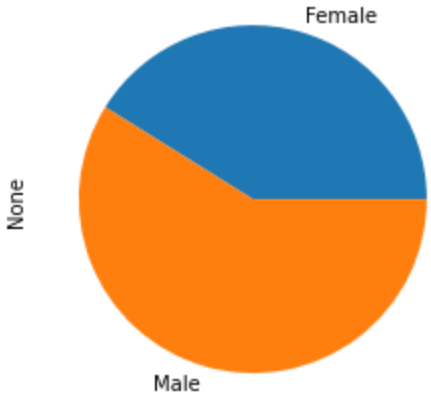
	BMI	PhysicalHealth	MentalHealth	SleepTime
count	301717.000000	301717.000000	301717.000000	301717.000000
mean	28.441970	3.572298	4.121475	7.084559
std	6.468134	8.140656	8.128288	1.467122
min	12.020000	0.000000	0.000000	1.000000
25%	24.030000	0.000000	0.000000	6.000000
50%	27.410000	0.000000	0.000000	7.000000
75%	31.650000	2.000000	4.000000	8.000000
max	94.850000	30.000000	30.000000	24.000000

In [63]:
```
#  Bar chart showing count of males and females having heart disease.
import seaborn as sns
import matplotlib.pyplot as plt
plt.title("Heart Disease by Gender")
df2= df[df.HeartDisease=="Yes"]
sns.countplot(x ='HeartDisease', hue = "Sex", data = df2)
plt.show()
```

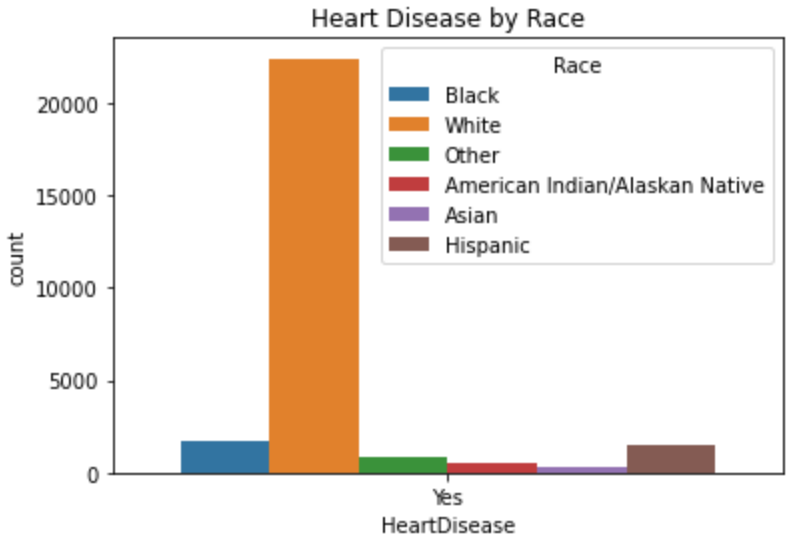


In [64]:
```
# Associated Piechart
df2.groupby('Sex').size().plot.pie()
```

Out[64]:
```
<AxesSubplot:ylabel='None'>
```

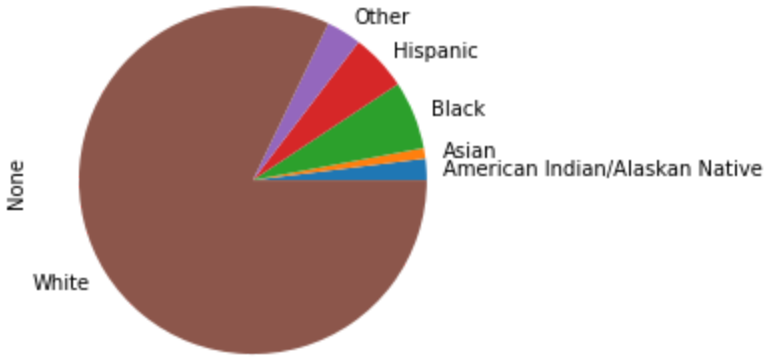


In [65]:
```
# Bar chart showing counts by races having heart disease.
plt.title("Heart Disease by Race")
sns.countplot(x ='HeartDisease', hue = "Race", data = df2)
plt.show()
```

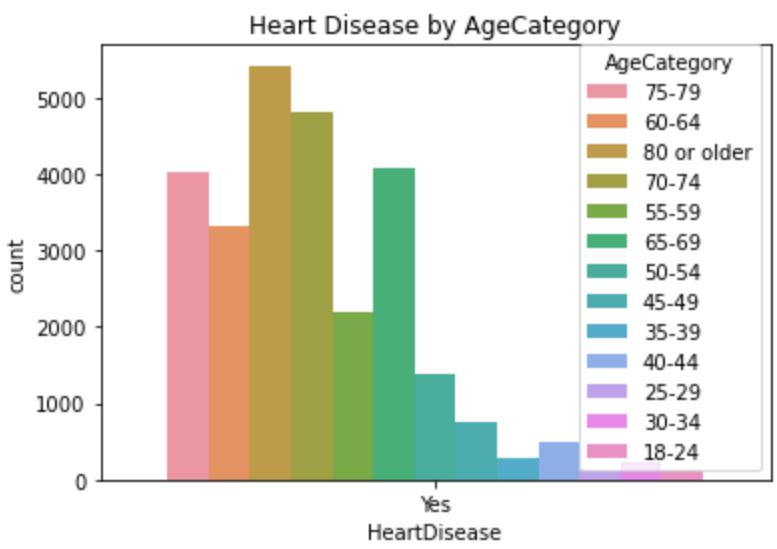


In [66]:
```
# Associated Pie Chart
df2.groupby('Race').size().plot.pie()
```

Out[66]:
```
<AxesSubplot:ylabel='None'>
```

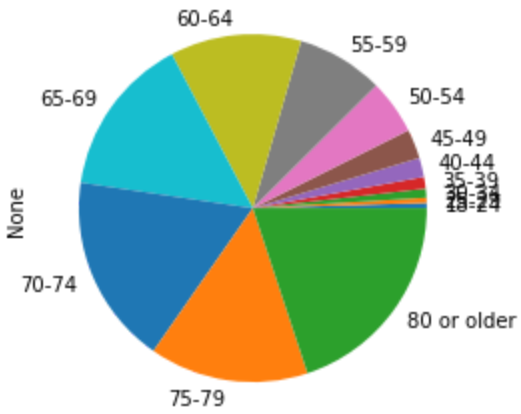


In [67]:
```
# Bar chart showing counts by age group having heart disease
plt.title("Heart Disease by AgeCategory")
sns.countplot(x ='HeartDisease', hue = "AgeCategory", data = df2)
plt.show()
```

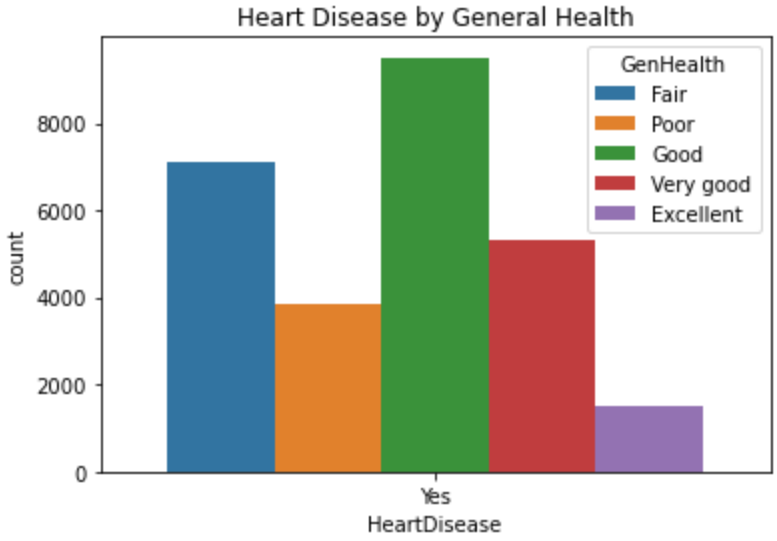


In [68]:
```
# Associated Pie Chart
df2.groupby('AgeCategory').size().plot.pie()
```

Out[68]:
```
<AxesSubplot:ylabel='None'>
```



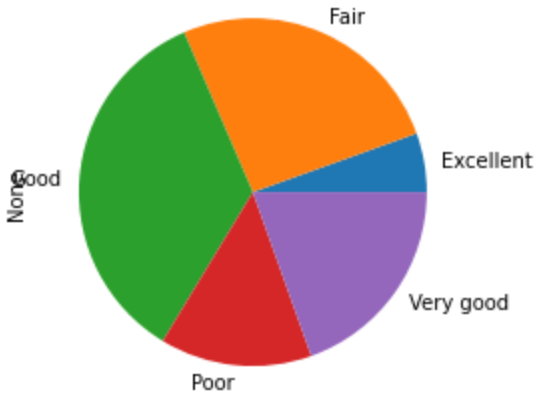
In [69]:
```
# Bar chart showing count by general health having heart disease.
plt.title("Heart Disease by General Health")
sns.countplot(x ='HeartDisease', hue = "GenHealth", data = df2)
plt.show()
```



In [70]:
```
# Associated Pie Chart
df2.groupby('GenHealth').size().plot.pie()
```

```
<AxesSubplot:ylabel='None'>
```

Out[70]:



In [71]:
```
# Checking if sample per race and gender categories for assuring normalcy
df.groupby(['Sex','HeartDisease'])['HeartDisease'].count()
```

Out[71]:
```
Sex     HeartDisease
Female  No             148458
        Yes             11213
Male    No             125998
        Yes             16048
Name: HeartDisease, dtype: int64
```

In [72]:
```
df.groupby(['Race','HeartDisease'])['HeartDisease'].count()
```

Out[72]:
```
Race                             HeartDisease
American Indian/Alaskan Native   No                4650
                                 Yes                542
Asian                            No                7727
                                 Yes                266
Black                            No               21081
                                 Yes               1729
Hispanic                         No               25664
                                 Yes               1443
Other                            No               10005
                                 Yes                886
White                            No              205329
                                 Yes              22395
Name: HeartDisease, dtype: int64
```

In [73]:
```
# All numbers are over 30. Therefore, all categories can be assumed to be normally distr
```

In [74]:
```
# Creating dummy variables for object features
df =pd.get_dummies(df)
df.head()
```

Out[74]:

	BMI	PhysicalHealth	MentalHealth	SleepTime	HeartDisease_No	HeartDisease_Yes	Smoking_No	Smoking_Yes
0	16.60	3.0	30.0	5.0	1	0	0	1
1	20.34	0.0	0.0	7.0	1	0	1	0
2	26.58	20.0	30.0	8.0	1	0	0	1
3	24.21	0.0	0.0	6.0	1	0	1	0
4	23.71	28.0	0.0	8.0	1	0	1	0

5 rows × 52 columns

In [75]:
```
# Removing redundant binary variables
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 301717 entries, 0 to 319794
Data columns (total 52 columns):
 #   Column                                Non-Null Count   Dtype  
---  ------                                --------------   -----  
 0   BMI                                   301717 non-null  float64
 1   PhysicalHealth                        301717 non-null  float64
 2   MentalHealth                          301717 non-null  float64
 3   SleepTime                             301717 non-null  float64
 4   HeartDisease_No                       301717 non-null  uint8  
 5   HeartDisease_Yes                      301717 non-null  uint8  
 6   Smoking_No                            301717 non-null  uint8  
 7   Smoking_Yes                           301717 non-null  uint8  
 8   AlcoholDrinking_No                    301717 non-null  uint8  
 9   AlcoholDrinking_Yes                   301717 non-null  uint8  
 10  Stroke_No                             301717 non-null  uint8  
 11  Stroke_Yes                            301717 non-null  uint8  
 12  DiffWalking_No                        301717 non-null  uint8  
 13  DiffWalking_Yes                       301717 non-null  uint8  
 14  Sex_Female                            301717 non-null  uint8  
 15  Sex_Male                              301717 non-null  uint8  
 16  AgeCategory_18-24                     301717 non-null  uint8  
 17  AgeCategory_25-29                     301717 non-null  uint8  
 18  AgeCategory_30-34                     301717 non-null  uint8  
 19  AgeCategory_35-39                     301717 non-null  uint8  
 20  AgeCategory_40-44                     301717 non-null  uint8  
 21  AgeCategory_45-49                     301717 non-null  uint8  
 22  AgeCategory_50-54                     301717 non-null  uint8  
 23  AgeCategory_55-59                     301717 non-null  uint8  
 24  AgeCategory_60-64                     301717 non-null  uint8  
 25  AgeCategory_65-69                     301717 non-null  uint8  
 26  AgeCategory_70-74                     301717 non-null  uint8  
 27  AgeCategory_75-79                     301717 non-null  uint8  
 28  AgeCategory_80 or older               301717 non-null  uint8  
 29  Race_American Indian/Alaskan Native   301717 non-null  uint8  
 30  Race_Asian                            301717 non-null  uint8  
 31  Race_Black                            301717 non-null  uint8  
 32  Race_Hispanic                         301717 non-null  uint8  
 33  Race_Other                            301717 non-null  uint8  
 34  Race_White                            301717 non-null  uint8  
 35  Diabetic_No                           301717 non-null  uint8  
 36  Diabetic_No, borderline diabetes      301717 non-null  uint8  
 37  Diabetic_Yes                          301717 non-null  uint8  
 38  Diabetic_Yes (during pregnancy)       301717 non-null  uint8  
 39  PhysicalActivity_No                   301717 non-null  uint8  
 40  PhysicalActivity_Yes                  301717 non-null  uint8  
 41  GenHealth_Excellent                   301717 non-null  uint8  
 42  GenHealth_Fair                        301717 non-null  uint8  
 43  GenHealth_Good                        301717 non-null  uint8  
 44  GenHealth_Poor                        301717 non-null  uint8  
 45  GenHealth_Very good                   301717 non-null  uint8  
 46  Asthma_No                             301717 non-null  uint8  
 47  Asthma_Yes                            301717 non-null  uint8  
 48  KidneyDisease_No                      301717 non-null  uint8  
 49  KidneyDisease_Yes                     301717 non-null  uint8  
 50  SkinCancer_No                         301717 non-null  uint8  
 51  SkinCancer_Yes                        301717 non-null  uint8  
dtypes: float64(4), uint8(48)
memory usage: 25.3 MB
```

In [76]:
```
df = df.drop(['HeartDisease_No', 'Smoking_No', 'AlcoholDrinking_No', 'Stroke_No', 'DiffW
```

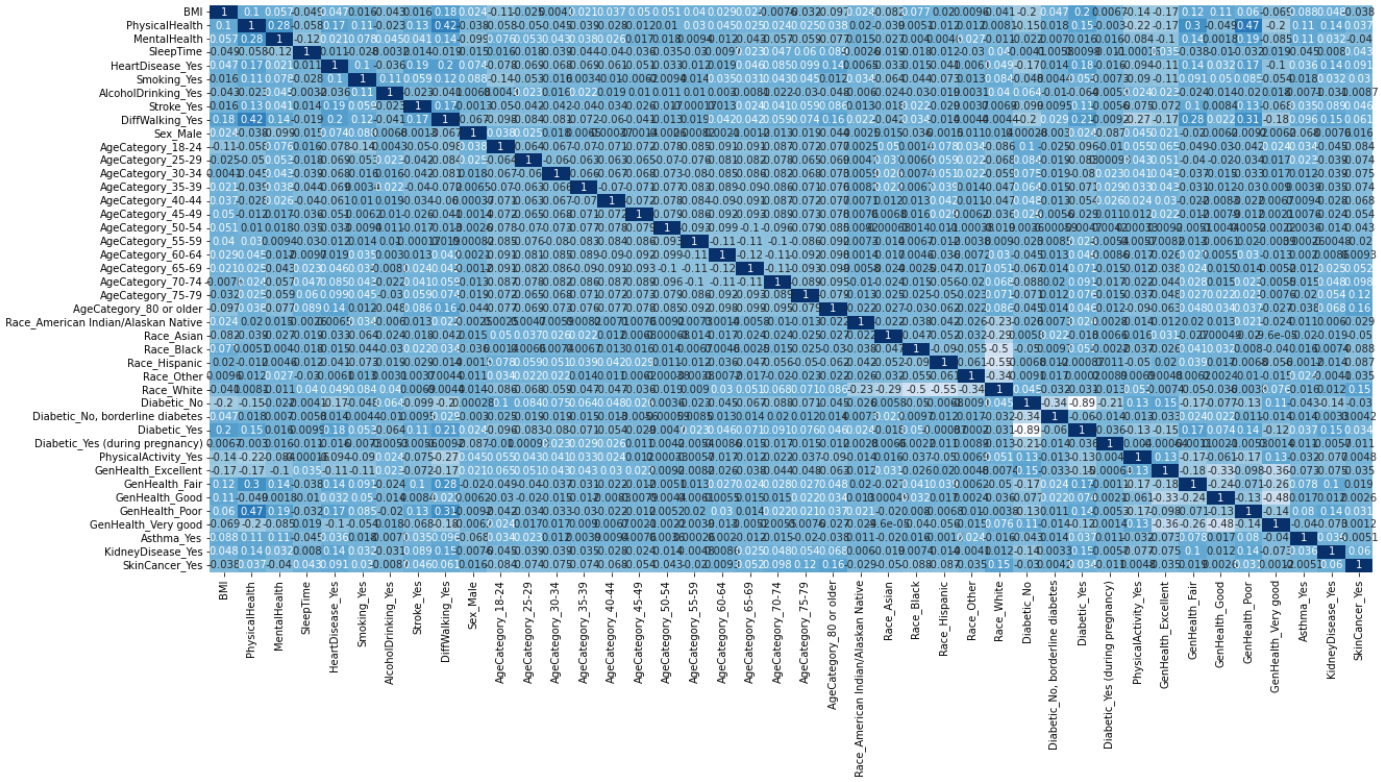
In [77]:
```
# Visualizations
import matplotlib.pyplot as plt
import seaborn as sns
# Develop a heatmap showing correlation between features, in particular checking correla
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True,cbar=False,cmap='Blues')
df.corr()
```

Out[77]:

	BMI	PhysicalHealth	MentalHealth	SleepTime	HeartDisease_Yes	Smoking_Yes	Alcohol
BMI	1.000000	0.103813	0.056724	-0.048653	0.047260	0.015890	
PhysicalHealth	0.103813	1.000000	0.279657	-0.058406	0.165235	0.110270	
MentalHealth	0.056724	0.279657	1.000000	-0.117078	0.020913	0.078364	
SleepTime	-0.048653	-0.058406	-0.117078	1.000000	0.010834	-0.027874	
HeartDisease_Yes	0.047260	0.165235	0.020913	0.010834	1.000000	0.104524	
Smoking_Yes	0.015890	0.110270	0.078364	-0.027874	0.104524	1.000000	
AlcoholDrinking_Yes	-0.043463	-0.023255	0.045421	-0.003172	-0.036289	0.109183	
Stroke_Yes	0.016314	0.132966	0.041324	0.013697	0.194665	0.058868	
DiffWalking_Yes	0.177388	0.422935	0.142964	-0.019155	0.196420	0.115789	
Sex_Male	0.024200	-0.038427	-0.098916	-0.014901	0.074435	0.087514	
AgeCategory_18-24	-0.106688	-0.058372	0.076056	0.016191	-0.077928	-0.139953	
AgeCategory_25-29	-0.024510	-0.049899	0.053466	-0.017985	-0.068546	-0.053471	
AgeCategory_30-34	0.004136	-0.045246	0.042997	-0.038782	-0.068228	-0.016337	
AgeCategory_35-39	0.021282	-0.039207	0.038107	-0.043994	-0.068994	0.003405	
AgeCategory_40-44	0.037156	-0.027583	0.026369	-0.040466	-0.060936	0.010383	
AgeCategory_45-49	0.050075	-0.012211	0.017310	-0.035957	-0.051013	-0.006159	
AgeCategory_50-54	0.051351	0.010235	0.017728	-0.035470	-0.032705	-0.009395	
AgeCategory_55-59	0.040367	0.029865	0.009426	-0.029716	-0.011854	0.013620	
AgeCategory_60-64	0.028590	0.045215	-0.012372	-0.009738	0.018989	0.034508	
AgeCategory_65-69	0.020799	0.024521	-0.042618	0.023445	0.045734	0.031456	
AgeCategory_70-74	-0.007596	0.023549	-0.056634	0.047296	0.084840	0.043120	
AgeCategory_75-79	-0.032469	0.025353	-0.058669	0.059873	0.098552	0.045448	
AgeCategory_80 or older	-0.097425	0.037537	-0.076764	0.089354	0.143466	0.011632	
Race_American Indian/Alaskan Native	0.024378	0.019960	0.015051	-0.002554	0.006480	0.034035	
Race_Asian	-0.081949	-0.039036	-0.027199	-0.018919	-0.032841	-0.064314	
Race_Black	0.077074	0.005113	0.004008	-0.018004	-0.014517	-0.044215	
Race_Hispanic	0.019804	-0.011764	0.004558	-0.011670	-0.040680	-0.073012	
Race_Other	0.009565	0.011739	0.026678	-0.029744	-0.006076	0.013278	
Race_White	-0.041448	0.008125	-0.011455	0.039549	0.048892	0.083659	

Diabetic_No	-0.203952	-0.145269	-0.021528	-0.004090	-0.165966	-0.048180
Diabetic_No, borderline diabetes	0.047106	0.017889	0.007045	-0.005763	0.013793	0.004426
Diabetic_Yes	0.199859	0.151242	0.015977	0.009910	0.178917	0.052500
Diabetic_Yes (during pregnancy)	0.006676	-0.003011	0.016415	-0.010997	-0.015508	-0.007271
PhysicalActivity_Yes	-0.144441	-0.224121	-0.084274	-0.000157	-0.093597	-0.089864
GenHealth_Excellent	-0.172330	-0.170597	-0.104461	0.035006	-0.113218	-0.107082
GenHealth_Fair	0.122887	0.297801	0.143266	-0.038369	0.143265	0.091346
GenHealth_Good	0.110611	-0.049225	0.001844	-0.009956	0.031748	0.050007
GenHealth_Poor	0.059747	0.470076	0.188198	-0.031684	0.172437	0.084953
GenHealth_Very good	-0.068584	-0.196522	-0.085304	0.018630	-0.100540	-0.053652
Asthma_Yes	0.087563	0.110083	0.105266	-0.045368	0.035784	0.017545
KidneyDisease_Yes	0.047796	0.138219	0.032105	0.008028	0.142672	0.031890
SkinCancer_Yes	-0.038060	0.036753	-0.040214	0.043241	0.090644	0.030438

42 rows × 42 columns



In [78]:
```
#  Bar chart showing count of males and females having heart disease.
# Performing feature selection using ANOVA using HeartDisease_Yes as target
y = df['HeartDisease_Yes']
X = df.drop(['HeartDisease_Yes'], axis=1)
```

In [79]:
```
from sklearn.feature_selection import f_regression, SelectKBest
fs=SelectKBest(score_func=f_regression,k=41)
fit=fs.fit(X,y)
```

In [80]:
```
features_score = pd.DataFrame(fit.scores_)
```

```
features_pvalue = pd.DataFrame(np.round(fit.pvalues_,4))
features = pd.DataFrame(X.columns)
feature_score = pd.concat([features,features_score,features_pvalue],axis=1)
feature_score.columns = ["Input_Features", "F_Score", "P_Value"]
print(feature_score.nlargest(49,columns="F_Score"))
```

	Input_Features	F_Score	P_Value
7	DiffWalking_Yes	12107.521989	0.0000
6	Stroke_Yes	11883.653926	0.0000
30	Diabetic_Yes	9977.628745	0.0000
36	GenHealth_Poor	9246.264913	0.0000
28	Diabetic_No	8546.074566	0.0000
1	PhysicalHealth	8468.849439	0.0000
21	AgeCategory_80 or older	6340.524990	0.0000
34	GenHealth_Fair	6322.399758	0.0000
39	KidneyDisease_Yes	6269.072727	0.0000
33	GenHealth_Excellent	3917.668974	0.0000
4	Smoking_Yes	3332.705949	0.0000
37	GenHealth_Very good	3080.976499	0.0000
20	AgeCategory_75-79	2959.167902	0.0000
32	PhysicalActivity_Yes	2666.493472	0.0000
40	SkinCancer_Yes	2499.555671	0.0000
19	AgeCategory_70-74	2187.431329	0.0000
9	AgeCategory_18-24	1843.431534	0.0000
8	Sex_Male	1680.969613	0.0000
12	AgeCategory_35-39	1443.065762	0.0000
10	AgeCategory_25-29	1424.327914	0.0000
11	AgeCategory_30-34	1411.075806	0.0000
13	AgeCategory_40-44	1124.509650	0.0000
14	AgeCategory_45-49	787.197590	0.0000
27	Race_White	722.964231	0.0000
0	BMI	675.390042	0.0000
18	AgeCategory_65-69	632.389630	0.0000
25	Race_Hispanic	500.113052	0.0000
5	AlcoholDrinking_Yes	397.840814	0.0000
38	Asthma_Yes	386.834991	0.0000
23	Race_Asian	325.757156	0.0000
15	AgeCategory_50-54	323.069657	0.0000
35	GenHealth_Good	304.414290	0.0000
2	MentalHealth	132.012696	0.0000
17	AgeCategory_60-64	108.835458	0.0000
31	Diabetic_Yes (during pregnancy)	72.580652	0.0000
24	Race_Black	63.596328	0.0000
29	Diabetic_No, borderline diabetes	57.407372	0.0000
16	AgeCategory_55-59	42.402526	0.0000
3	SleepTime	35.415915	0.0000
22	Race_American Indian/Alaskan Native	12.668185	0.0004
26	Race_Other	11.138931	0.0008

In [81]:
```
# We see that all features have P-Value of less than 0.05. Therefore, all features are s
```

In [82]:
```
# Importing the train_test_split Function and splitting dataset
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_sc
```

In [83]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2)
```

In [84]:
```
#I use the standard scaler function to scale the values into a common range.Then I build
scaler = StandardScaler()
lr = LogisticRegression()
scaler.fit(X_train)
X_train_scaler = scaler.transform(X_train)
X_test_scaler = scaler.transform(X_test)
```

In [85]:
```
# Train Logistic Regression on the training data
lr.fit(X_train_scaler,y_train)
```

Out[85]:
```
LogisticRegression()
```

In [86]:
```
#Evaluationg the model using accuracy score
y_test_pred = lr.predict(X_test_scaler)
accuracy = accuracy_score( y_test, y_test_pred)
accuracy
```

Out[86]:
```
0.9099993371337664
```

In [87]:
```
# Accuracy is 91%
```

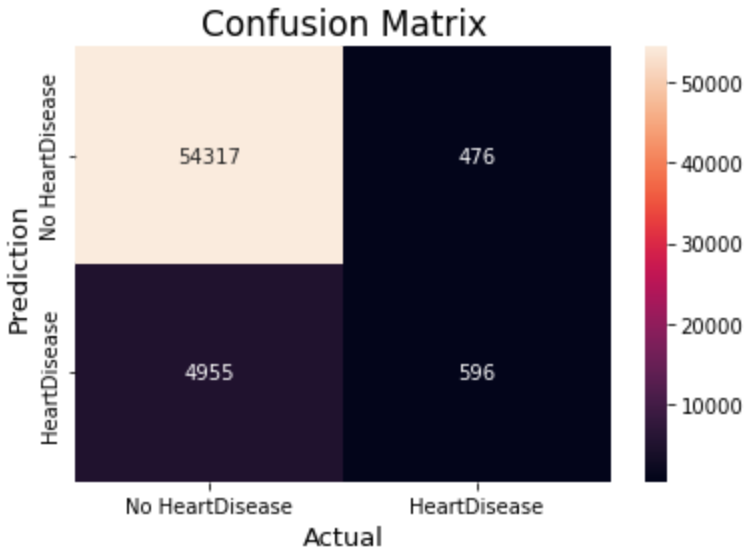
In [88]:
```
#Accuracy on model training data
y_train_pred =lr.predict(X_train_scaler)
accuracy = accuracy_score( y_train, y_train_pred)
accuracy
```

Out[88]:
```
0.9116346898783211
```

In [89]:
```
# Accuracy is 91%. #Accuracy is slightly higher to that calculated above. Therefore slig
```

In [90]:
```
# Confusion matrix plot
cm=confusion_matrix(y_test, y_test_pred)
```

In [91]:
```
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['No HeartDisease','HeartDisease'],
            yticklabels=['No HeartDisease','HeartDisease'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



In [92]:
```
print(classification_report(y_test, y_test_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.92      0.99      0.95     54793
           1       0.56      0.11      0.18      5551
```

```
    accuracy                           0.91     60344
   macro avg       0.74      0.55      0.57     60344
weighted avg       0.88      0.91      0.88     60344
```

In [93]:
```
'''We see that while the accuracy of the model is good, its performance is not very good for the precision, recall and f1-score for "1."'''
```

Out[93]:
```
'We see that while the accuracy of the model is good, its performance is not very good while predicting heart diseases as seen\nfor the precision, recall and f1-score for "1."'
```

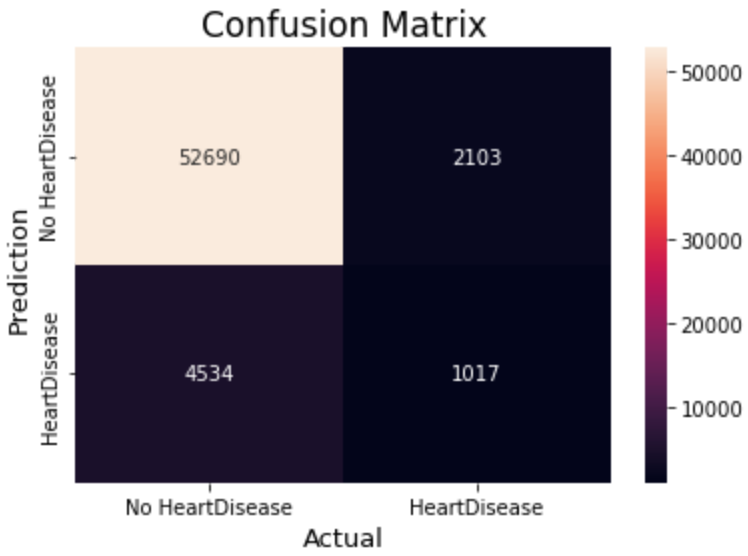
In [94]:
```
# Now training on KNN model
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train_scaler,y_train)
```

Out[94]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [95]:
```
#Evaluationg the model using accuracy score
y_pred = knn.predict(X_test_scaler)
accuracy = accuracy_score(y_test, y_pred)
accuracy
```

Out[95]:
```
0.8900139201909055
```

In [96]:
```
# Confusion matrix plot
cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['No HeartDisease','HeartDisease'],
            yticklabels=['No HeartDisease','HeartDisease'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```



In [97]:
```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.92      0.96      0.94     54793
           1       0.33      0.18      0.23      5551

    accuracy                           0.89     60344
   macro avg       0.62      0.57      0.59     60344
```

```
weighted avg       0.87      0.89      0.88     60344
```

In [98]:
```
'''We see that while the accuracy of the model is good; however, its performance is not
diseases as seen for the precision, recall and f1-score for "1."'''
```

Out[98]:
```
'We see that while the accuracy of the model is good; however, its performance is not ve
ry good while predicting heart \ndiseases as seen for the precision, recall and f1-score
for "1."'
```

In [99]:
```
# Explain your process for prepping the data
'''
Data was primarily consisting of objects and therefore converted into dummy variables.
I checked for missing data but no missing data was observed.
I removed the duplicates.
Created barcharts and piecharts to visualize the data and see representation of categori
ANOVA was used for feature selection. However, all p-values were noted to be less than 0
dropped.
'''
```

Out[99]:
```
'\nData was primarily consisting of objects and therefore converted into dummy variable
s.\nI checked for missing data but no missing data was observed.\nI removed the duplicat
es.\nCreated barcharts and piecharts to visualize the data and see representation of cat
egories such as race and gender.\nANOVA was used for feature selection. However, all p-v
alues were noted to be less than 0.05 and therefore, no feature was\ndropped.\n'
```

In [100…
```
#Build and evaluate at least one model
'''
I built and evaluated two models namely Logistic Regression and K Nearest Neighbor.
Used K as three as the model produced optimum results.
'''
```

Out[100]:
```
'\nI built and evaluated two models namely Logistic Regression and K Nearest Neighbor.\n
Used K as three as the model produced optimum results.\n'
```

In [101…
```
# Interpret your results
'''
Following are my observations from the visualizations:
Heart disease is more prevalent in males than females.
White people had the maximum number in positive heart disease cases. However , this is r
Age does play a role in heart disease as the barchart showed that people in higher age r
The surprise finding was that a lot the highest number of people having heart disease we

Three models produced high accuracy percentages with Logistic Regression  and Knn
producing 91% and 89% respectively. However, upon generation of the classification repor
it was observed that though the model precision, recall, and f-1 scores are very high fo
howver, these scores are not good for predicting "heart disease." Logistic Regression Cl
precision score.
'''
```

Out[101]:
```
'\nFollowing are my observations from the visualizations:\nHeart disease is more prevale
nt in males than females.\nWhite people had the maximum number in positive heart disease
cases. However , this is reflective of the population proportion in the U.S.\nAge does p
lay a role in heart disease as the barchart showed that people in higher age ranges had
more positive heart disease cases.\nThe surprise finding was that a lot the highest numb
er of people having heart disease were in good general health.\n\nThree models produced
high accuracy percentages with Logistic Regression  and Knn  \nproducing 91% and 89% res
pectively. However, upon generation of the classification reports for the two models,\ni
t was observed that though the model precision, recall, and f-1 scores are very high for
predicting "No heart disease;"\nhowver, these scores are not good for predicting "heart
disease." Logistic Regression Classifer has a decent 57% as \nprecision score.\n'
```

In [102…
```
# Initial Conclusions and Recommendations
'''
Since the maximum number of people in the data set having heart disease were in good gen
becomes important.
```

```
Precision is the proportion of every observation predicted to be positive that is actual
recommend use of Logistic Regression classifer for predicting the heart disease.

'''
```

Out[102]:

```
'\nSince the maximum number of people in the data set having heart disease were in good
general health, a prediction model\nbecomes important.\n\nPrecision is the proportion of
every observation predicted to be positive that is actually positive. Therefore, I\nreco
mmend use of Logistic Regression classifer for predicting the heart disease.\n\n'
```