CAPSTONE PROJECT

Data Exploration and Modelling Documentation

Shreya Jain

Contents

Introduction

The World Health Organization estimates that heart disease causes 12 million deaths worldwide each year. Numerous studies have been carried out to identify the most important risk factors for heart disease and to precisely estimate the overall risk. Heart disease is also referred to as a silent killer because it causes a person to pass away without any evident signs. To avoid complications in high-risk patients and make decisions about lifestyle changes, early detection of heart disease is crucial. By examining patient data that uses machine learning algorithms to categorise whether a patient has heart disease or not, this study hopes to predict future cases of heart disease.

According to estimates, 17.9 million people die from heart disease each year, accounting for 32% of all fatalities worldwide. With the help of data, we can investigate and identify the factors that most influence a person's probability of developing heart disease.

Problem Statement

The main difficulty with heart disease is detecting it. There are tools that can forecast heart disease, but they are either expensive or ineffective at calculating the likelihood of heart disease in humans. The mortality rate and total consequences can be reduced by early identification of heart disorders. Since it takes more intelligence, time, and knowledge, it is not always possible to accurately monitor patients every day, and a doctor cannot consult with a patient for a whole 24 hours. Since there is a lot of data available nowadays, we can use a variety of machine learning methods to search for hidden patterns. In medical data, the hidden patterns might be used for health diagnosis.

Motivation

For many different types of data science applications, machine learning approaches have been around, compared, and used for analysis. This research-based effort was primarily driven by the desire to investigate the methodologies for feature selection, data preparation, and processing used in machine learning training models. The difficulty we have today with first-hand models and libraries is data, which, in addition to being plentiful and our cooked models, has a bigger variation than the accuracy we observe during training, testing, and actual validation. Therefore, the goal of this study is to investigate the mechanisms underlying the models that will be used to train the gathered data.

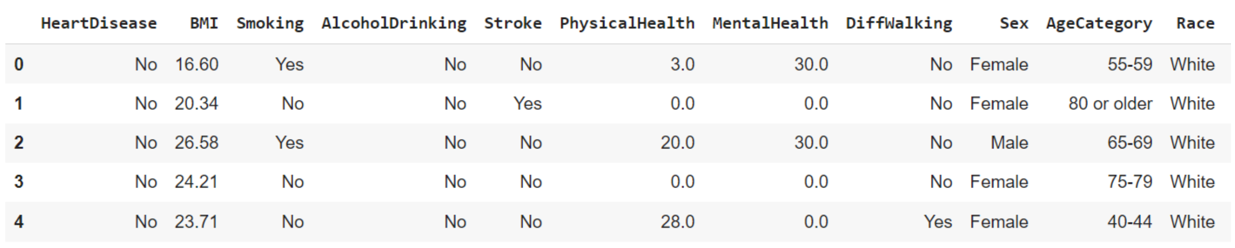
Objectives

The following are the key goals for creating this project:

1. To implement Logistic Regression in a machine learning model to predict the likelihood of heart disease in the future.
2. To identify significant risk factors for heart disease based on medical data.
3. To improve their chances of survival, people with heart disease or those who are at high risk for developing it need early detection and treatment.
4. To pinpoint the essential characteristics of cardiac disease so that patients can get a quick diagnosis and help.

Project Overview

Information on patients is included in the dataset, which has around 300000 records and 18 attributes. The following characteristics are listed among the attributes: HeartDisease, BMI, Smoking, AlcoholDrinking, Stroke, PhysicalHealth, MentalHealth, DiffWalking, Sex, AgeCategory, Race, Diabetes, PhysicalActivity, GenHealth, SleepTime, Asthma, KidneyDisease, SkinCancer. The variable "HeartDisease" is the target variable which is binary ("Yes": The patient has a heart disease; "No": The patient does not have a heart disease), however the classes are not balanced. The data collection is created as a data frame using the Pandas package in Python. Pre-processing and experiments are then conducted using this dataset.



	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race
0	No	16.60	Yes	No	No	3.0	30.0	No	Female	55-59	White
1	No	20.34	No	No	Yes	0.0	0.0	No	Female	80 or older	White
2	No	26.58	Yes	No	No	20.0	30.0	No	Male	65-69	White
3	No	24.21	No	No	No	0.0	0.0	No	Female	75-79	White
4	No	23.71	No	No	No	28.0	0.0	Yes	Female	40-44	White

The features

- HeartDisease ➔ Binary (Yes or No)
- BMI (Body mass Index) ➔ (values)
- Smoking ➔ Have you smoked at least 100 cigarettes in your entire life? (The answer is binary (Yes or No))
- AlcoholDrinking ➔ Heavy drinkers: adult men having more than 14 drinks per week and adult women having more than 7 drinks per week (The answer is binary (Yes or No))
- Stroke ➔ (Ever told) (you had) a stroke? (The answer is binary (Yes or No))

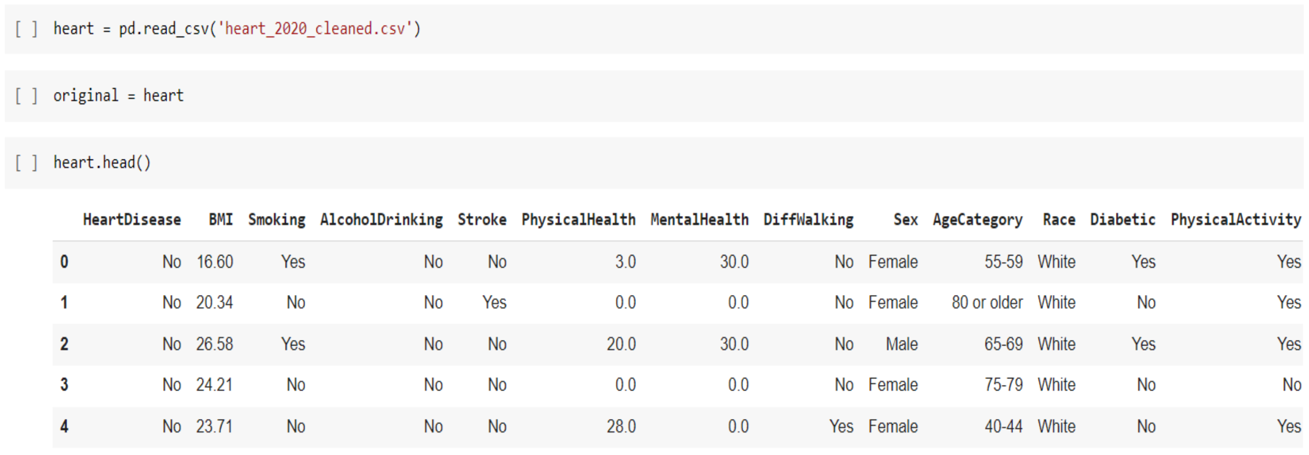
- PhysicalHealth ➔ Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days was your physical health not good? (0-30 days).
- MentalHealth ➔ Thinking about your mental health, for how many days during the past 30 days was your mental health not good? (0-30 days).
- DiffWalking ➔ Do you have serious difficulty walking or climbing stairs? (The answer is binary (Yes or No))
- Sex ➔ Are you male or female? (The response is binary (Female or Male))
- AgeCategory ➔ Fourteen-level age category.
- Race ➔ Imputed race/ethnicity value.
- Diabetic ➔ (Ever told) (you had) diabetes?
- PhysicalActivity ➔ Adults who reported doing physical activity or exercise during the past 30 days other than their regular job.
- GenHealth ➔ Would you say that in general your health is...
- SleepTime ➔ On average, how many hours of sleep do you get in a 24-hour period?
- Asthma ➔ (Ever told) (you had) asthma?
- KidneyDisease ➔ Not including kidney stones, bladder infection or incontinence, were you ever told you had kidney disease?
- SkinCancer ➔ (Ever told) (you had) skin cancer?

Categorical Features: HeartDisease, Smoking, AlcoholDrinking, Stroke, DiffWalking, Sex, Race, Diabetic, PhysicalActivity, GenHealth, Asthma, KidneyDisease, SkinCancer

Continuous Features: BMI, PhysicalHealth, MentalHealth, AgeCategory, SleepTime

Workspace

Importing file using pandas



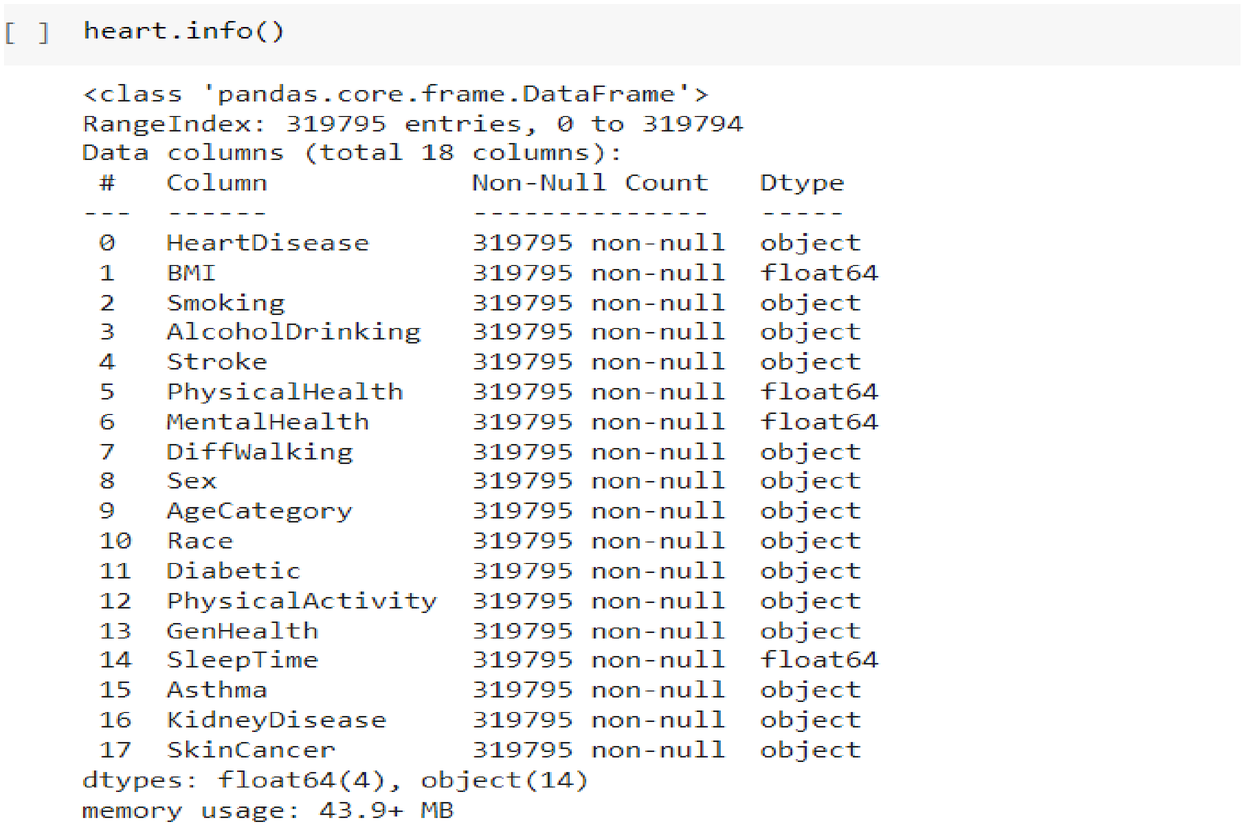
```
[ ] heart = pd.read_csv('heart_2020_cleaned.csv')
```

```
[ ] original = heart
```

```
[ ] heart.head()
```

	HeartDisease	BMI	Smoking	AlcoholDrinking	Stroke	PhysicalHealth	MentalHealth	DiffWalking	Sex	AgeCategory	Race	Diabetic	PhysicalActivity
0	No	16.60	Yes	No	No	3.0	30.0	No	Female	55-59	White	Yes	Yes
1	No	20.34	No	No	Yes	0.0	0.0	No	Female	80 or older	White	No	Yes
2	No	26.58	Yes	No	No	20.0	30.0	No	Male	65-69	White	Yes	Yes
3	No	24.21	No	No	No	0.0	0.0	No	Female	75-79	White	No	No
4	No	23.71	No	No	No	28.0	0.0	Yes	Female	40-44	White	No	Yes

Dataset information

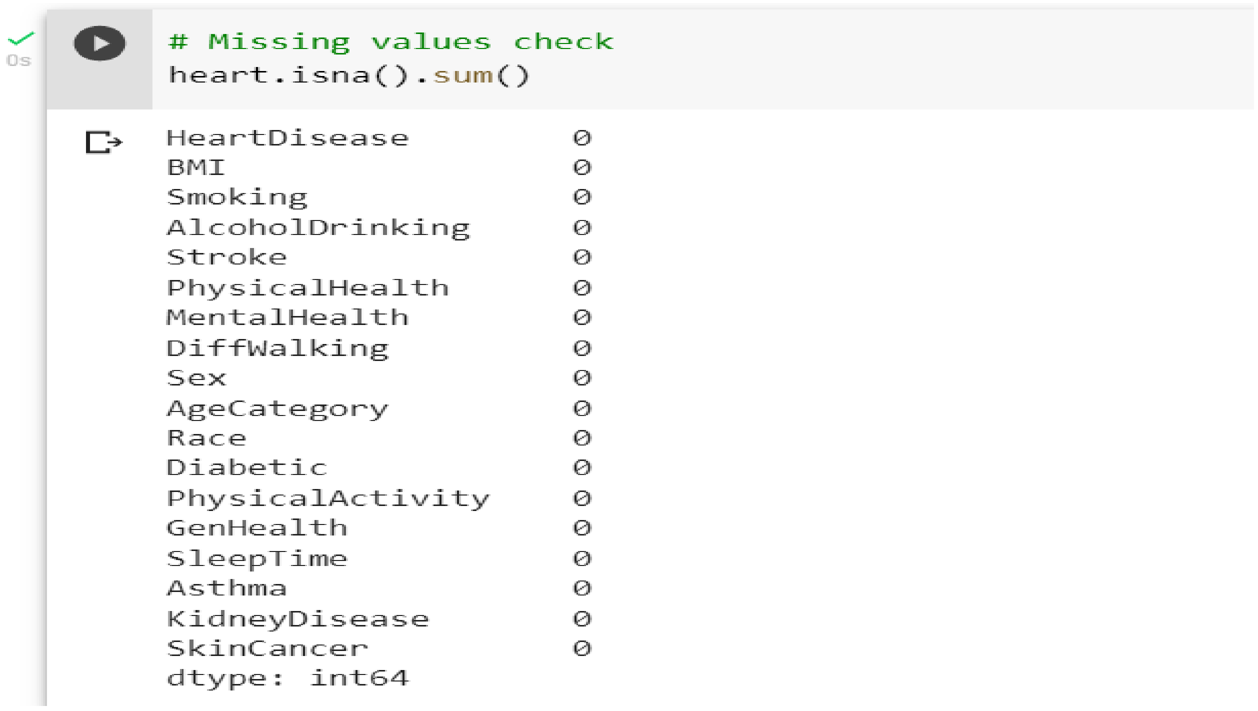


```
[ ]  heart.info()

     <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
```

Exploratory Data Analysis

Checking missing values

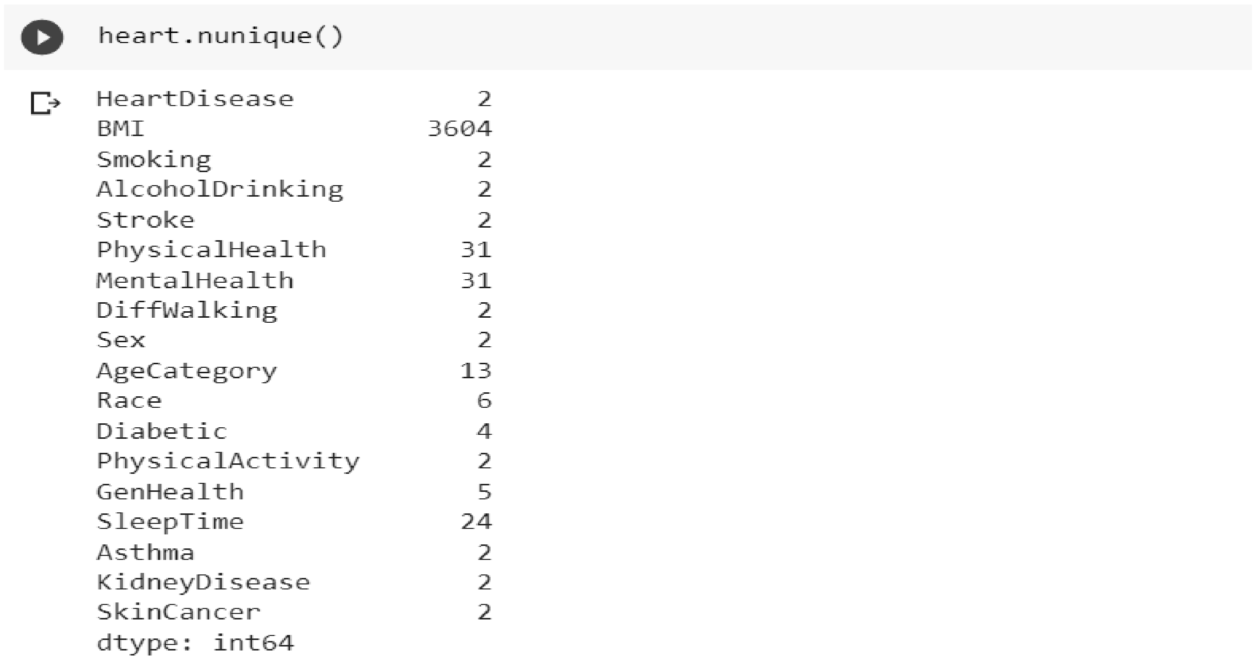
To properly handle the remaining data, it is crucial to determine whether any values are missing from the data. Even if some values are randomly absent, the data sample nevertheless probably represents the population. But the analysis might be skewed if the values are absent systematically.



```
# Missing values check
heart.isna().sum()
```

```
HeartDisease        0
BMI                 0
Smoking             0
AlcoholDrinking     0
Stroke              0
PhysicalHealth      0
MentalHealth        0
DiffWalking         0
Sex                 0
AgeCategory         0
Race                0
Diabetic            0
PhysicalActivity    0
GenHealth           0
SleepTime           0
Asthma              0
KidneyDisease       0
SkinCancer          0
dtype: int64
```

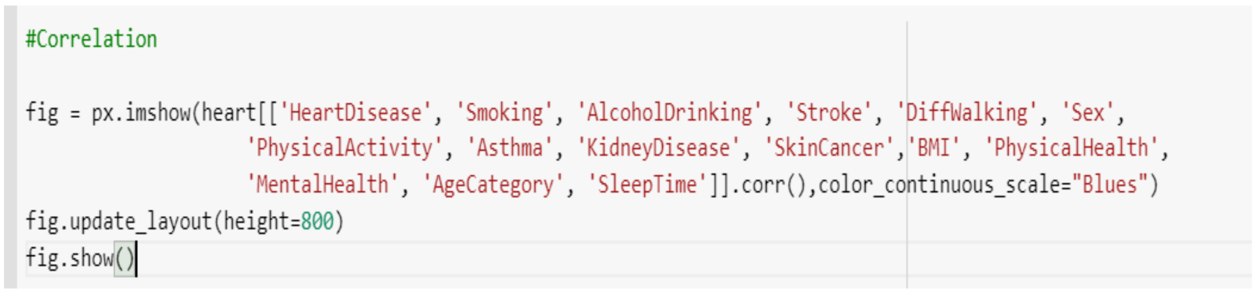
Checking unique values



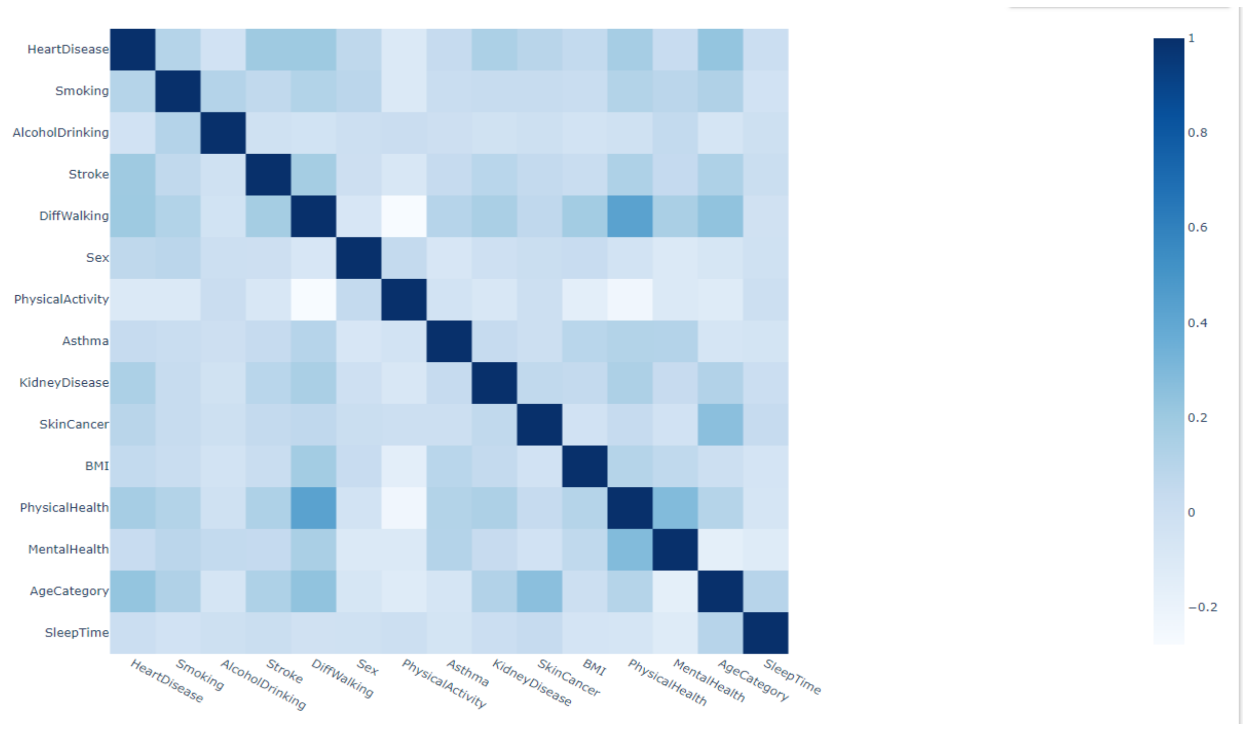
```
heart.nunique()
```

```
HeartDisease           2
BMI                 3604
Smoking                2
AlcoholDrinking        2
Stroke                 2
PhysicalHealth        31
MentalHealth          31
DiffWalking            2
Sex                    2
AgeCategory           13
Race                   6
Diabetic               4
PhysicalActivity       2
GenHealth              5
SleepTime             24
Asthma                 2
KidneyDisease          2
SkinCancer             2
dtype: int64
```

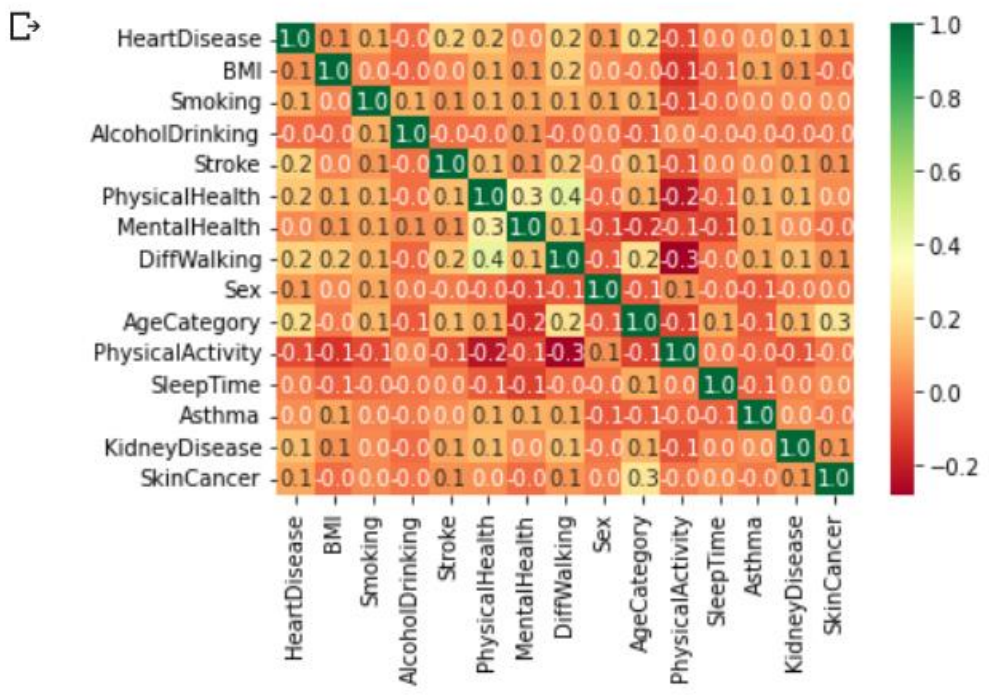
Correlation Analysis



```
#Correlation

fig = px.imshow(heart[['HeartDisease', 'Smoking', 'AlcoholDrinking', 'Stroke', 'DiffWalking', 'Sex',
                 'PhysicalActivity', 'Asthma', 'KidneyDisease', 'SkinCancer','BMI', 'PhysicalHealth',
                 'MentalHealth', 'AgeCategory', 'SleepTime']].corr(),color_continuous_scale="Blues")
fig.update_layout(height=800)
fig.show()
```

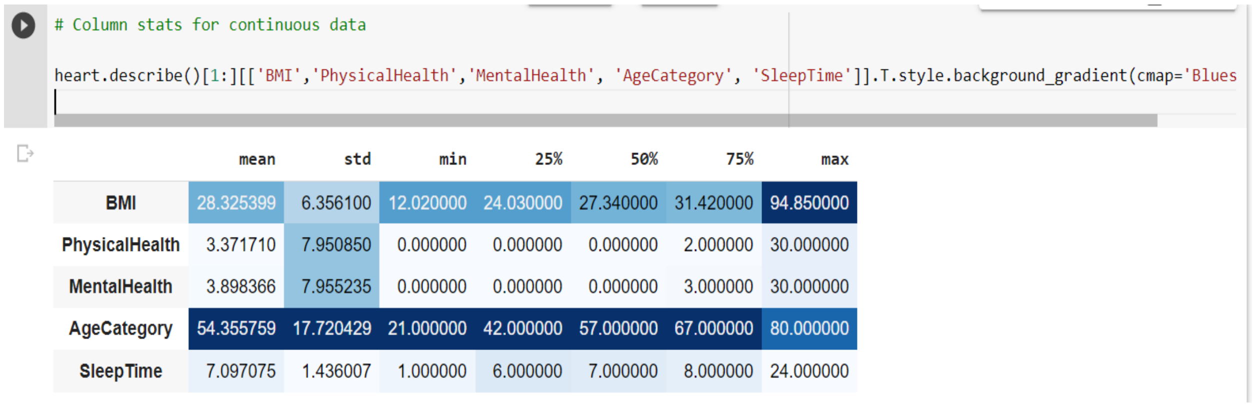


Correlation Heatmap



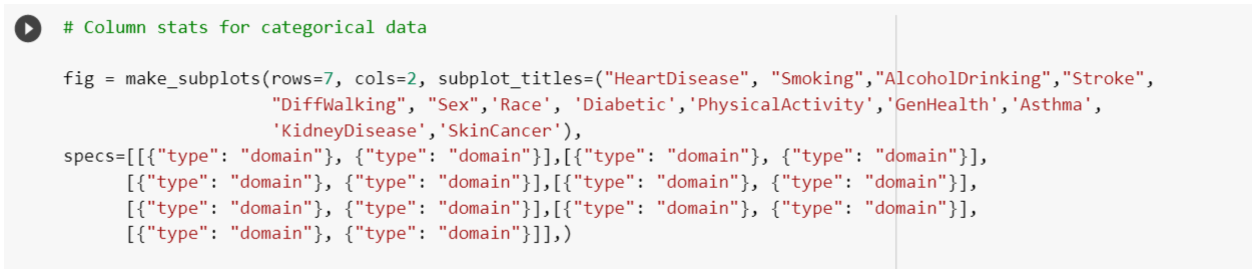
No strong correlation has been found among the variables.

Data Distribution

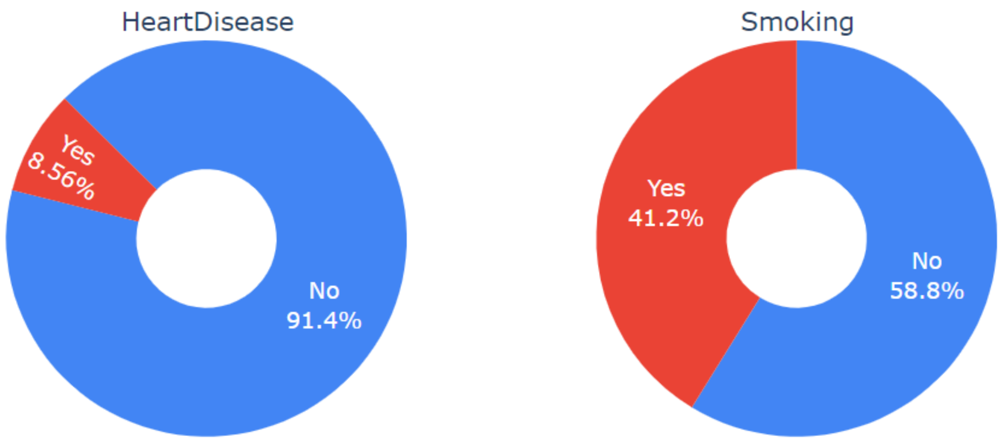


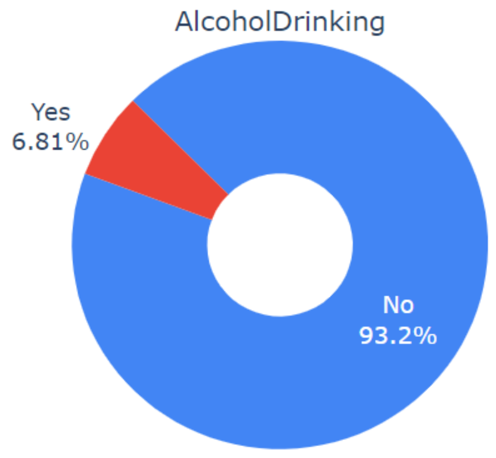
```
# Column stats for continuous data

heart.describe()[1:][['BMI','PhysicalHealth','MentalHealth', 'AgeCategory', 'SleepTime']].T.style.background_gradient(cmap='Blues
```

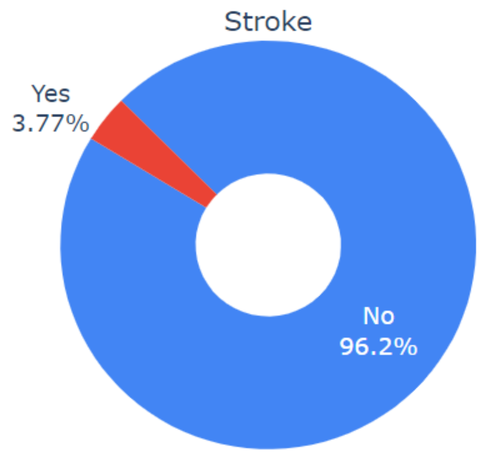
	mean	std	min	25%	50%	75%	max
BMI	28.325399	6.356100	12.020000	24.030000	27.340000	31.420000	94.850000
PhysicalHealth	3.371710	7.950850	0.000000	0.000000	0.000000	2.000000	30.000000
MentalHealth	3.898366	7.955235	0.000000	0.000000	0.000000	3.000000	30.000000
AgeCategory	54.355759	17.720429	21.000000	42.000000	57.000000	67.000000	80.000000
SleepTime	7.097075	1.436007	1.000000	6.000000	7.000000	8.000000	24.000000

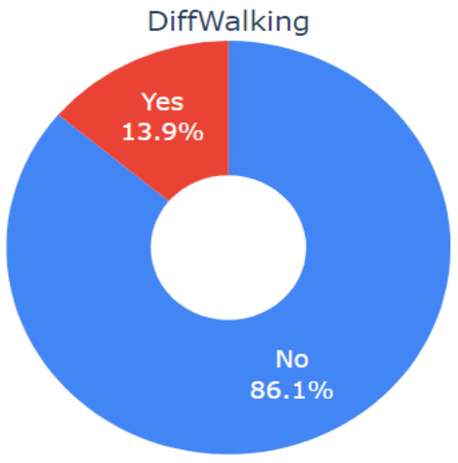


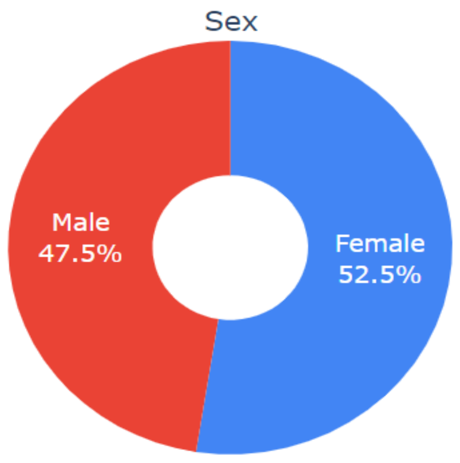
```
# Column stats for categorical data

fig = make_subplots(rows=7, cols=2, subplot_titles=("HeartDisease", "Smoking","AlcoholDrinking","Stroke",
                    "DiffWalking", "Sex",'Race', 'Diabetic','PhysicalActivity','GenHealth','Asthma',
                    'KidneyDisease','SkinCancer'),
specs=[[{"type": "domain"}, {"type": "domain"}],[{"type": "domain"}, {"type": "domain"}],
      [{"type": "domain"}, {"type": "domain"}],[{"type": "domain"}, {"type": "domain"}],
      [{"type": "domain"}, {"type": "domain"}],[{"type": "domain"}, {"type": "domain"}],
      [{"type": "domain"}, {"type": "domain"}]],)
```

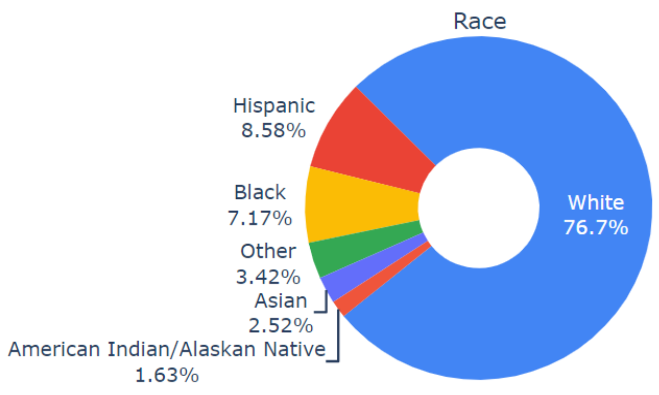


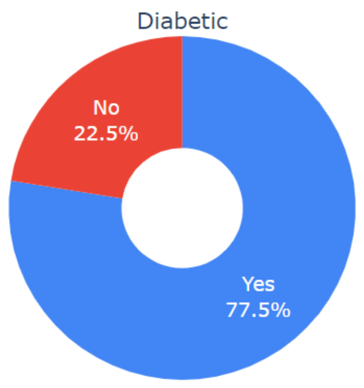

AlcoholDrinking
Yes
6.81%
No
93.2%

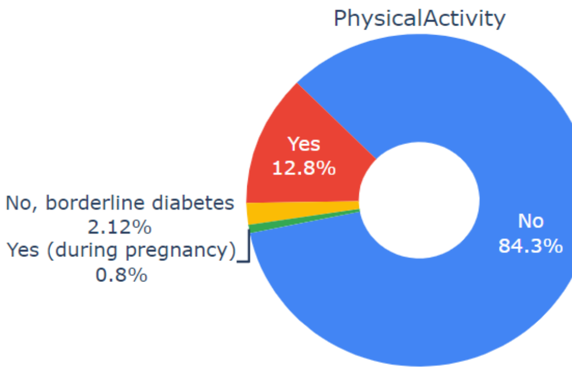

Stroke
Yes
3.77%
No
96.2%

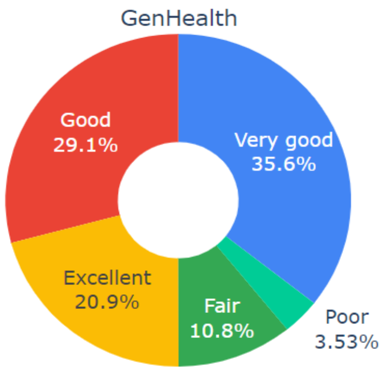

DiffWalking
Yes
13.9%
No
86.1%

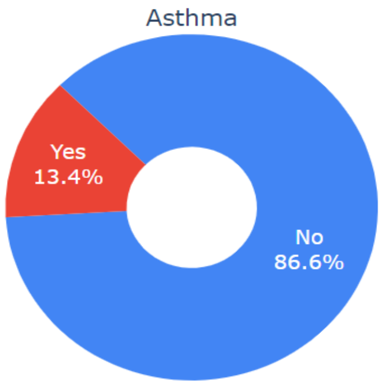

Sex
Male
47.5%
Female
52.5%

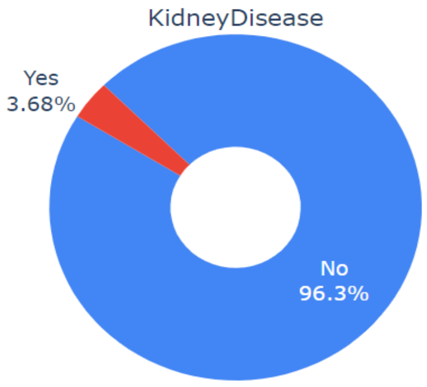

Race
Hispanic
8.58%
Black
7.17%
Other
3.42%
Asian
2.52%
American Indian/Alaskan Native
1.63%
White
76.7%

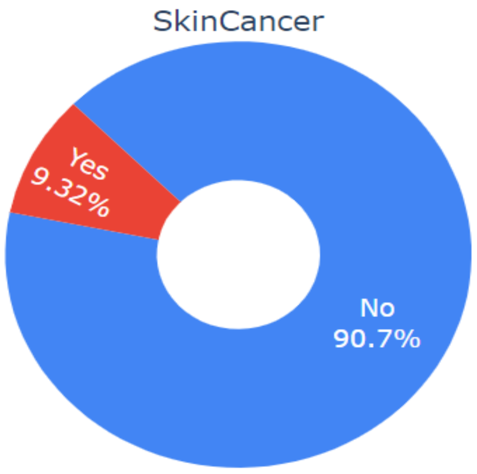

Diabetic
No
22.5%
Yes
77.5%


PhysicalActivity
Yes
12.8%
No, borderline diabetes
2.12%
Yes (during pregnancy)
0.8%
No
84.3%


GenHealth
Good
29.1%
Very good
35.6%
Excellent
20.9%
Fair
10.8%
Poor
3.53%

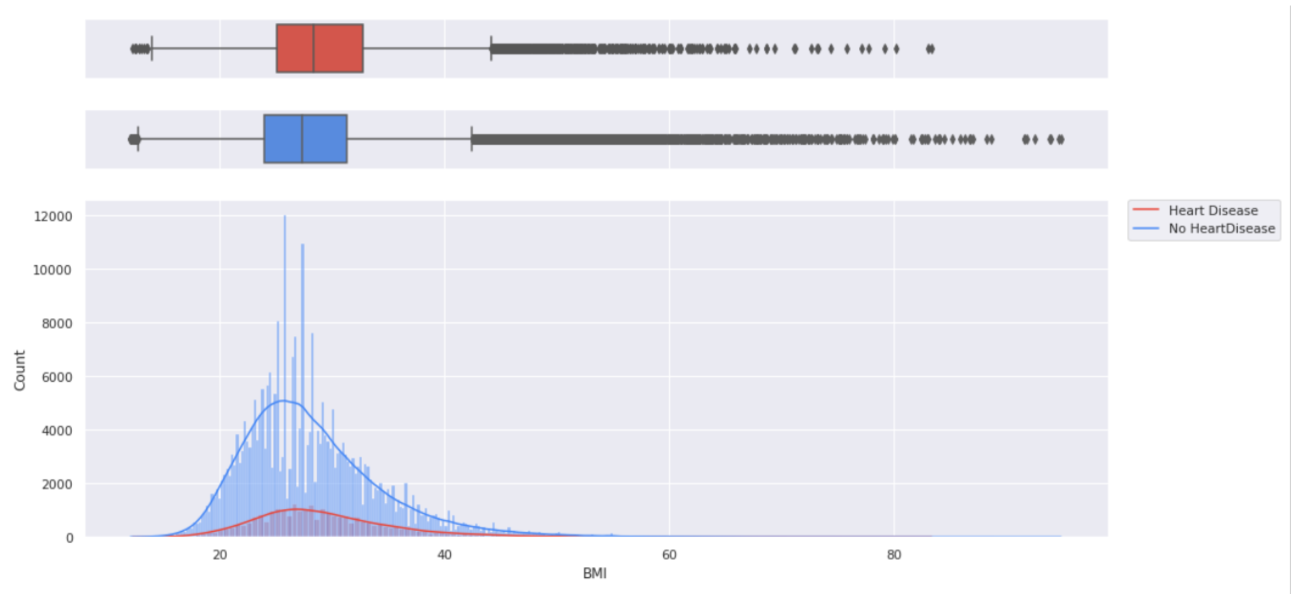

Asthma
Yes
13.4%
No
86.6%

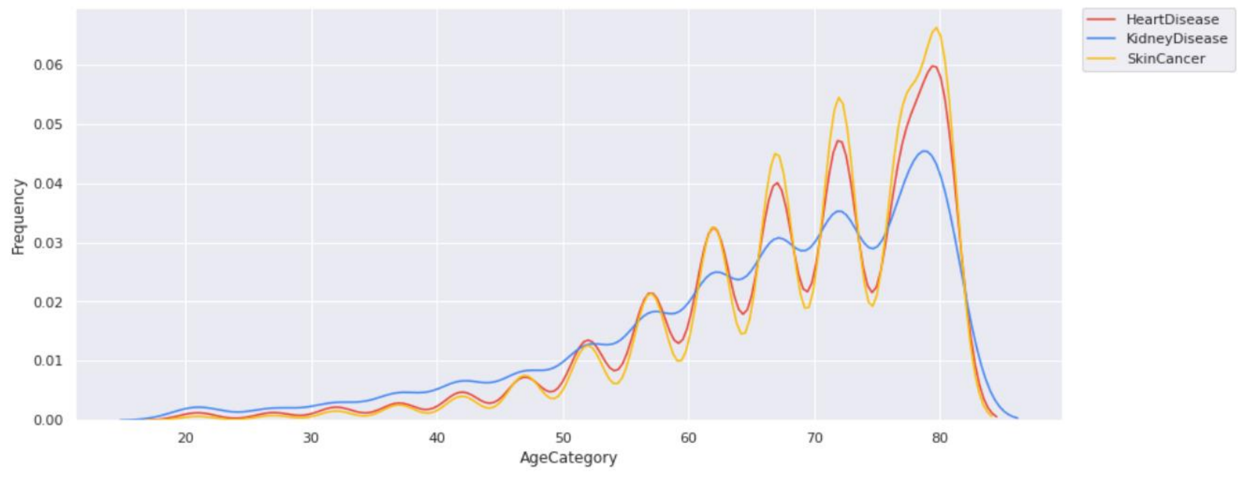

KidneyDisease
Yes
3.68%
No
96.3%


SkinCancer
Yes
9.32%
No
90.7%

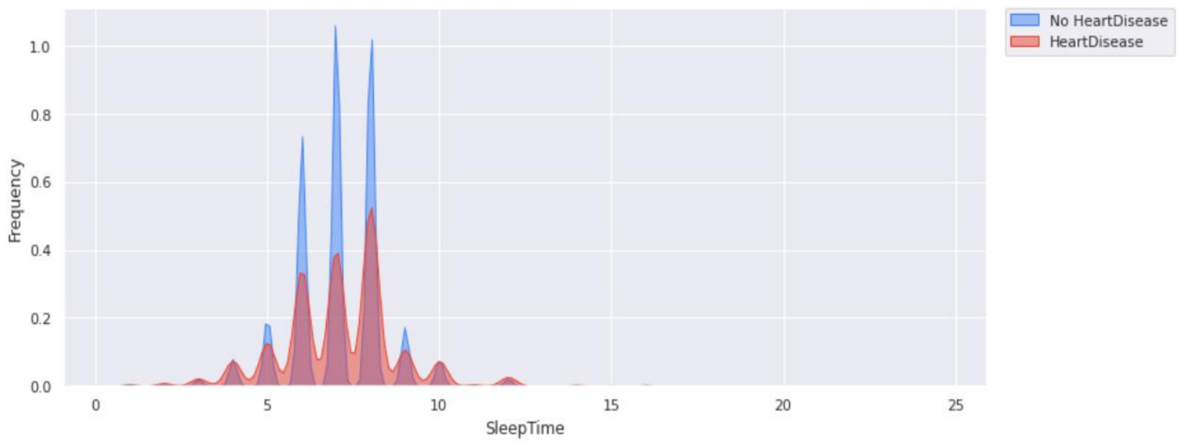
The target variable (HeartDisease) has an uneven distribution, as seen in the plots above, so the data needs to be balanced before developing the model.

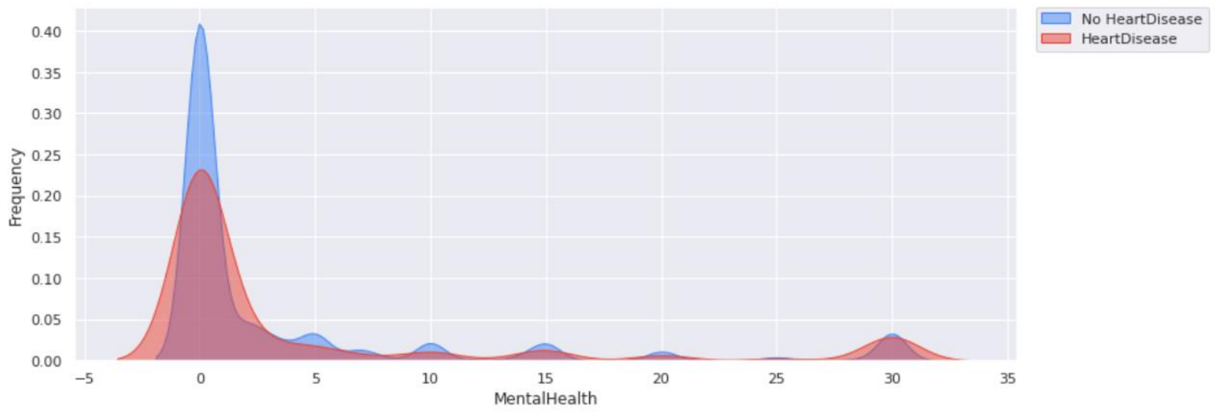
Data Visualization

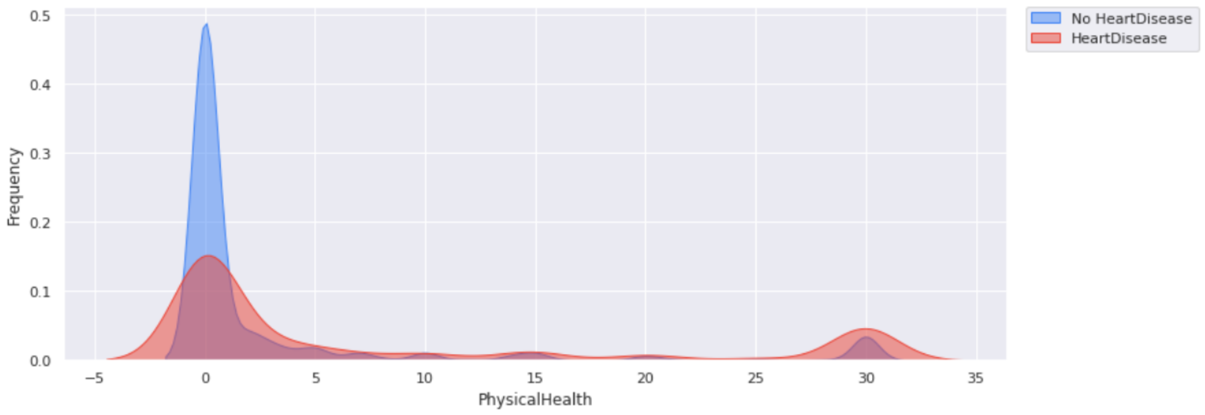


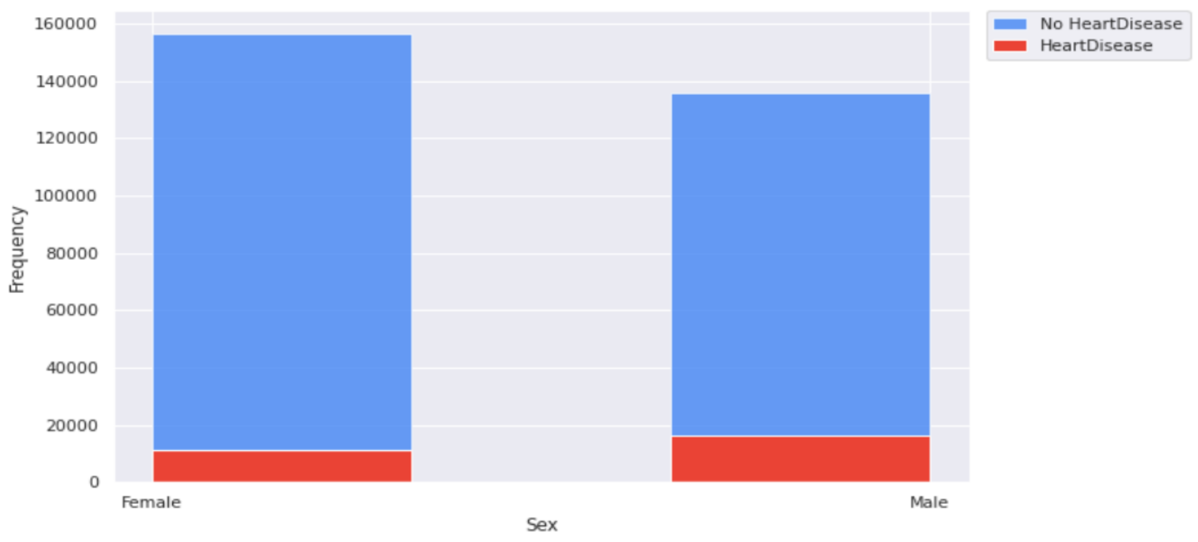


Insights - People having heart disease, skin cancer and kidney disease are mostly old people


No HeartDisease
HeartDisease
Frequency
SleepTime


No HeartDisease
HeartDisease
Frequency
MentalHealth


No HeartDisease
HeartDisease
Frequency
PhysicalHealth

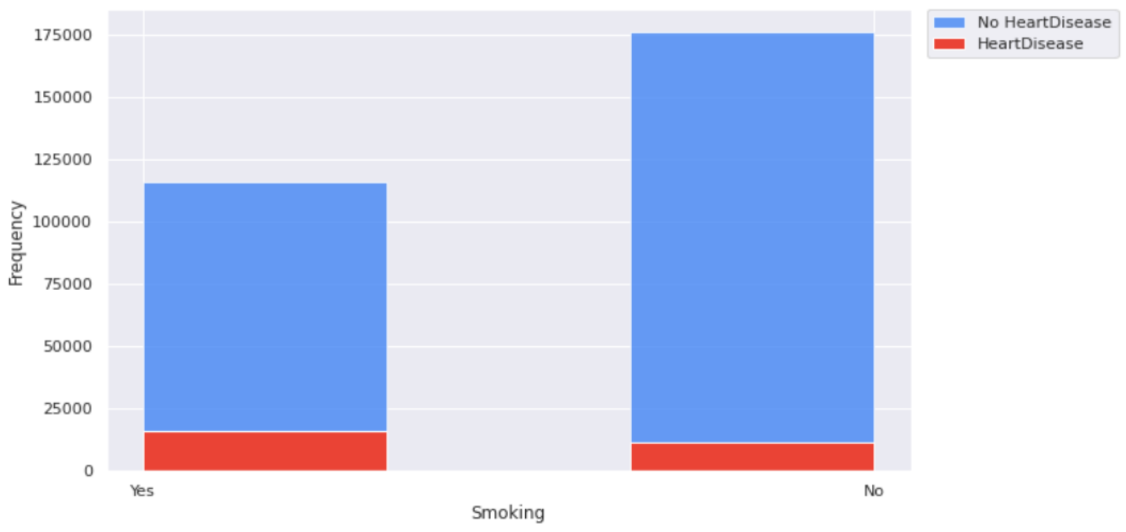


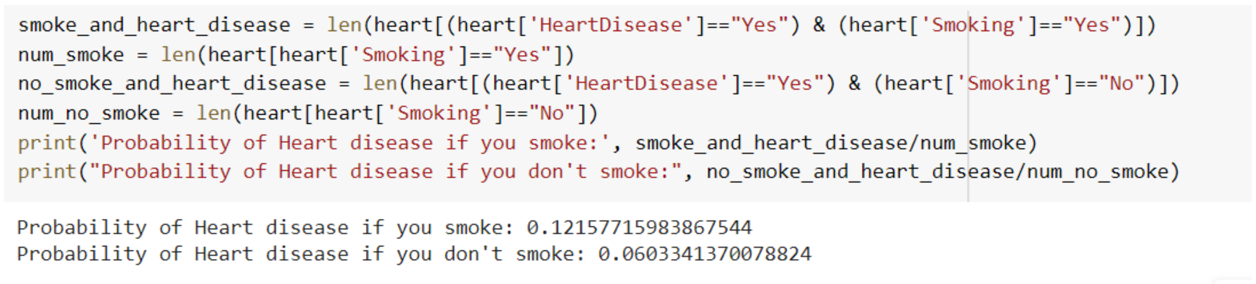
```
female_with_heart_disease = len(heart[(heart['HeartDisease']=="Yes") & (heart['Sex']=="Female")])
num_female = len(heart[heart['Sex']=="Female"])
male_with_heart_disease = len(heart[(heart['HeartDisease']=="Yes") & (heart['Sex']=="Male")])
num_male = len(heart[heart['Sex']=="Male"])
print('Probability of Male to have Heart disease:', male_with_heart_disease/num_male)
print('Probability of Female to have Heart disease:', female_with_heart_disease/num_female)
```

```
Probability of Male to have Heart disease: 0.10618461740904007
Probability of Female to have Heart disease: 0.06694675367241738
```

Insights -

1. Mostly heart disease patients are Males than Females
2. More Females were tested than Males
3. Males are approximately 1.6 times more likely to have heart disease than females



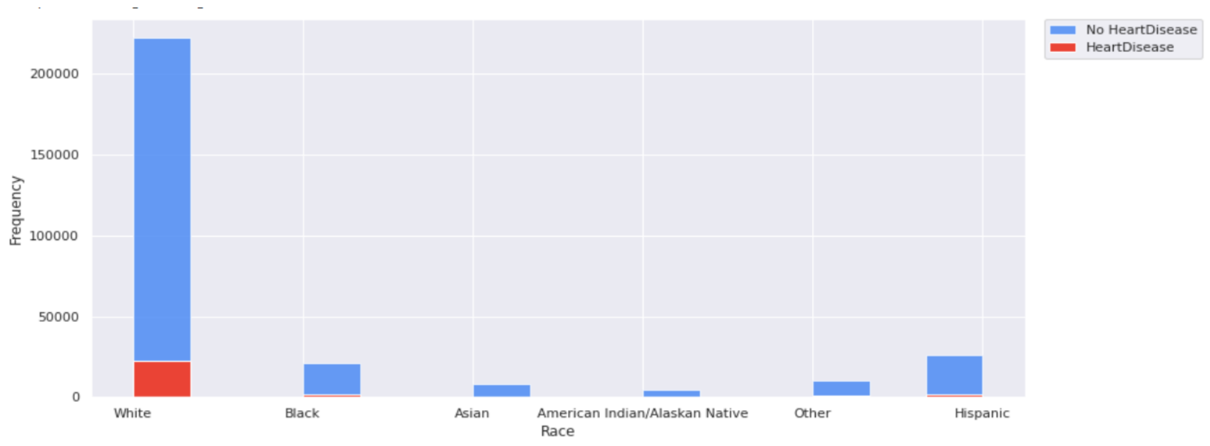


```
smoke_and_heart_disease = len(heart[(heart['HeartDisease']=="Yes") & (heart['Smoking']=="Yes")])
num_smoke = len(heart[heart['Smoking']=="Yes"])
no_smoke_and_heart_disease = len(heart[(heart['HeartDisease']=="Yes") & (heart['Smoking']=="No")])
num_no_smoke = len(heart[heart['Smoking']=="No"])
print('Probability of Heart disease if you smoke:', smoke_and_heart_disease/num_smoke)
print("Probability of Heart disease if you don't smoke:", no_smoke_and_heart_disease/num_no_smoke)
```

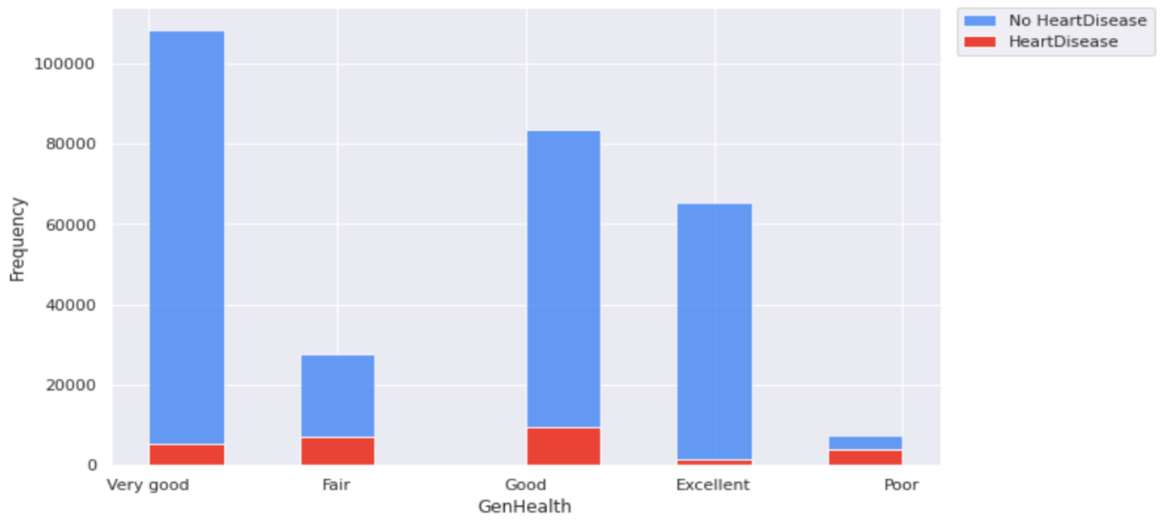
```
Probability of Heart disease if you smoke: 0.12157715983867544
Probability of Heart disease if you don't smoke: 0.0603341370078824
```

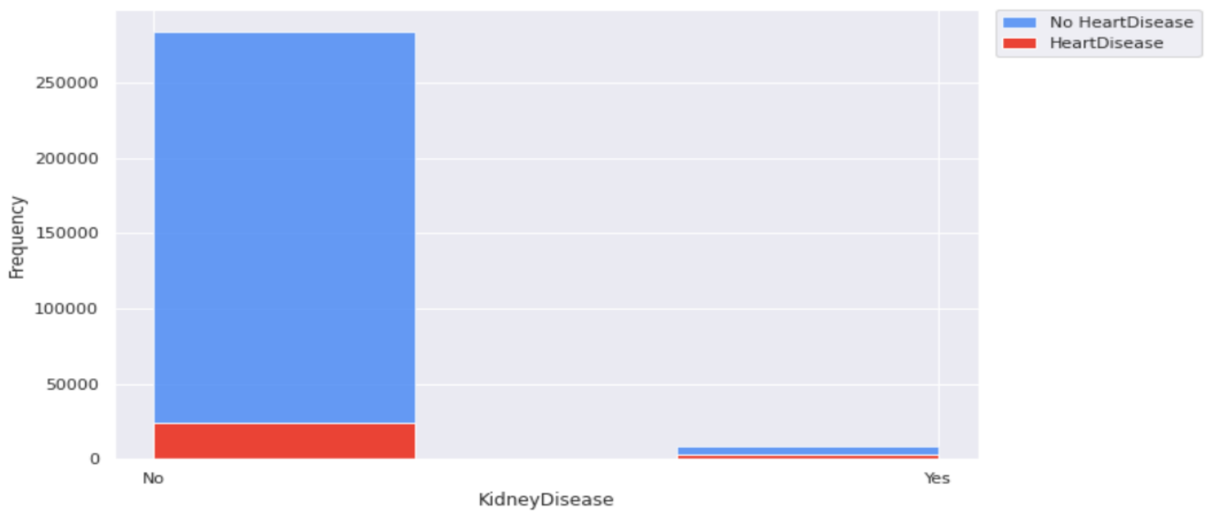
Insights -

1. Most heart disease patients smoke.
2. People who smoke are approximately twice as likely to have heart disease than people who don't smoke.

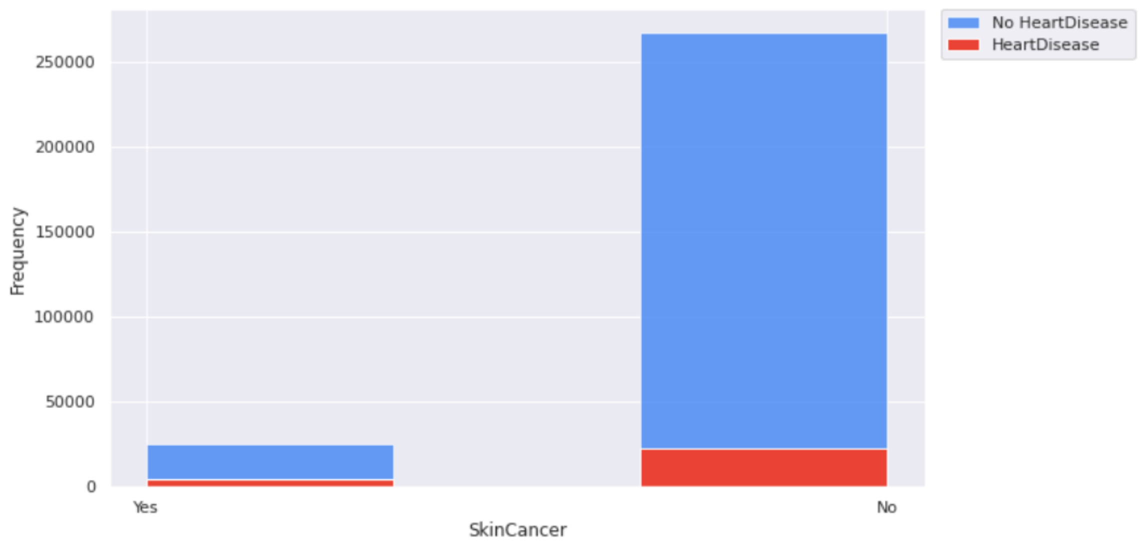


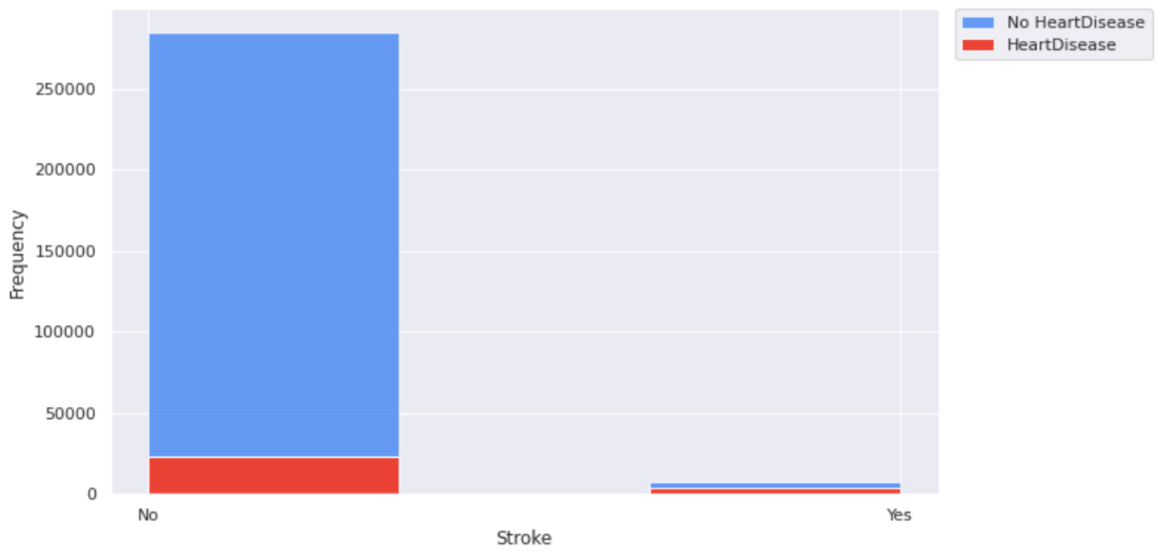
Insights - Most heart disease patients are white people.

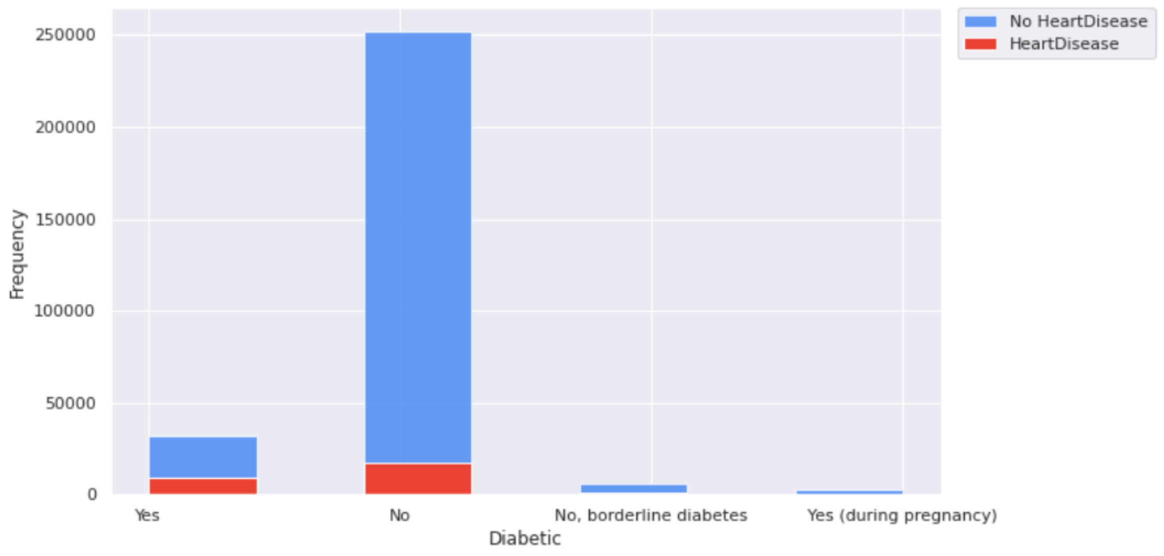


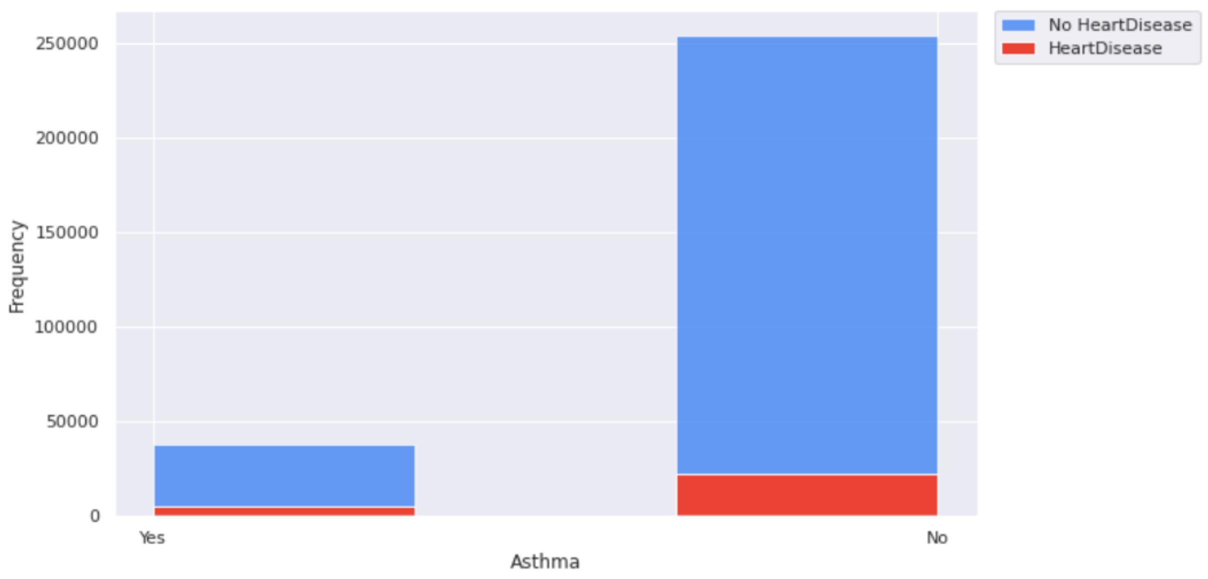


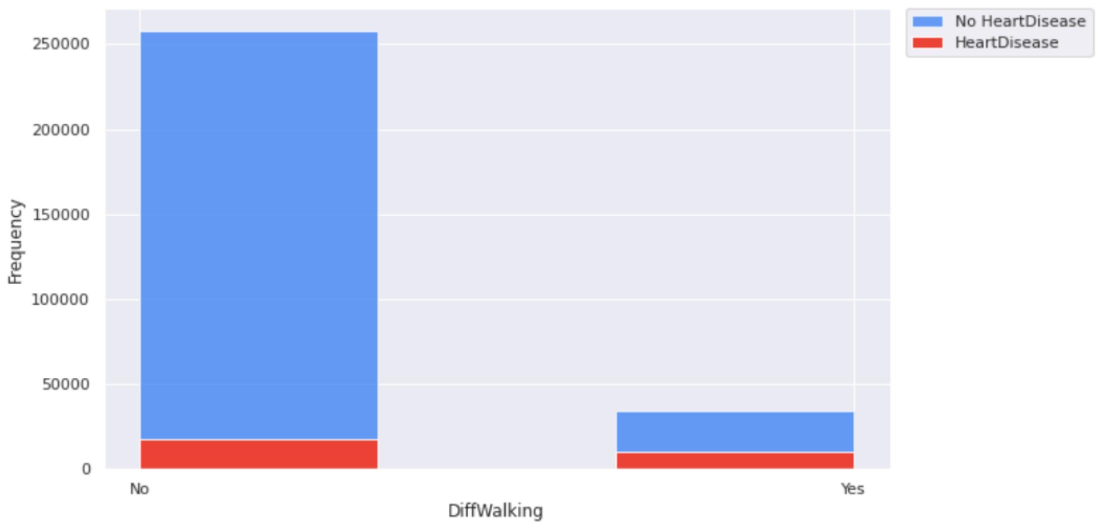
Insights - Most of the people tested have no kidney disease.

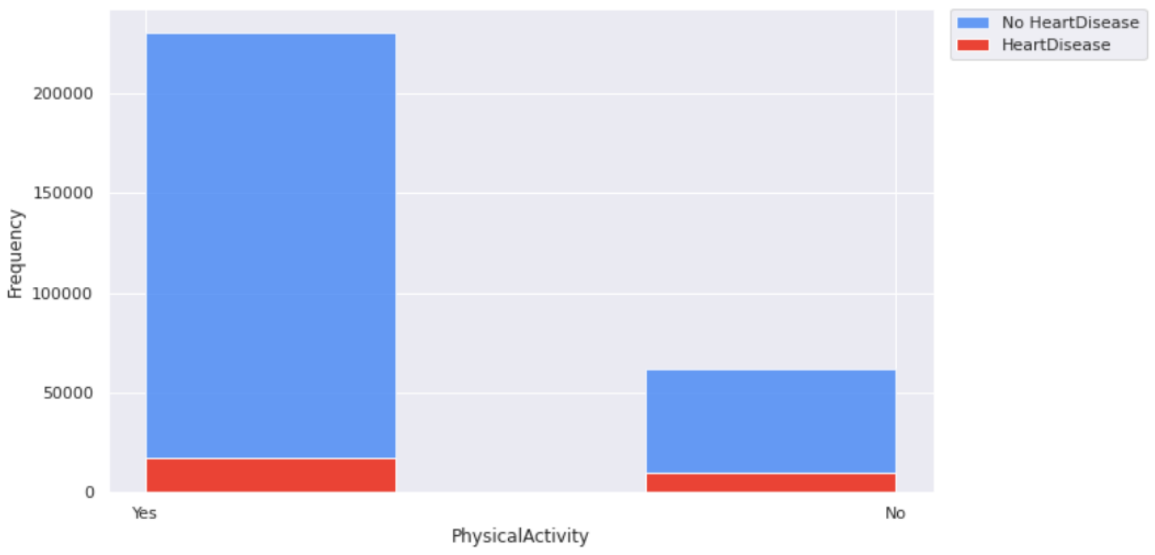


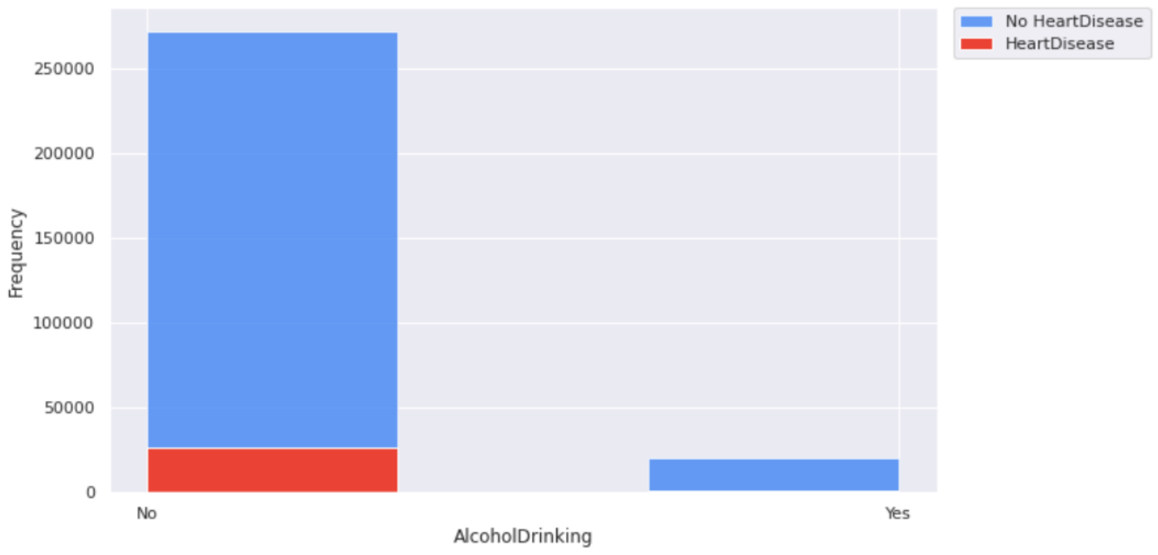

No HeartDisease
HeartDisease
250000
200000
150000
100000
50000
0
Frequency
No
Yes
Stroke


No HeartDisease
HeartDisease
250000
200000
150000
100000
50000
0
Frequency
Yes
No
No, borderline diabetes
Yes (during pregnancy)
Diabetic

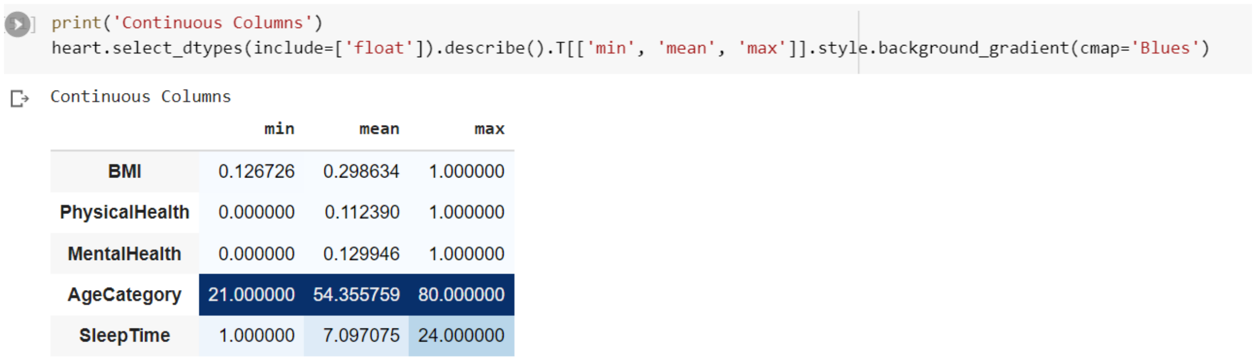

No HeartDisease
HeartDisease
Frequency
250000
200000
150000
100000
50000
0
Yes
No
Asthma


No HeartDisease
HeartDisease
Frequency
250000
200000
150000
100000
50000
0
No
Yes
DiffWalking


No HeartDisease
HeartDisease
200000
150000
100000
50000
0
Frequency
Yes
No
PhysicalActivity


No HeartDisease
HeartDisease
250000
200000
150000
100000
50000
0
Frequency
No
Yes
AlcoholDrinking

Data Cleaning and Preprocessing



```
print('Continuous Columns')
heart.select_dtypes(include=['float']).describe().T[['min', 'mean', 'max']].style.background_gradient(cmap='Blues')
```

```
Continuous Columns
```

	min	mean	max
BMI	0.126726	0.298634	1.000000
PhysicalHealth	0.000000	0.112390	1.000000
MentalHealth	0.000000	0.129946	1.000000
AgeCategory	21.000000	54.355759	80.000000
SleepTime	1.000000	7.097075	24.000000

The range of continuous features is different.

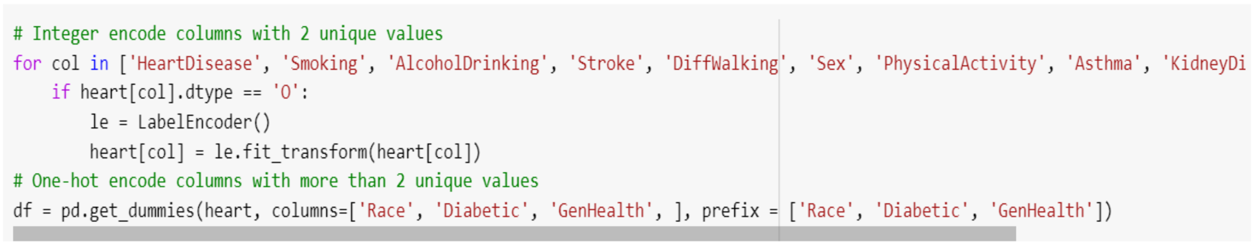
So, dividing them by the highest value of each column to scale them to be between 0 and 1.

```
for col in ['BMI', 'PhysicalHealth', 'MentalHealth', 'AgeCategory', 'SleepTime']:
  heart[col] = heart[col]/heart[col].max()
```

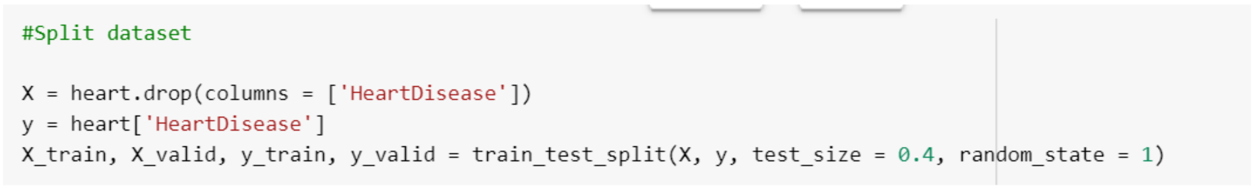
```
print('\nCategorical Columns\n')
heart.select_dtypes(include=['O']).nunique()
```

```
Categorical Columns

HeartDisease        2
Smoking             2
AlcoholDrinking     2
Stroke              2
DiffWalking         2
Sex                 2
Race                6
Diabetic            4
PhysicalActivity    2
GenHealth           5
Asthma              2
KidneyDisease       2
SkinCancer          2
dtype: int64
```

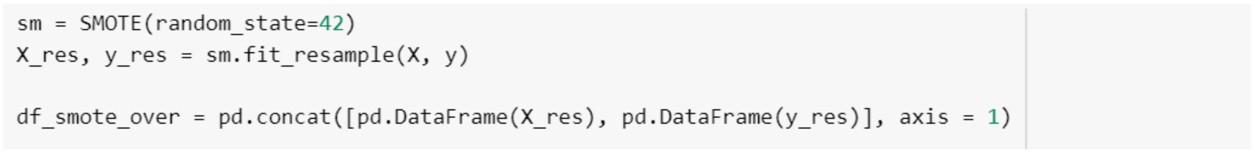
There are categories columns with only two values and others with multiple values. Here, the columns with two distinct values will be converted to binary (either 1 or 0), while the other categorical columns with more than two distinct values will be one-hot encoded.

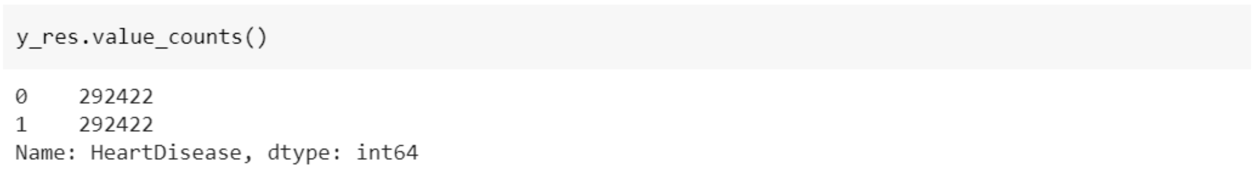

```
# Integer encode columns with 2 unique values
for col in ['HeartDisease', 'Smoking', 'AlcoholDrinking', 'Stroke', 'DiffWalking', 'Sex', 'PhysicalActivity', 'Asthma', 'KidneyDi
    if heart[col].dtype == 'O':
        le = LabelEncoder()
        heart[col] = le.fit_transform(heart[col])
# One-hot encode columns with more than 2 unique values
df = pd.get_dummies(heart, columns=['Race', 'Diabetic', 'GenHealth', ], prefix = ['Race', 'Diabetic', 'GenHealth'])
```

Splitting the dataset

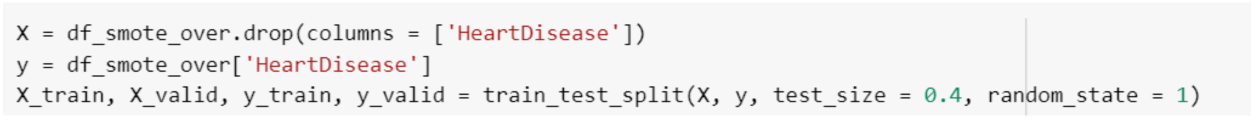

```
#Split dataset

X = heart.drop(columns = ['HeartDisease'])
y = heart['HeartDisease']
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size = 0.4, random_state = 1)
```

As mentioned earlier, the dataset is not balanced, there are a greater number of samples for people who have no heart disease than people with heart disease. So, this can be fixed by oversampling the data using SMOTE technique.


```
sm = SMOTE(random_state=42)
X_res, y_res = sm.fit_resample(X, y)

df_smote_over = pd.concat([pd.DataFrame(X_res), pd.DataFrame(y_res)], axis = 1)
```


```
y_res.value_counts()
```

```
0    292422
1    292422
Name: HeartDisease, dtype: int64
```


```
X = df_smote_over.drop(columns = ['HeartDisease'])
y = df_smote_over['HeartDisease']
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size = 0.4, random_state = 1)
```

One of the most popular oversampling techniques to address the imbalance issue is SMOTE (synthetic minority oversampling technique). By increasing the minority class samples at random and duplicating them, it seeks to balance the distribution of classes.

So, now the data is balanced, and modelling can be performed.

Model building

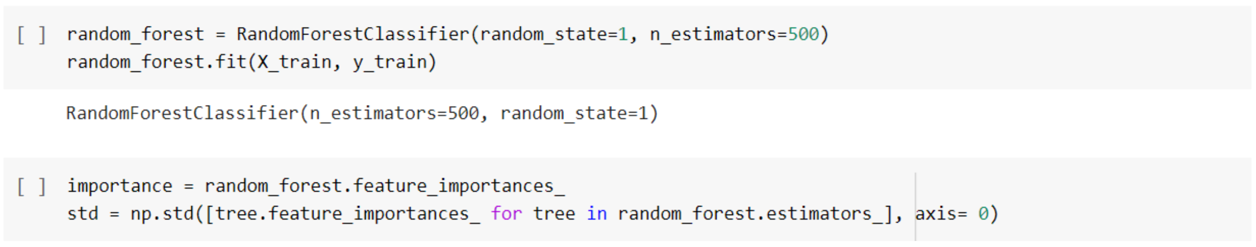
Random Forest (variable/feature importance)

A classification system made up of several decision trees is called the random forest. It attempts to produce an uncorrelated forest of trees whose forecast by committee is more accurate than that of any individual tree by using bagging and feature randomness when generating each individual tree.

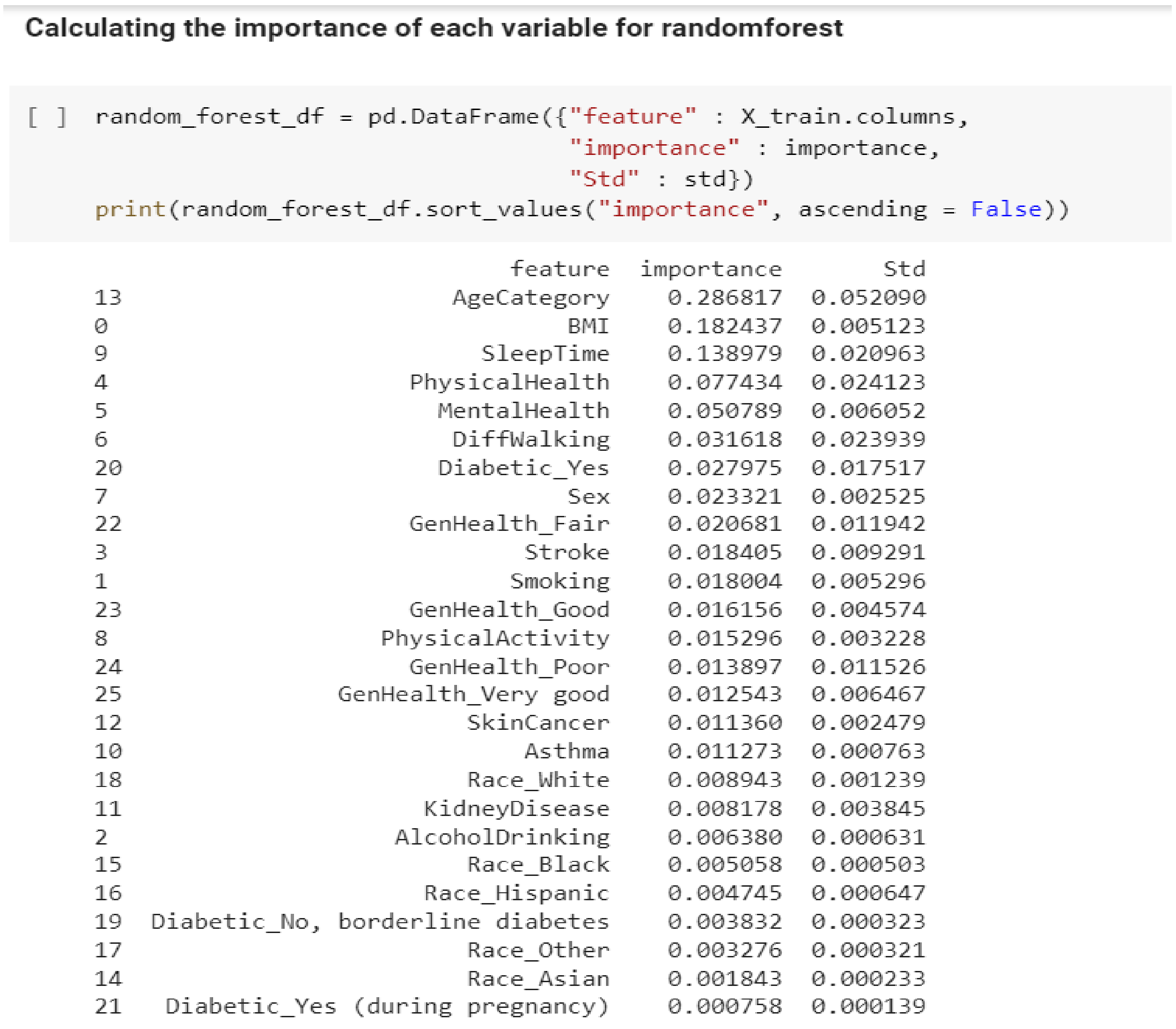
Knowing the feature importance suggested by machine learning models can be useful in a variety of ways, such as:

- Understanding the model's logic will enable you to not only confirm that it is accurate but also to work on improving the model by concentrating just on the crucial variables.
- It can be used for variable selection, allowing you to eliminate x variables that have little bearing on performance and require significantly less training time.
- In certain commercial situations, it makes sense to give up some accuracy in favour of interpretability. For instance, if a bank denies a loan application, it must provide the customer with the justification for the decision.

Random forest



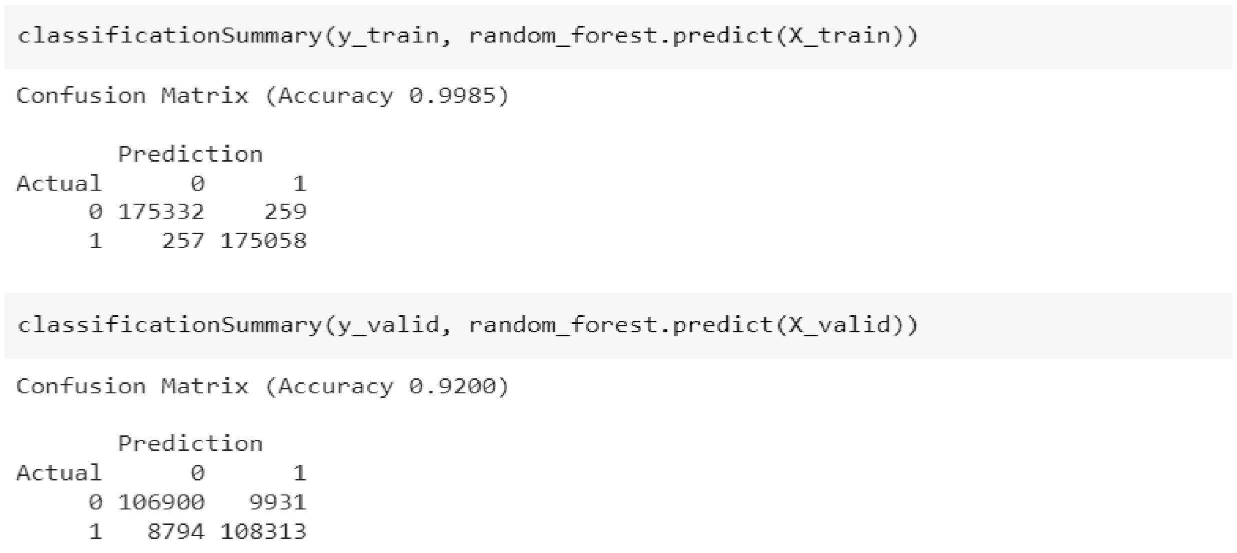
```
[ ]  random_forest = RandomForestClassifier(random_state=1, n_estimators=500)
     random_forest.fit(X_train, y_train)

     RandomForestClassifier(n_estimators=500, random_state=1)

[ ]  importance = random_forest.feature_importances_
     std = np.std([tree.feature_importances_ for tree in random_forest.estimators_], axis= 0)
```



Calculating the importance of each variable for randomforest

```
[ ]  random_forest_df = pd.DataFrame({"feature" : X_train.columns,
                                      "importance" : importance,
                                      "Std" : std})
     print(random_forest_df.sort_values("importance", ascending = False))
```

```
                          feature  importance       Std
13                    AgeCategory    0.286817  0.052090
0                             BMI    0.182437  0.005123
9                       SleepTime    0.138979  0.020963
4                  PhysicalHealth    0.077434  0.024123
5                    MentalHealth    0.050789  0.006052
6                     DiffWalking    0.031618  0.023939
20                   Diabetic_Yes    0.027975  0.017517
7                             Sex    0.023321  0.002525
22                 GenHealth_Fair    0.020681  0.011942
3                          Stroke    0.018405  0.009291
1                         Smoking    0.018004  0.005296
23                 GenHealth_Good    0.016156  0.004574
8                PhysicalActivity    0.015296  0.003228
24                 GenHealth_Poor    0.013897  0.011526
25            GenHealth_Very good    0.012543  0.006467
12                     SkinCancer    0.011360  0.002479
10                         Asthma    0.011273  0.000763
18                     Race_White    0.008943  0.001239
11                  KidneyDisease    0.008178  0.003845
2                  AlcoholDrinking    0.006380  0.000631
15                     Race_Black    0.005058  0.000503
16                  Race_Hispanic    0.004745  0.000647
19  Diabetic_No, borderline diabetes    0.003832  0.000323
17                     Race_Other    0.003276  0.000321
14                     Race_Asian    0.001843  0.000233
21   Diabetic_Yes (during pregnancy)    0.000758  0.000139
```



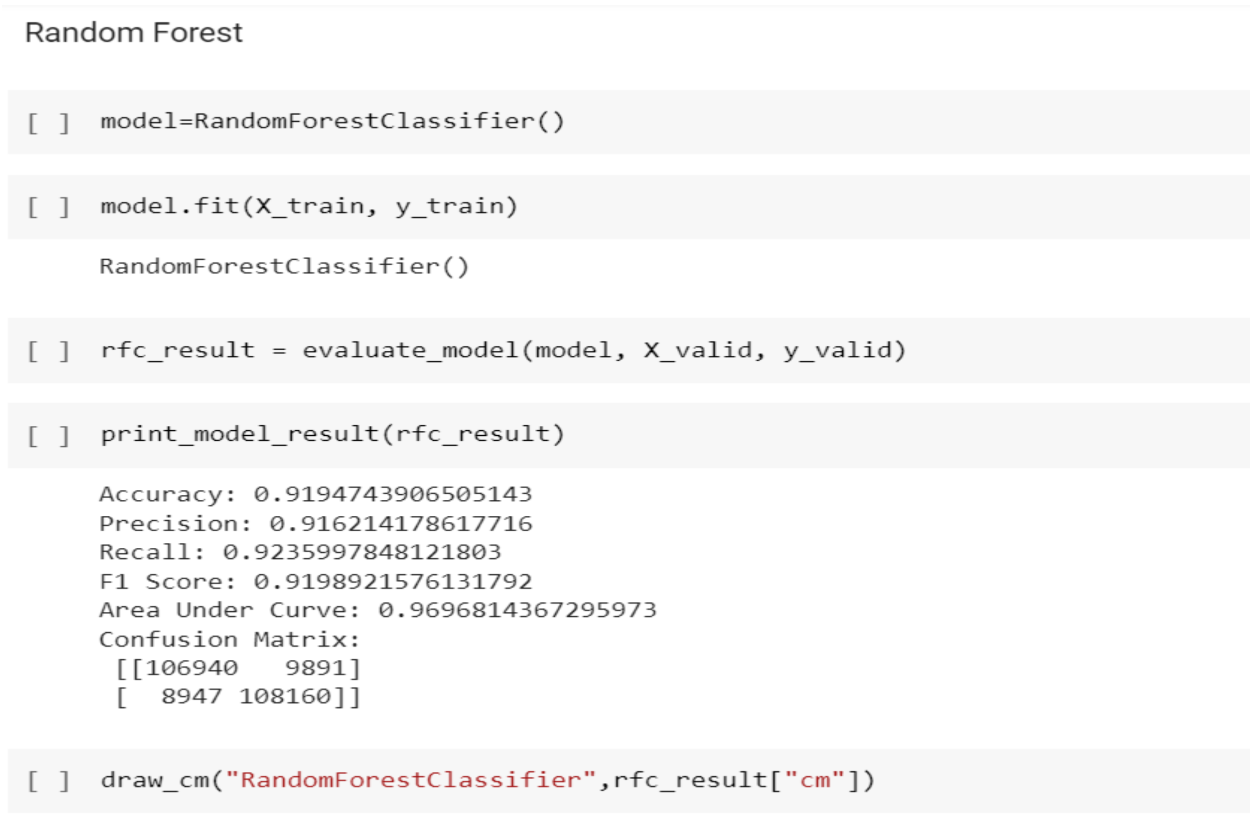
```
classificationSummary(y_train, random_forest.predict(X_train))
```

```
Confusion Matrix (Accuracy 0.9985)

       Prediction
Actual      0      1
     0 175332    259
     1    257 175058
```

```
classificationSummary(y_valid, random_forest.predict(X_valid))
```

```
Confusion Matrix (Accuracy 0.9200)

       Prediction
Actual      0      1
     0 106900   9931
     1   8794 108313
```

Random Forest Classifier



Random Forest

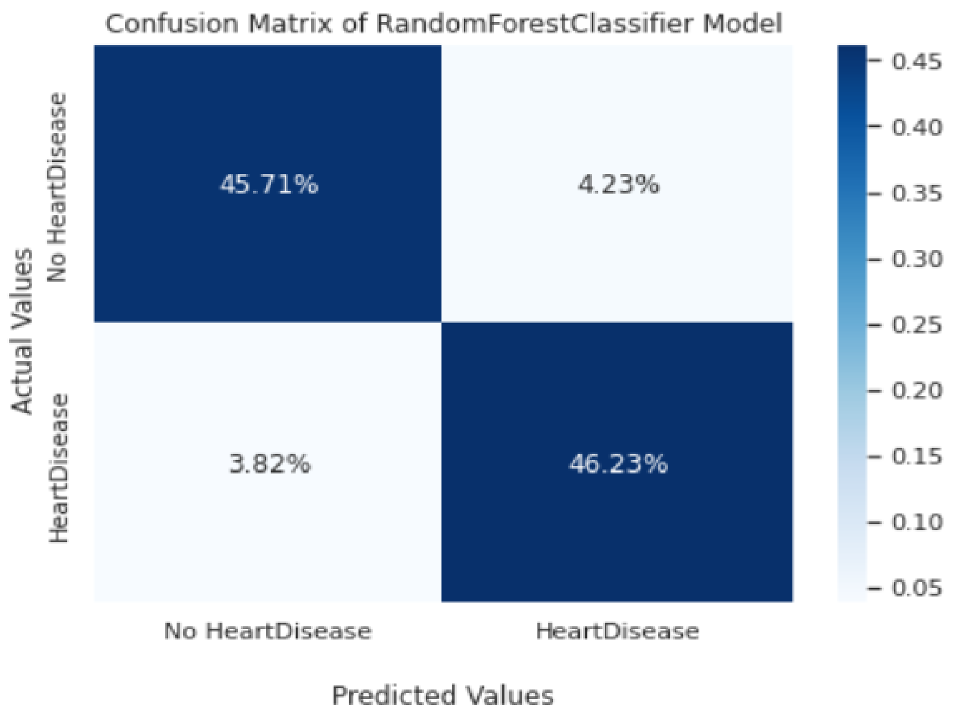
```
[ ]  model=RandomForestClassifier()
```

```
[ ]  model.fit(X_train, y_train)

     RandomForestClassifier()
```

```
[ ]  rfc_result = evaluate_model(model, X_valid, y_valid)
```

```
[ ]  print_model_result(rfc_result)

     Accuracy: 0.9194743906505143
     Precision: 0.916214178617716
     Recall: 0.9235997848121803
     F1 Score: 0.9198921576131792
     Area Under Curve: 0.9696814367295973
     Confusion Matrix:
      [[106940   9891]
      [  8947 108160]]
```

```
[ ]  draw_cm("RandomForestClassifier",rfc_result["cm"])
```



Decision Tree

Making decisions with decision trees is effective because they clearly state the problem so that all alternative solutions may be tested. Decision tree allows us to carefully consider any possible consequences of a decision. It creates a framework for calculating outcome values and success probabilities.

```
#Decision Tree

model=tree.DecisionTreeClassifier()
```

```
model.fit(X_train, y_train)
```

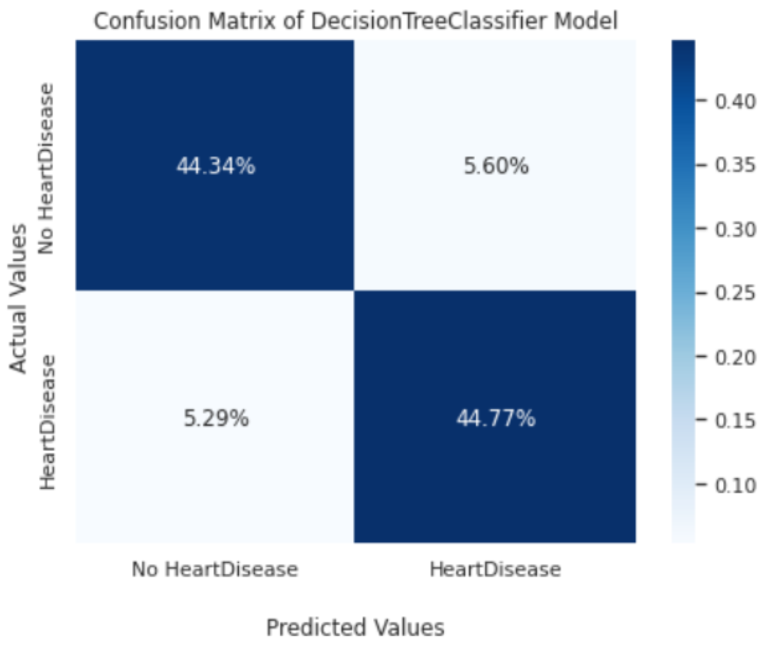
```
DecisionTreeClassifier()
```

```
dt_result = evaluate_model(model, X_valid, y_valid)
```

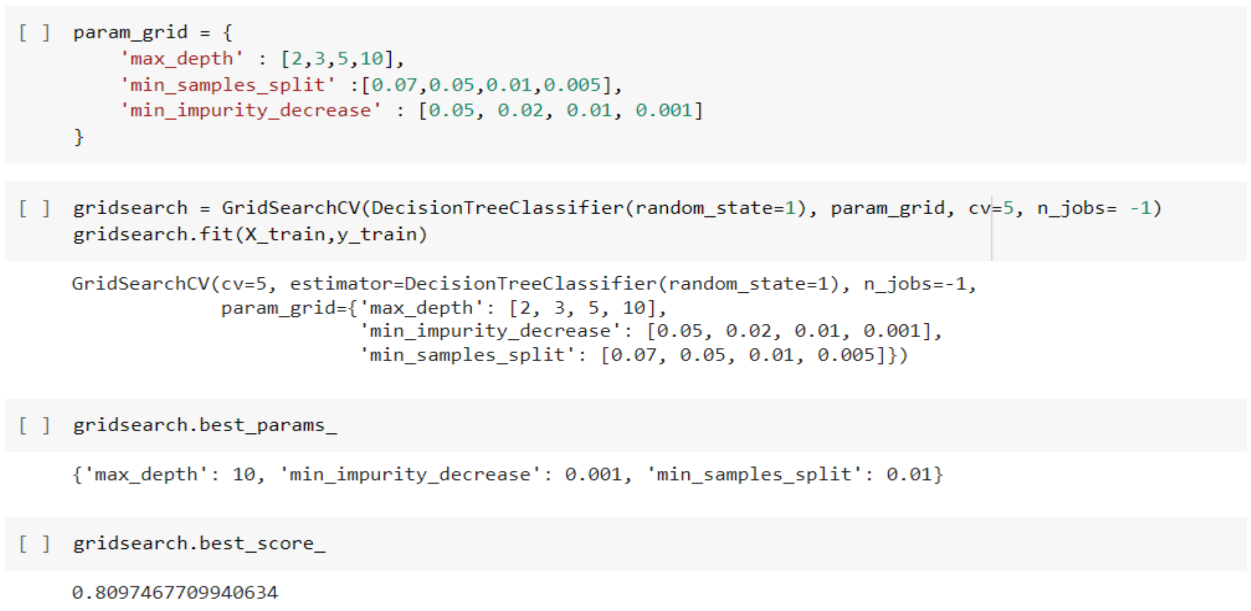
```
print_model_result(dt_result)
```

```
Accuracy: 0.8910950764732536
Precision: 0.8887700897781851
Recall: 0.8943786451706559
F1 Score: 0.8915655471519836
Area Under Curve: 0.8917678098860559
Confusion Matrix:
 [[103723  13108]
 [ 12369 104738]]
```

```
draw_cm("DecisionTreeClassifier",dt_result["cm"])
```



Grid search



```
[ ] param_grid = {
        'max_depth' : [2,3,5,10],
        'min_samples_split' :[0.07,0.05,0.01,0.005],
        'min_impurity_decrease' : [0.05, 0.02, 0.01, 0.001]
    }
```

```
[ ] gridsearch = GridSearchCV(DecisionTreeClassifier(random_state=1), param_grid, cv=5, n_jobs= -1)
    gridsearch.fit(X_train,y_train)
```

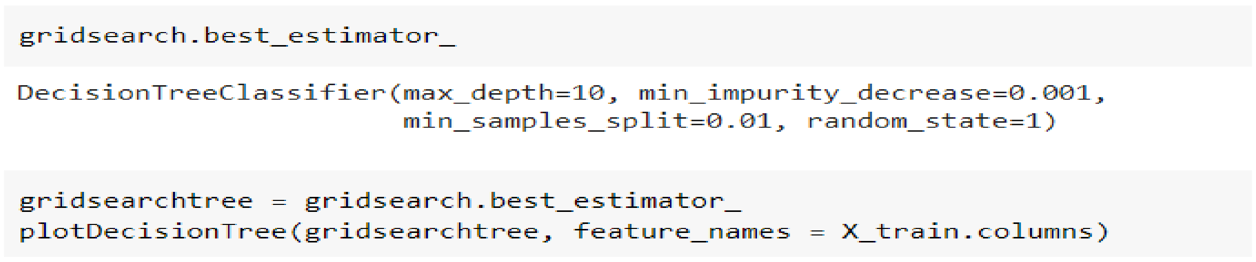
```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=1), n_jobs=-1,
             param_grid={'max_depth': [2, 3, 5, 10],
                         'min_impurity_decrease': [0.05, 0.02, 0.01, 0.001],
                         'min_samples_split': [0.07, 0.05, 0.01, 0.005]})
```

```
[ ] gridsearch.best_params_
```

```
{'max_depth': 10, 'min_impurity_decrease': 0.001, 'min_samples_split': 0.01}
```

```
[ ] gridsearch.best_score_
```

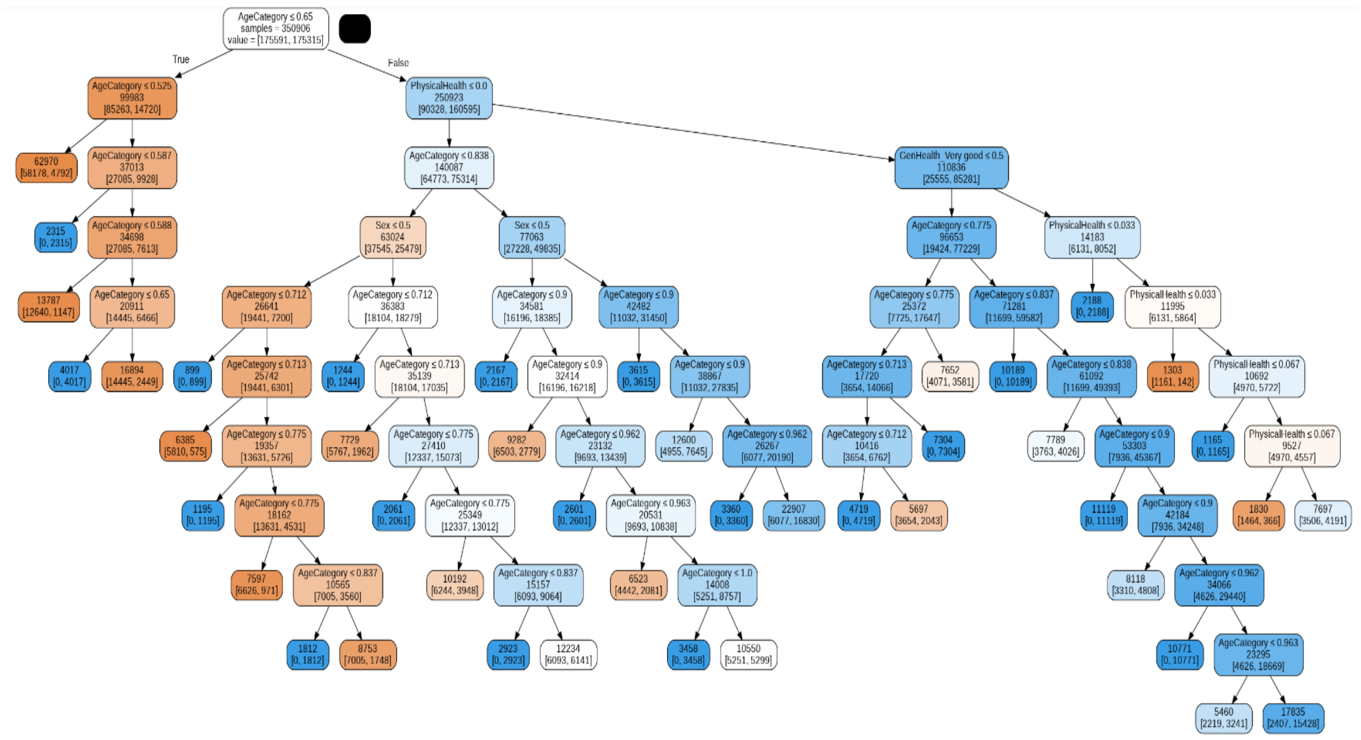
```
0.8097467709940634
```



```
gridsearch.best_estimator_
```

```
DecisionTreeClassifier(max_depth=10, min_impurity_decrease=0.001,
                       min_samples_split=0.01, random_state=1)
```

```
gridsearchtree = gridsearch.best_estimator_
plotDecisionTree(gridsearchtree, feature_names = X_train.columns)
```



```
regressionSummary(y_train, gridsearchtree.predict(X_train))
```

```
Regression statistics

                      Mean Error (ME) : -0.0256
       Root Mean Squared Error (RMSE) : 0.4342
            Mean Absolute Error (MAE) : 0.1886
```

```
regressionSummary(y_valid, gridsearchtree.predict(X_valid))
```

```
Regression statistics

                      Mean Error (ME) : -0.0252
       Root Mean Squared Error (RMSE) : 0.4325
            Mean Absolute Error (MAE) : 0.1870
```

```
classificationSummary(y_train, gridsearchtree.predict(X_train))
```

```
Confusion Matrix (Accuracy 0.8114)

       Prediction
Actual      0      1
     0 138010  37581
     1  28584 146731
```

```
classificationSummary(y_valid, gridsearchtree.predict(X_valid))
```

```
Confusion Matrix (Accuracy 0.8130)

       Prediction
Actual     0     1
     0 92002 24829
     1 18924 98183
```

Logistic Regression

Based on previous observations of a data set, the statistical analysis technique of logistic regression can be used to forecast a binary outcome, such as yes or no. A logistic regression model uses an analysis of the correlation between one or more pre-existing independent variables to predict a dependent data variable.

The technique of logistic regression has grown in significance in the field of machine learning. It enables machine learning algorithms to categorise incoming input based on previous data. The algorithms get more accurate at predicting classes within data sets when new pertinent data is added.

```
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
```

```
logit_reg = LogisticRegression(solver='liblinear', C=1e42, random_state=1)
logit_reg.fit(X_train, y_train)
```

```
LogisticRegression(C=1e+42, random_state=1, solver='liblinear')
```

```
print(pd.DataFrame({'coef': logit_reg.coef_[0]}, index=X.columns))
```

```
                                    coef
BMI                             0.784984
Smoking                         0.370586
AlcoholDrinking                -0.494361
Stroke                          1.039995
PhysicalHealth                  0.087509
MentalHealth                    0.130238
DiffWalking                     0.209328
Sex                             0.722479
PhysicalActivity               -0.031046
SleepTime                      -0.721561
Asthma                          0.275082
KidneyDisease                   0.414194
SkinCancer                      0.045557
AgeCategory                     5.267971
Race_Asian                     -0.860504
Race_Black                     -0.514394
Race_Hispanic                  -0.378614
Race_Other                     -0.285395
Race_White                     -0.387344
Diabetic_No, borderline diabetes -0.335998
Diabetic_Yes                    0.443877
Diabetic_Yes (during pregnancy) -0.976448
GenHealth_Fair                  1.559602
GenHealth_Good                  1.033959
GenHealth_Poor                  1.994703
GenHealth_Very good             0.428447
```

```
logit_reg_prob = logit_reg.predict_proba(X_valid)
```

```
logit_reg_pred = logit_reg.predict(X_valid)
```

```
logit_result = pd.DataFrame({'actual' : y_valid,
                             'p_0' : [p[0] for p in logit_reg_prob],
                             'p_1' : [p[1] for p in logit_reg_prob],
                             'predicted': logit_reg_pred})
logit_result
```

	actual	p_0	p_1	predicted
348389	1	0.674225	0.325775	0
296815	0	0.464427	0.535573	1
498134	1	0.549959	0.450041	0
538898	1	0.467675	0.532325	1
575047	1	0.085597	0.914403	1
...	...	...	...	...
477789	1	0.166636	0.833364	1
68364	0	0.342609	0.657391	1
248603	0	0.217977	0.782023	1
392711	1	0.676344	0.323656	0
180660	0	0.988944	0.011056	0

233938 rows × 4 columns

```
classificationSummary(y_train, logit_reg.predict(X_train))
```

```
Confusion Matrix (Accuracy 0.7650)

       Prediction
Actual      0      1
     0 129632  45959
     1  36504 138811
```

```
classificationSummary(y_valid, logit_reg.predict(X_valid))
```

```
Confusion Matrix (Accuracy 0.7657)

       Prediction
Actual     0     1
     0 86364 30467
     1 24339 92768
```

Logistic Regression

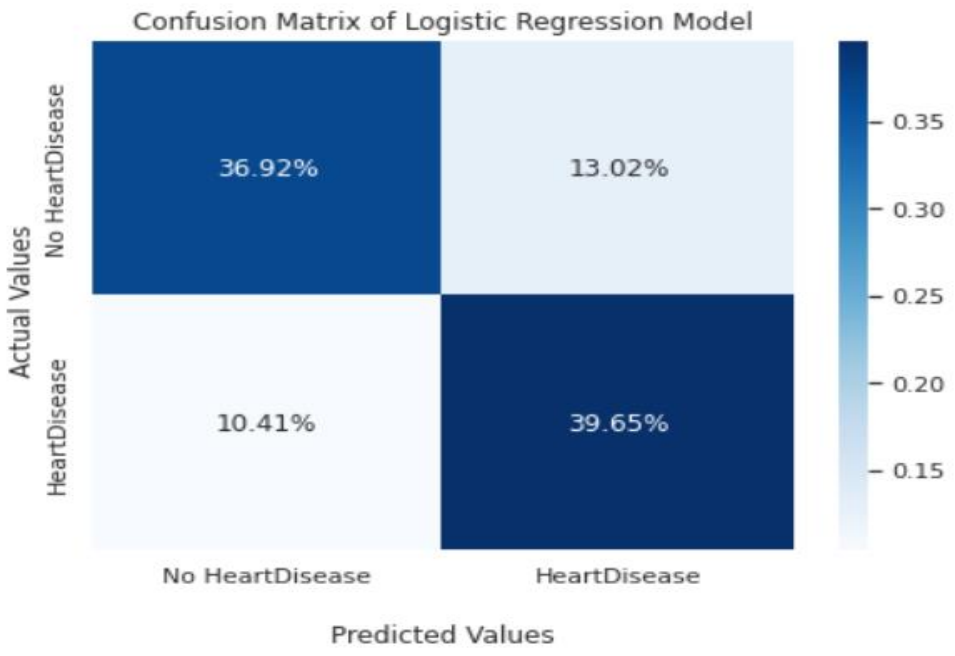
```
[ ]  model=sklin.LogisticRegression()
```

```
[ ]  model.fit(X_train, y_train)

     LogisticRegression()
```

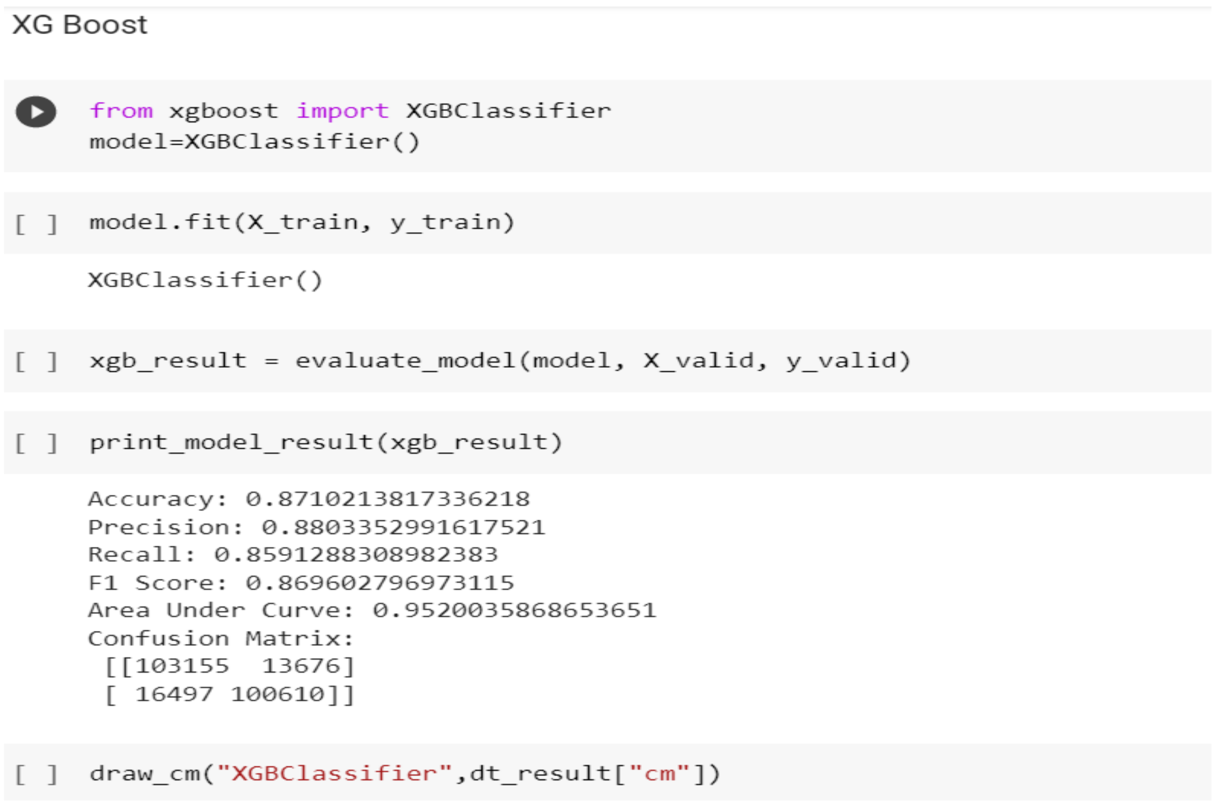
```
[ ]  logistic_regression_result = evaluate_model(model, X_valid, y_valid)
```

```
[ ]  print_model_result(logistic_regression_result)

     Accuracy: 0.7657199770879464
     Precision: 0.7527793556763775
     Recall: 0.792138813222096
     F1 Score: 0.7719577093830747
     Area Under Curve: 0.8414418540801041
     Confusion Matrix:
      [[86366 30465]
      [24342 92765]]
```

```
[ ]  draw_cm("Logistic Regression",logistic_regression_result["cm"])
```



XG Boost



XG Boost

```
from xgboost import XGBClassifier
model=XGBClassifier()
```

```
[ ]  model.fit(X_train, y_train)
```

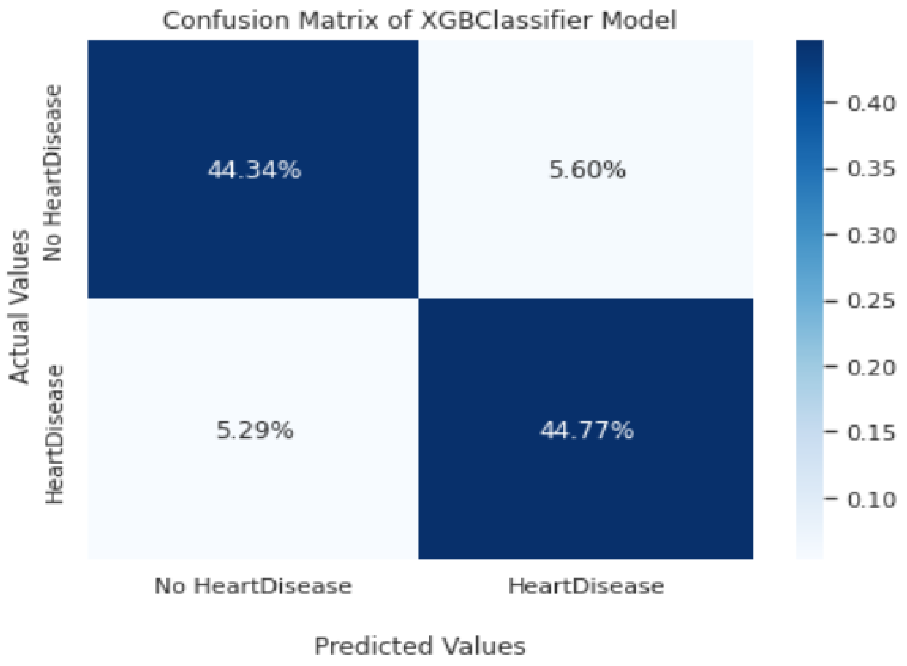
```
XGBClassifier()
```

```
[ ]  xgb_result = evaluate_model(model, X_valid, y_valid)
```

```
[ ]  print_model_result(xgb_result)
```

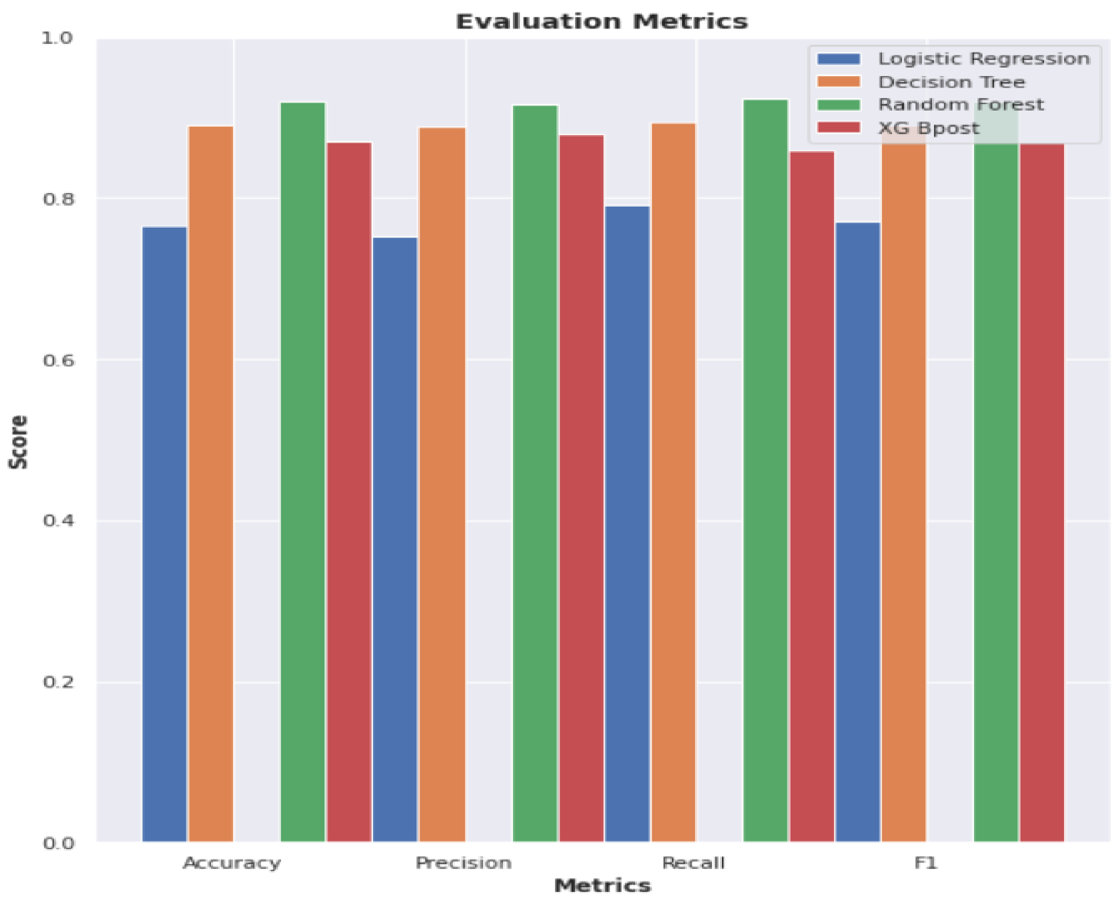
```
Accuracy: 0.8710213817336218
Precision: 0.8803352991617521
Recall: 0.8591288308982383
F1 Score: 0.869602796973115
Area Under Curve: 0.9520035868653651
Confusion Matrix:
 [[103155  13676]
 [ 16497 100610]]
```

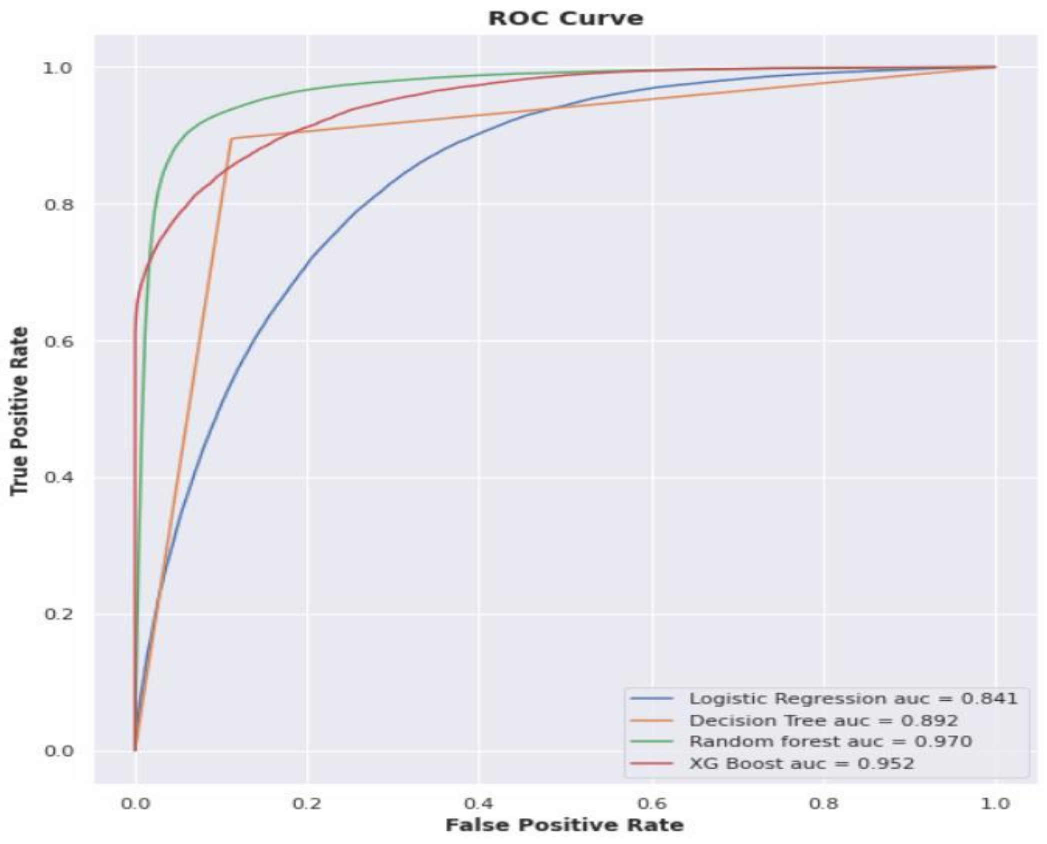
```
[ ]  draw_cm("XGBClassifier",dt_result["cm"])
```



Model Comparison (Best Model)

Among all the models, Random Forest is the best model with highest accuracy. Also, models are being evaluated on various metrics like Accuracy, Precision, Recall and F1 score. Along with accuracy, the best model is considered based on the ROC curve value.





Interpretation and Conclusion

- Adults can work on improving their overall health.
- The BMI affects heart disease. Heart disease tends to be more likely to affect people with a BMI of 20 or above.
- Reduce chances of having heart disease by doing physical activity and eating healthy food.
- Smoking causes heart disease, among other things. Contrary to popular belief, the data indicates that alcohol consumption does not increase the risk of heart disease.

- Heart disease is influenced by the number of physical health issues faced throughout the previous 30 days. On the other hand, the research indicates that the incidence of mental health issues during the same time only contributes marginally to heart disease.
- Stroke contributes to heart disease.
- Heart disease is more likely to affect those who have trouble walking.
- In terms of sexes, men typically have a larger risk of developing heart disease than women.
- The amount of physical exercise that was not work-related throughout the previous 30 days reduces the risk of developing heart disease. The general health conditions come to the same conclusion.
- A day's worth of sleep does not appear to have a significant impact on the development of heart disease. However, those who are heart disease-free often get roughly 6 to 8 hours of better sleep per day.
- Heart disease is influenced by asthma, renal illness, skin cancer, and diabetes. Asthma, however, only makes a minor impact.
- Heart disease is more common in persons over 60 than in those under 60.
- It does not appear that race affects the risk of heart disease. Any race could be affected by heart disease.

Recommendations

- People should routinely have their heart checkup. Even if a person is not currently dealing with any health issues, the person could still be in danger. Annual checkups should include a heart disease screening.
- For balancing blood sugar and cholesterol levels, doctors should emphasise the significance of a healthy diet and exercise. People should refrain from binge drinking and smoking since these vices raise the risk of developing heart disease.
- The likelihood of developing heart disease is significantly increased in anyone who already experienced a heart attack. Heart attack survivors should have routine screenings to look for any abnormalities that could be signs of heart disease.