GPU Status

In [1]:
```
#To view information about the GPU's usage, temperature, memory, and other  NVIDIA
!nvidia-smi
```

```
Wed Jan 10 15:06:38 2024
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 462.31       Driver Version: 462.31       CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  RTX A4000           WDDM | 00000000:01:00.0  On |                  Off |
| 41%   34C    P8     9W / 140W |    564MiB / 16376MiB |      6%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      4280    C+G   Insufficient Permissions        N/A      |
|    0   N/A  N/A      5752    C+G   ...artMenuExperienceHost.exe    N/A      |
|    0   N/A  N/A      9932    C+G   ...ge\Application\msedge.exe    N/A      |
|    0   N/A  N/A     10168    C+G   ...oft OneDrive\OneDrive.exe    N/A      |
|    0   N/A  N/A     11736    C+G   ...me\Application\chrome.exe    N/A      |
|    0   N/A  N/A     12748    C+G   ...5n1h2txyewy\SearchApp.exe    N/A      |
|    0   N/A  N/A     14032    C+G   ...Anywhere\AppsAnywhere.exe    N/A      |
|    0   N/A  N/A     15528    C+G   ...2txyewy\TextInputHost.exe    N/A      |
|    0   N/A  N/A     17404    C+G   C:\Windows\explorer.exe         N/A      |
+-----------------------------------------------------------------------------+
```

Library Installation

In [2]:
```
#To install the Plotly library for interactive graphs and charts
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\programdata\anaconda3\lib\site-package
s (5.14.1)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-pack
ages (from plotly) (23.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\programdata\anaconda3\lib\sit
e-packages (from plotly) (8.2.2)
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [3]:
```
#For numerical computation using data flow graphs
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in c:\programdata\anaconda3\lib\site-pac
kages (2.15.0)
Requirement already satisfied: tensorflow-intel==2.15.0 in c:\programdata\anaconda
3\lib\site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: astunparse>=1.6.0 in c:\programdata\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (1.6.3)
Requirement already satisfied: h5py>=2.9.0 in c:\programdata\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.15.0->tensorflow) (3.7.0)
Requirement already satisfied: google-pasta>=0.1.1 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in c:\programdata\anaconda3
\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.1)
Requirement already satisfied: typing-extensions>=3.6.6 in c:\programdata\anaconda
3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (4.5.0)
Requirement already satisfied: libclang>=13.0.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (16.0.6)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.60.0)
Requirement already satisfied: opt-einsum>=2.3.2 in c:\programdata\anaconda3\lib\s
ite-packages (from tensorflow-intel==2.15.0->tensorflow) (3.3.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in c:\programdata\anaconda3\li
b\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.24.3)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-pac
kages (from tensorflow-intel==2.15.0->tensorflow) (65.6.3)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.
4,!=4.21.5,<5.0.0dev,>=3.20.3 in c:\programdata\anaconda3\lib\site-packages (from
tensorflow-intel==2.15.0->tensorflow) (4.23.4)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in c:\programd
ata\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.31.
0)
Requirement already satisfied: termcolor>=1.1.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (2.4.0)
Requirement already satisfied: absl-py>=1.0.0 in c:\programdata\anaconda3\lib\site
-packages (from tensorflow-intel==2.15.0->tensorflow) (2.0.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorflow-intel==2.15.0->tensorflow) (1.14.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in c:\programdata\anaconda3\lib\si
te-packages (from tensorflow-intel==2.15.0->tensorflow) (0.2.0)
Requirement already satisfied: six>=1.12.0 in c:\programdata\anaconda3\lib\site-pa
ckages (from tensorflow-intel==2.15.0->tensorflow) (1.16.0)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-pack
ages (from tensorflow-intel==2.15.0->tensorflow) (23.0)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in c:\programda
ta\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (0.5.4)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in c:\programdat
a\anaconda3\lib\site-packages (from tensorflow-intel==2.15.0->tensorflow) (2.15.0)
Requirement already satisfied: flatbuffers>=23.5.26 in c:\programdata\anaconda3\li
b\site-packages (from tensorflow-intel==2.15.0->tensorflow) (23.5.26)
Requirement already satisfied: wheel<1.0,>=0.23.0 in c:\programdata\anaconda3\lib
\site-packages (from astunparse>=1.6.0->tensorflow-intel==2.15.0->tensorflow) (0.3
8.4)
Requirement already satisfied: markdown>=2.6.8 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(3.4.3)
Requirement already satisfied: google-auth<3,>=1.6.3 in c:\programdata\anaconda3\l
ib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorfl
ow) (2.26.1)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in c:\program
data\anaconda3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==
2.15.0->tensorflow) (0.7.2)
Requirement already satisfied: requests<3,>=2.21.0 in c:\programdata\anaconda3\lib
\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflo
```

```
w) (2.29.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in c:\programdata\anac
onda3\lib\site-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->t
ensorflow) (1.2.0)
Requirement already satisfied: werkzeug>=1.0.1 in c:\programdata\anaconda3\lib\sit
e-packages (from tensorboard<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow)
(3.0.1)
Requirement already satisfied: rsa<5,>=3.1.4 in c:\programdata\anaconda3\lib\site-
packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-intel==
2.15.0->tensorflow) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in c:\programdata\anaconda3
\lib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflo
w-intel==2.15.0->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\programdata\anaconda3\l
ib\site-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow-
intel==2.15.0->tensorflow) (0.3.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\programdata\anaconda
3\lib\site-packages (from google-auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->t
ensorflow-intel==2.15.0->tensorflow) (1.3.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib
\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-inte
l==2.15.0->tensorflow) (2023.5.7)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\programdata\anaconda
3\lib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow
-intel==2.15.0->tensorflow) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-p
ackages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-intel==2.1
5.0->tensorflow) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\l
ib\site-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow-in
tel==2.15.0->tensorflow) (1.26.15)
Requirement already satisfied: MarkupSafe>=2.1.1 in c:\programdata\anaconda3\lib\s
ite-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow-intel==2.1
5.0->tensorflow) (2.1.1)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in c:\programdata\anaconda3\li
b\site-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard<2.
16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in c:\programdata\anaconda3\lib\sit
e-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5->tensorboa
rd<2.16,>=2.15->tensorflow-intel==2.15.0->tensorflow) (3.2.2)
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [4]:

```
#Installed to handle missing data
!pip install missingno
```

```
Requirement already satisfied: missingno in c:\programdata\anaconda3\lib\site-packages (0.5.2)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from missingno) (1.10.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-packages (from missingno) (1.24.3)
Requirement already satisfied: seaborn in c:\programdata\anaconda3\lib\site-packages (from missingno) (0.12.2)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-packages (from missingno) (3.7.1)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (23.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (1.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (1.4.4)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (9.4.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (4.25.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\site-packages (from matplotlib->missingno) (3.0.9)
Requirement already satisfied: pandas>=0.25 in c:\programdata\anaconda3\lib\site-packages (from seaborn->missingno) (2.0.2)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missingno) (2022.7)
Requirement already satisfied: tzdata>=2022.1 in c:\programdata\anaconda3\lib\site-packages (from pandas>=0.25->seaborn->missingno) (2023.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->missingno) (1.16.0)
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [5]:

```
#Installed to address problem of imbalanced dataset
!pip install imblearn
```

```
Requirement already satisfied: imblearn in c:\programdata\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: imbalanced-learn in c:\programdata\anaconda3\lib\site-packages (from imblearn) (0.11.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (2.2.0)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.2.0)
Requirement already satisfied: scipy>=1.5.0 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.1.3)
Requirement already satisfied: numpy>=1.17.3 in c:\programdata\anaconda3\lib\site-packages (from imbalanced-learn->imblearn) (1.24.3)
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [6]:

```
 #To create waffle charts.
!pip install pywaffle
```

```
Requirement already satisfied: pywaffle in c:\programdata\anaconda3\lib\site-packa
ges (1.1.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\site-pac
kages (from pywaffle) (3.7.1)
Requirement already satisfied: fontawesomefree in c:\programdata\anaconda3\lib\sit
e-packages (from pywaffle) (6.5.1)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\site-p
ackages (from matplotlib->pywaffle) (0.11.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anaconda3\lib\si
te-packages (from matplotlib->pywaffle) (1.0.5)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3\lib\sit
e-packages (from matplotlib->pywaffle) (23.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3\lib\s
ite-packages (from matplotlib->pywaffle) (1.4.4)
Requirement already satisfied: numpy>=1.20 in c:\programdata\anaconda3\lib\site-pa
ckages (from matplotlib->pywaffle) (1.24.3)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib\site-
packages (from matplotlib->pywaffle) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\programdata\anaconda3\lib\si
te-packages (from matplotlib->pywaffle) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib->pywaffle) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaconda3\lib\s
ite-packages (from matplotlib->pywaffle) (4.25.0)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packa
ges (from python-dateutil>=2.7->matplotlib->pywaffle) (1.16.0)
```

```
[notice] A new release of pip is available: 23.1.2 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Library Importation

In [7]:

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
import warnings
warnings.filterwarnings('ignore')

# Artificial Neural Network Libraries
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import plot_model
# Data Visualization Libraries,
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msv
from plotly.subplots import make_subplots
import plotly.graph_objects as go
from matplotlib.gridspec import GridSpec
from plotly.subplots import make_subplots
from plotly.offline import init_notebook_mode
from pywaffle import Waffle

# Machine Learning Libraries
from imblearn.over_sampling import SMOTE
from sklearn.metrics import precision_score,f1_score,recall_score,confusion_matrix
from sklearn.model_selection import train_test_split, RandomizedSearchCV, GridSearc
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import roc_auc_score
```

```
WARNING:tensorflow:From C:\ProgramData\anaconda3\lib\site-packages\keras\src\losse
s.py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please u
se tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

Data Loading and Checks

In [59]:
```
db_data=pd.read_csv(r'C:\Users\b1605208\Desktop\Diabetes_Health_Indicators.csv').re
```

In [60]:
```
db_data.head()
```

Out[60]:

	Diabetes_Status	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	Pl
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	

5 rows × 22 columns

In [61]:
```
# Function to get the number of columns and rows in the dataset
db_data.shape
```

Out[61]:
```
(253680, 22)
```

In [62]:
```
# Function to get necessary information from the data set in a dataframe
def get_data_info(df):
    data = pd.DataFrame(index=df.columns)
    data['Data_type'] = df.dtypes
    data['Total Value'] = df.count()
    data['Null_count'] = df.isnull().sum()
    data['Unique_count'] = df.nunique()
    data = data.style.set_table_styles([{
        'selector': 'th.row_heading',
        'props': [('text-align', 'left')]
    }, {
        'selector': 'td.row_heading',
        'props': [('text-align', 'left')]
    }])
    return data

get_data_info(db_data)
```

Out[62]:

	Data_type	Total Value	Null_count	Unique_count
Diabetes_Status	float64	253680	0	3
HighBP	float64	253680	0	2
HighChol	float64	253680	0	2
CholCheck	float64	253680	0	2
BMI	float64	253680	0	84
Smoker	float64	253680	0	2
Stroke	float64	253680	0	2
HeartDiseaseorAttack	float64	253680	0	2
PhysActivity	float64	253680	0	2
Fruits	float64	253680	0	2
Veggies	float64	253680	0	2
HvyAlcoholConsump	float64	253680	0	2
AnyHealthcare	float64	253680	0	2
NoDocbcCost	float64	253680	0	2
GenHlth	float64	253680	0	5
MentHlth	float64	253680	0	31
PhysHlth	float64	253680	0	31
DiffWalk	float64	253680	0	2
Sex	float64	253680	0	2
Age	float64	253680	0	13
Education	float64	253680	0	6
Income	float64	253680	0	8

In [63]:

```
# Function to fetch the duplicate rows in the dataset
duplicates = db_data[db_data.duplicated()]
print("Duplicate Rows : ",len(duplicates))
duplicates.head()
```

```
Duplicate Rows :  23899
```

Out[63]:

	Diabetes_Status	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack
1242	2.0	1.0	1.0	1.0	27.0	1.0	0.0	0.0
1563	0.0	0.0	0.0	1.0	21.0	1.0	0.0	0.0
2700	0.0	0.0	0.0	1.0	32.0	0.0	0.0	0.0
3160	0.0	0.0	0.0	1.0	21.0	0.0	0.0	0.0
3332	0.0	0.0	0.0	1.0	24.0	0.0	0.0	0.0

5 rows × 22 columns

In [64]:

```
# eliminating 23,899 duplicate rows from the dataset df1
```

```
db_data.drop_duplicates(inplace = True)
```

Comment

- The Data Set has 253680 rows and 22 columns.
- The dataset is clean; having no null values.
- The dataset contained duplicated data points. We dropped the duplicates, totalling 23,899 rows.

Exploratory Data Analysis

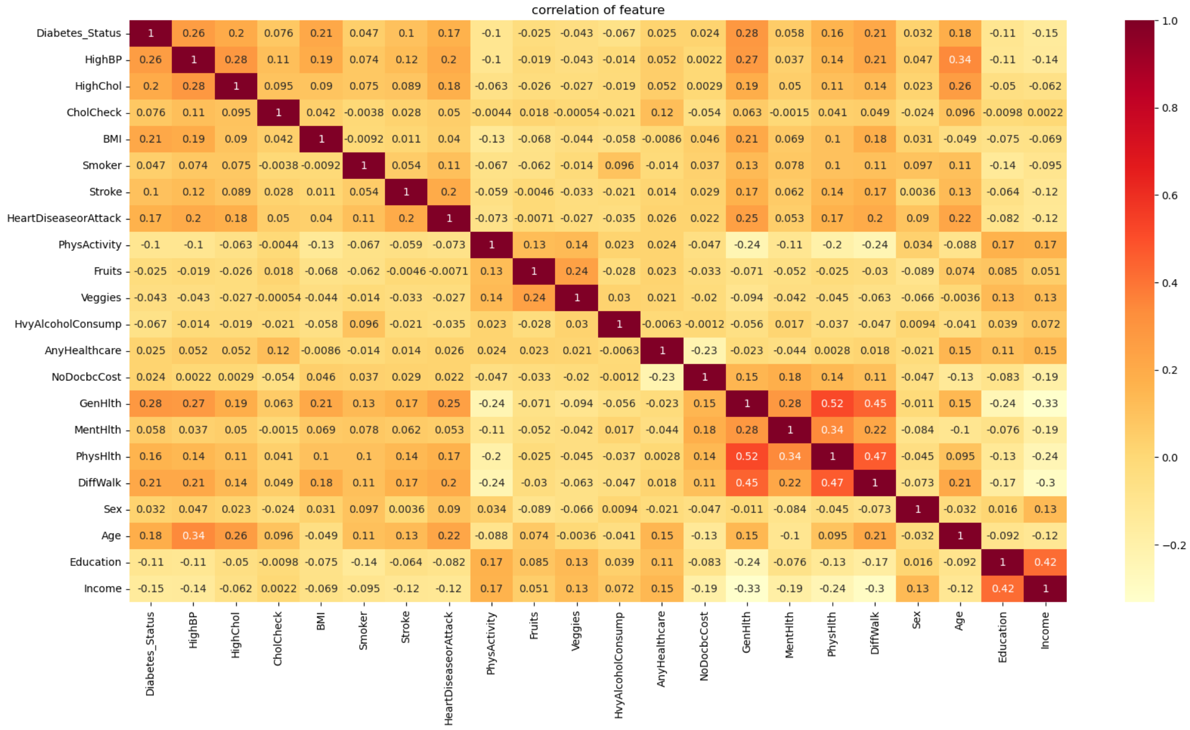
In [65]:
```
# Checking correlation between columns of dataset db_data
db_data.corr()
```

Out[65]:

	Diabetes_Status	HighBP	HighChol	CholCheck	BMI	Smoker	St
Diabetes_Status	1.000000	0.261976	0.203327	0.075701	0.212027	0.046774	0.10
HighBP	0.261976	1.000000	0.284186	0.111220	0.194126	0.074237	0.12
HighChol	0.203327	0.284186	1.000000	0.094753	0.089615	0.074627	0.08
CholCheck	0.075701	0.111220	0.094753	1.000000	0.042420	-0.003776	0.02
BMI	0.212027	0.194126	0.089615	0.042420	1.000000	-0.009196	0.01
Smoker	0.046774	0.074237	0.074627	-0.003776	-0.009196	1.000000	0.05
Stroke	0.100276	0.124426	0.089258	0.027894	0.011062	0.054438	1.00
HeartDiseaseorAttack	0.170816	0.201271	0.176279	0.049995	0.039926	0.105154	0.19
PhysActivity	-0.103408	-0.104131	-0.063266	-0.004409	-0.127864	-0.066981	-0.05
Fruits	-0.025462	-0.019329	-0.026125	0.017973	-0.067528	-0.061947	-0.00
Veggies	-0.043446	-0.042853	-0.027254	-0.000537	-0.044185	-0.013892	-0.03
HvyAlcoholConsump	-0.067164	-0.014218	-0.019104	-0.021043	-0.058334	0.096052	-0.02
AnyHealthcare	0.024911	0.052084	0.052412	0.115539	-0.008560	-0.013983	0.01
NoDocbcCost	0.023568	0.002216	0.002900	-0.054198	0.045837	0.037353	0.02
GenHlth	0.284881	0.272562	0.187929	0.062782	0.208411	0.134979	0.16
MentHlth	0.057698	0.037374	0.050212	-0.001549	0.068653	0.077715	0.06
PhysHlth	0.160485	0.144413	0.110801	0.040612	0.102844	0.100514	0.14
DiffWalk	0.210638	0.211498	0.135826	0.048969	0.182604	0.108179	0.16
Sex	0.032243	0.047155	0.022894	-0.024255	0.030989	0.096709	0.00
Age	0.184642	0.339808	0.263866	0.095996	-0.049347	0.107653	0.12
Education	-0.107742	-0.112676	-0.049838	-0.009758	-0.074568	-0.135793	-0.06
Income	-0.147102	-0.139782	-0.061871	0.002161	-0.069192	-0.095418	-0.11

22 rows × 22 columns

In [66]:
```
#visualizing the correlation on a heatmap
# Heatmap of correlation
plt.figure(figsize = (20,10))
sns.heatmap(db_data.corr(),annot=True , cmap ='YlOrRd' )
plt.title("correlation of feature")
plt.show()
```



HighBP: High blood pressure has a positive correlation, suggesting that individuals with high blood pressure may have a higher likelihood of diabetes.

HighChol: High cholesterol also shows a positive correlation with diabetes.

CholCheck: Cholesterol checking has a positive correlation, which may indicate that those who get checked might also be more likely to have diabetes, or it could reflect a general trend in healthcare behavior.

BMI: Body Mass Index (BMI) has a positive correlation, consistent with the understanding that higher BMI can be a risk factor for diabetes.

Smoker: Smoking status shows a smaller positive correlation.

Stroke: Having had a stroke shows a positive correlation with diabetes.

HeartDiseaseorAttack: Heart disease or having had a heart attack shows a significant positive correlation, suggesting a strong link with diabetes.

PhysActivity: Physical activity shows a negative correlation, implying that more active individuals might have a lower risk of diabetes.

Fruits: Eating fruits shows a slight negative correlation. Veggies: Consuming vegetables also has a slight negative correlation with diabetes.

HvyAlcoholConsump: Heavy alcohol consumption shows a small negative correlation, but this is not necessarily indicative of a protective effect; it could be due to other factors not shown in the chart.

AnyHealthcare: Having any healthcare shows a very small positive correlation with diabetes.

NoDocbcCost: Not seeing a doctor because of cost shows a small positive correlation.

GenHlth: General health perception has a positive correlation, suggesting that those who perceive their health as poorer might have a higher prevalence of diabetes.

MentHlth: Mental health status shows a small positive correlation with diabetes.

PhysHlth: Physical health status has a positive correlation.

DiffWalk: Difficulty walking has a significant positive correlation, which might be associated with diabetes complications.

Sex: Sex has a very small positive correlation, but without more context, it's hard to interpret this result.

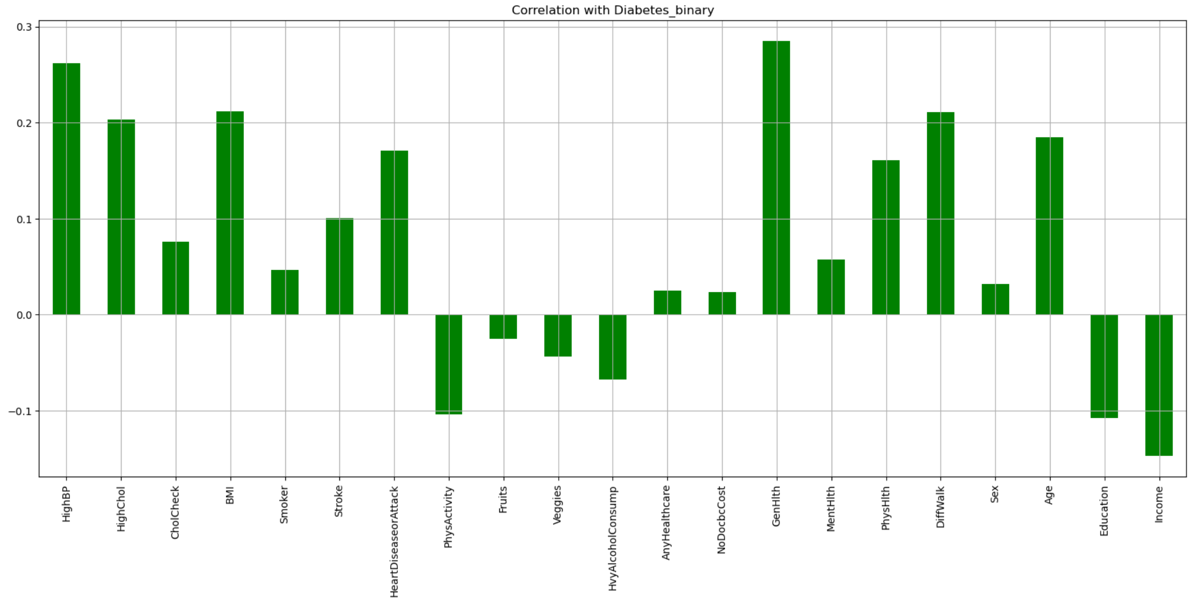
Age: Age shows a strong positive correlation, indicating that older individuals are more likely to have diabetes.

Education: Education level has a small negative correlation with diabetes.

Income: Income shows a negative correlation, suggesting that higher income individuals may have a lower incidence of diabetes.

In [67]:

```
# Let's check columns that are highly collerated with our target column, Diabetic_E
db_data.drop('Diabetes_Status', axis=1).corrwith(db_data.Diabetes_Status).plot(kinc
,title="Correlation with Diabetes_binary",color="Green");
```



Feature Engineering

In [68]:
```
"""
Creating a new Column (DiabeticS) to aid in visualization
Replacing 0 into Non-Diabetic and 1 into Diabetic
adding new column Diabetes_binary_str
"""
db_data["DiabeticS"]= db_data["Diabetes_Status"].replace({0:"Non-Diabetic",1:"Diabe
```

In [69]:
```
db_data['DiabeticS']
```

Out[69]:
```
0         Non-Diabetic
1         Non-Diabetic
2         Non-Diabetic
3         Non-Diabetic
4         Non-Diabetic
              ...
253675    Non-Diabetic
253676        Diabetic
253677    Non-Diabetic
253678    Non-Diabetic
253679        Diabetic
Name: DiabeticS, Length: 229781, dtype: object
```

Data Visualization

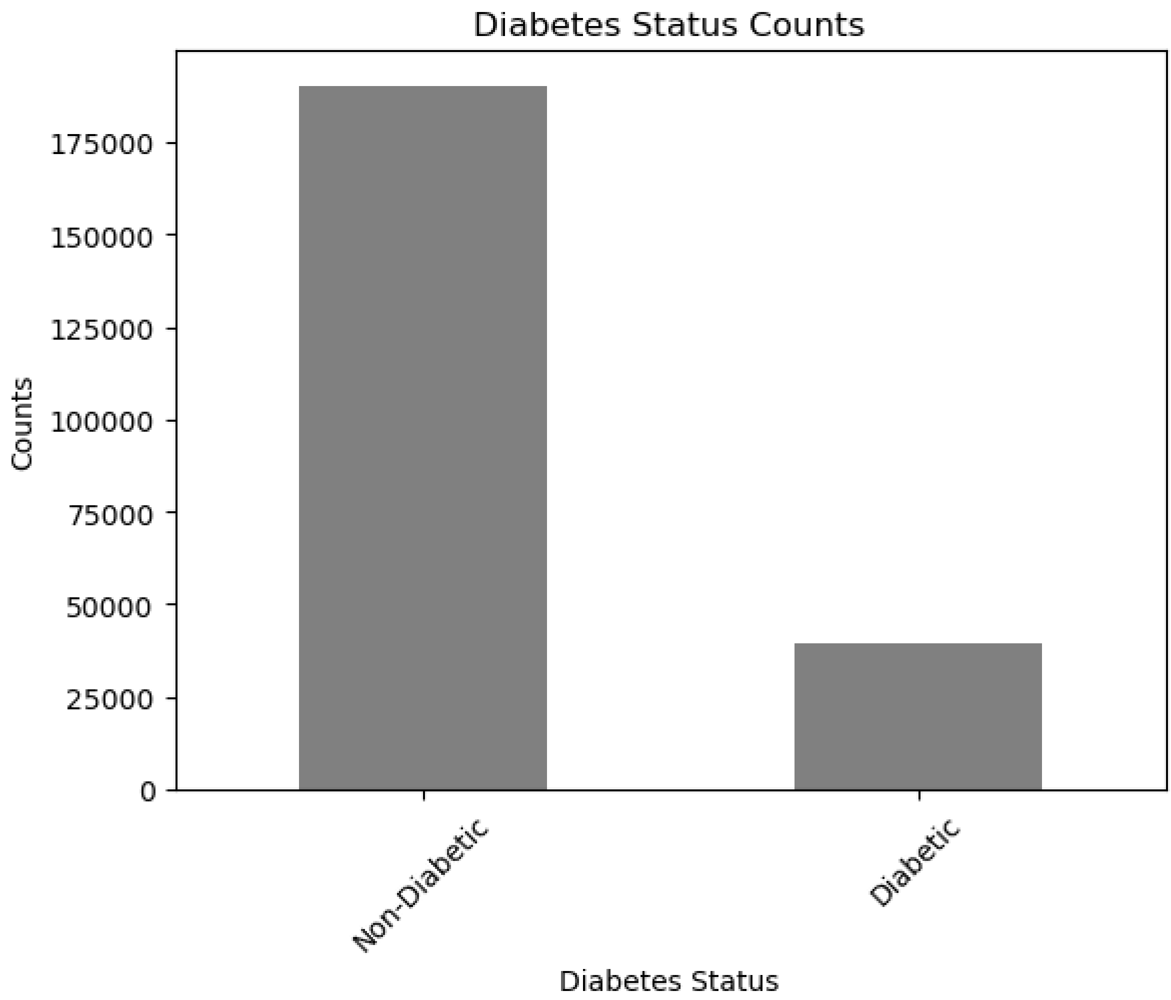
Figure 1; Distribution of Diabetic and Non Non diabetic Patients

In [70]:
```
db_data["DiabeticS"].value_counts()
```

Out[70]:
```
Non-Diabetic    190055
Diabetic         39726
Name: DiabeticS, dtype: int64
```

In [71]:
```
# Applying value_counts to the 'Diabetes_Status' column
diabetes_status_counts = db_data["DiabeticS"].value_counts()

# Visualizing the result
diabetes_status_counts.plot(kind='bar', color='grey')
plt.title('Diabetes Status Counts')
```

```
plt.xlabel('Diabetes Status')
plt.ylabel('Counts')
plt.xticks(rotation=45)
plt.show()
```



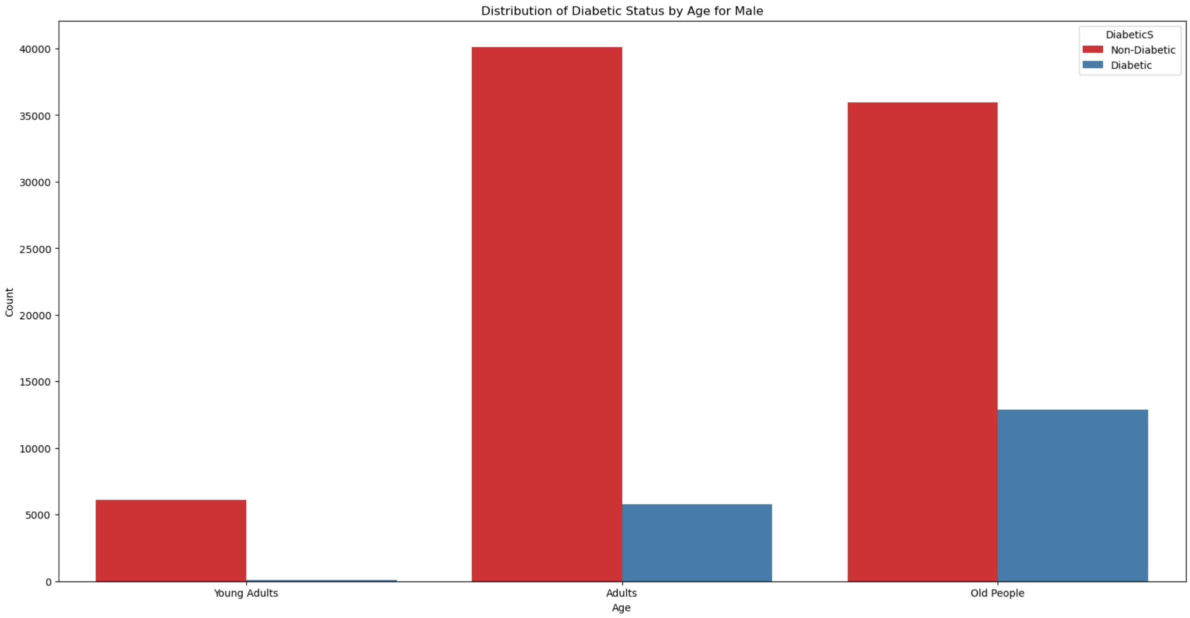
Note

Non-diabetic patients = 190,055 (82.714% of the sample size)\ Diabetic patients = 39,726 (17.29% of the sample size)

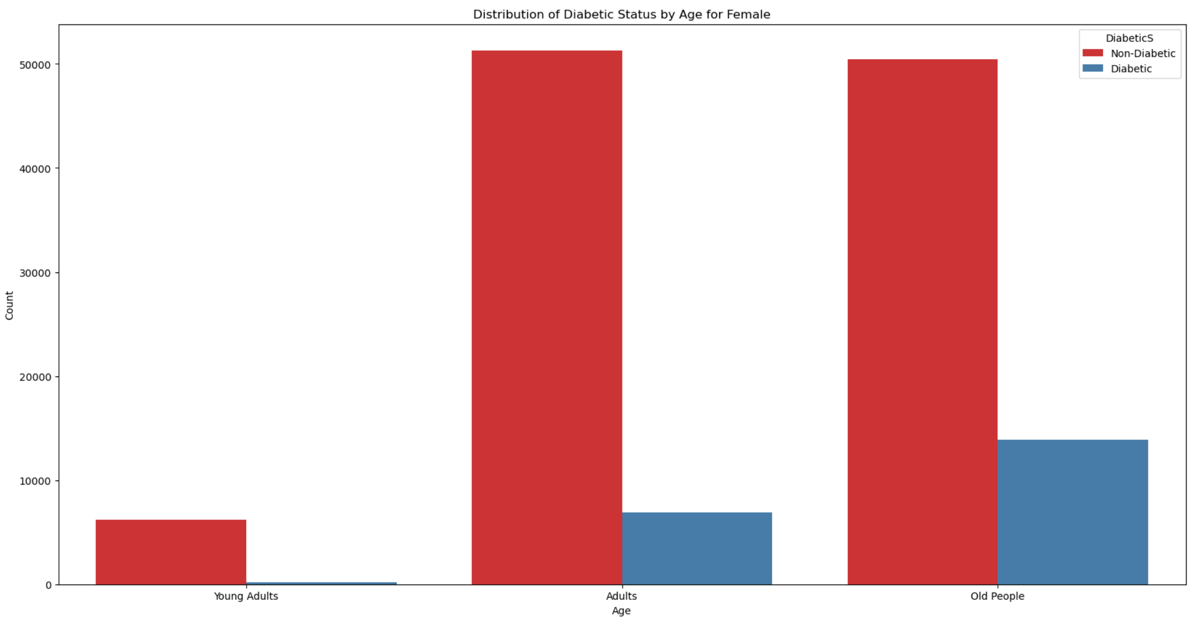
Figure 2; Distribution of Diabetes by Sex

In [74]:
```
# Assuming 'db_data' is your DataFrame

# Filter the data for Sex = 1
filtered_data = db_data[db_data['Sex'] == 1]

db_data['age_cat'] = pd.cut(db_data['Age'], bins = [0,2,8, 15], labels = ['Young Ad

# Create the plot
plt.figure(figsize=(20, 10))
sns.countplot(x='age_cat', hue='DiabeticS', data=filtered_data, palette='Set1')

# Adding titles and labels
plt.title('Distribution of Diabetic Status by Age for Male')
plt.xlabel('Age')
plt.ylabel('Count')
```

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



In [73]:
```
# Assuming 'db_data' is your DataFrame

# Filter the data for Sex = 0
filtered_data = db_data[db_data['Sex'] == 0]

db_data['age_cat'] = pd.cut(db_data['Age'], bins = [0,2,8, 15], labels = ['Young Ad

# Create the plot
plt.figure(figsize=(20, 10))
sns.countplot(x='age_cat', hue='DiabeticS', data=filtered_data, palette='Set1')

# Adding titles and labels
plt.title('Distribution of Diabetic Status by Age for Female')
plt.xlabel('Age')
plt.ylabel('Count')

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



Research Question\ What are the primary risk factors that contribute to the development and progression of diabetes, and how do these risk factors vary across different populations and demographics?

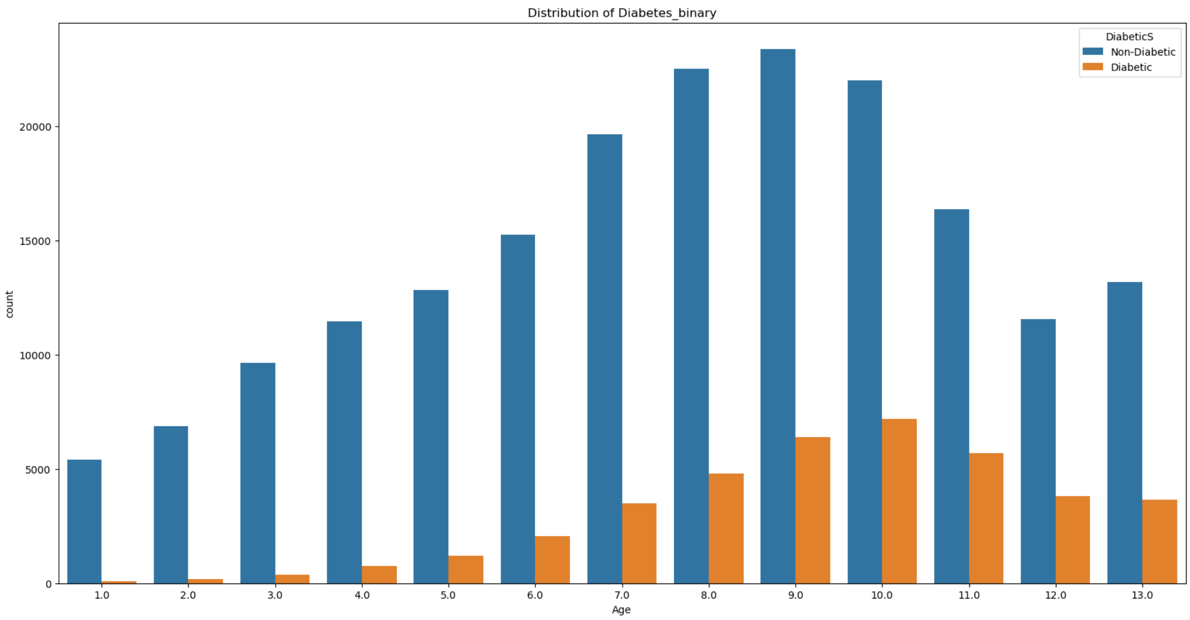
Conclusion\ Gender is identified as a primary risk factor and it is observed that

1. Women who are in their pre menopause age have a lower risk of developing type 2 diabetes compared to men.
2. The risk of developing type 2 diabetes increases in women in their poat menopause age

---

Figure3: Distribution of patients by Age

In [75]:
```
plt.figure(figsize=(20, 10))
sns.countplot(x = db_data['Age'] , hue=db_data['DiabeticS'])
plt.title('Distribution of Diabetes_binary')
plt.show()
```



In [ ]:
```
#Categorizing  BMI, Age and Glucose Columns into categorical values using range.
#Categories used
#Age: 1-2    :  Young Adults,     BMI: 0-18  : Underweight       Income Scale:   1-4
#     3-8   :  Adults,                19-24 : Ideal                              4-7
#     9-15   :  Old                   25-30 : Overweight                         8-15
#                                     30-50 : Obesity

db_data['bmi_cat'] = pd.cut(db_data['BMI'], bins = [0, 19, 25,30,50], labels = ['Ur
db_data['age_cat'] = pd.cut(db_data['Age'], bins = [0,2,8, 15], labels = ['Young Ac
db_data['income_Cat'] = pd.cut(db_data['Income'], bins = [0,4,7,15], labels = ['Low
```

In [24]:
```
db_data.head()
```

Out[24]:

	Diabetes_Status	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	Pl
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	

5 rows × 26 columns

In [25]:

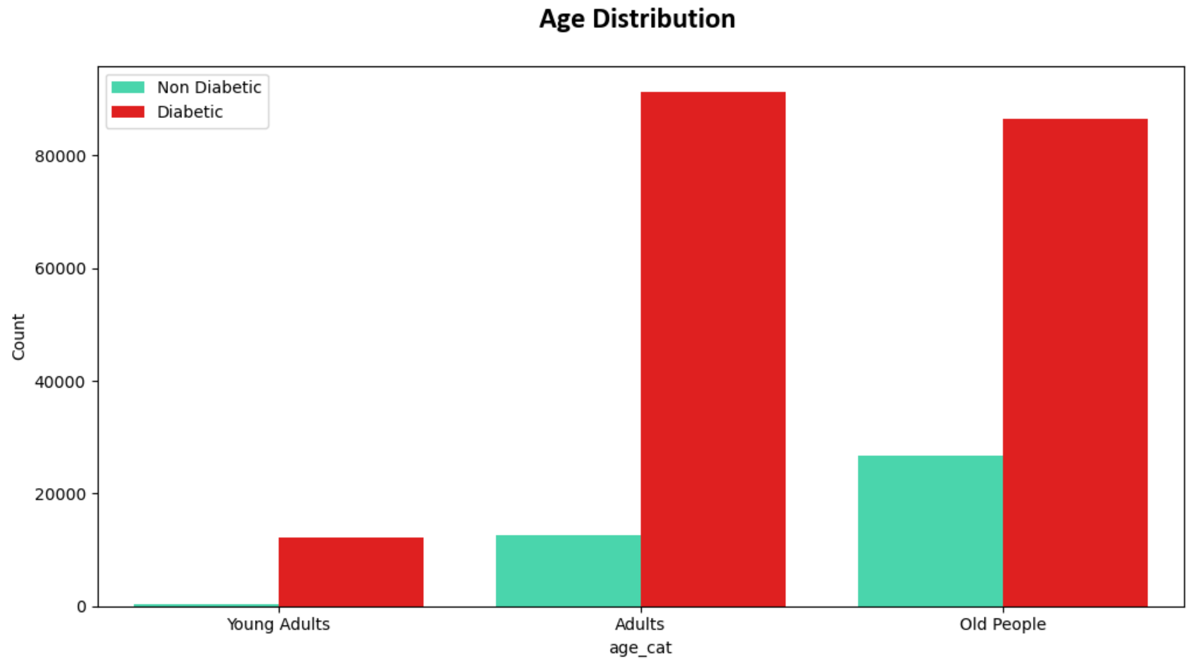
```
breakdownbyage = db_data.groupby(['age_cat','DiabeticS']).size().reset_index(name =
breakdownbyage
```

Out[25]:

	age_cat	DiabeticS	Count
0	Young Adults	Diabetic	293
1	Young Adults	Non-Diabetic	12287
2	Adults	Diabetic	12699
3	Adults	Non-Diabetic	91350
4	Old People	Diabetic	26734
5	Old People	Non-Diabetic	86418

In [26]:

```
palette2 = ['#33ECB5','#ff0000']
hfont = {'fontname':'calibri', 'weight': 'bold'}
plt.figure(figsize = (12,6))
labels=["Non Diabetic","Diabetic"]
g = sns.barplot(x ='age_cat', y = 'Count', hue = 'DiabeticS', data = breakdownbyage
h, l = g.get_legend_handles_labels()
g.legend(h, labels, title="")
plt.title("Age Distribution \n",size=18, **hfont)
plt.savefig("AgeDistribution.png")
plt.show()
```



Research Question\ What are the primary risk factors that contribute to the development and progression of diabetes, and how do these risk factors vary across different populations and demographics?

Conclusion\ Age is identified as a primary risk factor and prevalence of diabetics is observed that \

1. For most age groups, the number of non-diabetic individuals is higher than that of diabetic individuals.
2. Diabetics is most common among Old individuals, followed by Adults. However, how this factor varies across different populations and demographics cannot be established with the dataset.

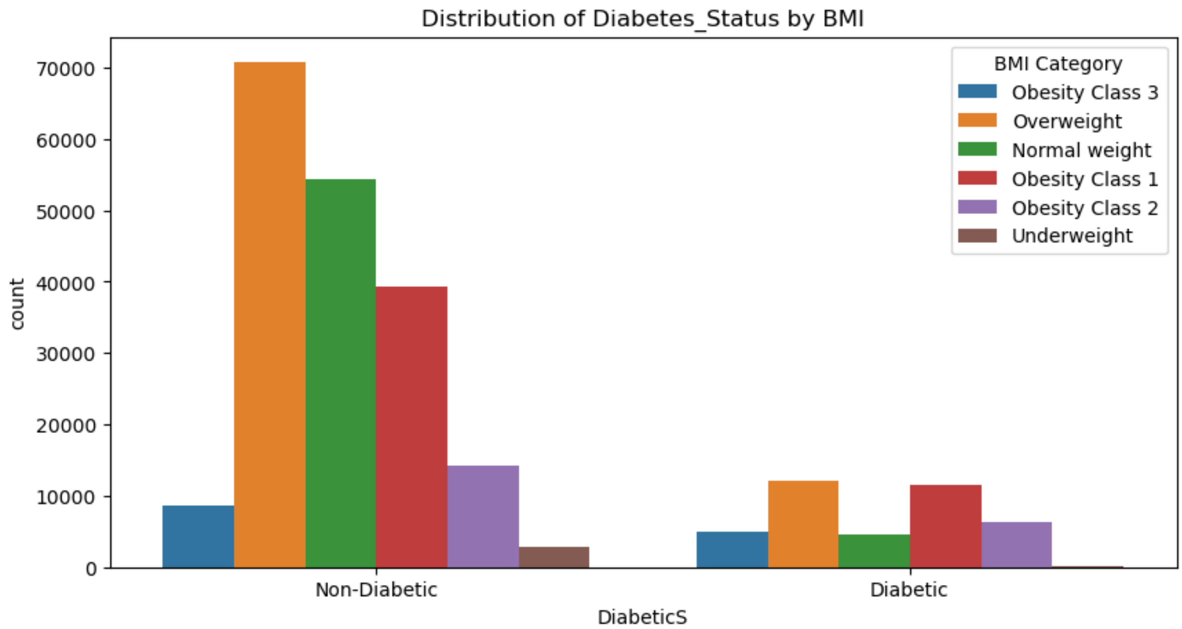
Figure4: Distribution of Diabetics patients by BMI

In [27]:
```
def BMI_Descrabtion (BMI):
    # Example mapping: Group BMI into broader categories
    if BMI < 18.5 :
        return 'Underweight'
    elif BMI >= 18.5 and BMI <= 24.9:
        return 'Normal weight'
    elif BMI >= 25 and BMI <= 29.9 :
        return 'Overweight'
    elif BMI >= 30 and BMI <= 34.9 :
        return 'Obesity Class 1'
    elif BMI >= 35 and BMI <= 39.9 :
        return 'Obesity Class 2'
    else :
        return 'Obesity Class 3'
db_data['BMI Category'] = db_data['BMI'].apply(BMI_Descrabtion)
```

In [28]:
```
#Function to visualize the BMI stats in the Dataframe
import plotly.express as px
px.pie(values=db_data['BMI Category'].value_counts(),names =db_data['BMI Category']
```

BMI Category

In [29]:
```
# Function to visualize distribution of BMI categories among diabetic patients
plt.figure(figsize=(10, 5))
sns.countplot(x = db_data['DiabeticS'] , hue=db_data['BMI Category'])
plt.title('Distribution of Diabetes_Status by BMI')
plt.show()
```



Research Question

What are the primary risk factors that contribute to the development and progression of diabetes, and how do these risk factors vary across different populations and demographics?

Conclusion\ Body Mass Index is identified as a primary risk factor and It is observed that overweight and obese people have a higher risk of type 2 diabetes. However, how this factor varies across different populations and demographics cannot be established with the dataset.

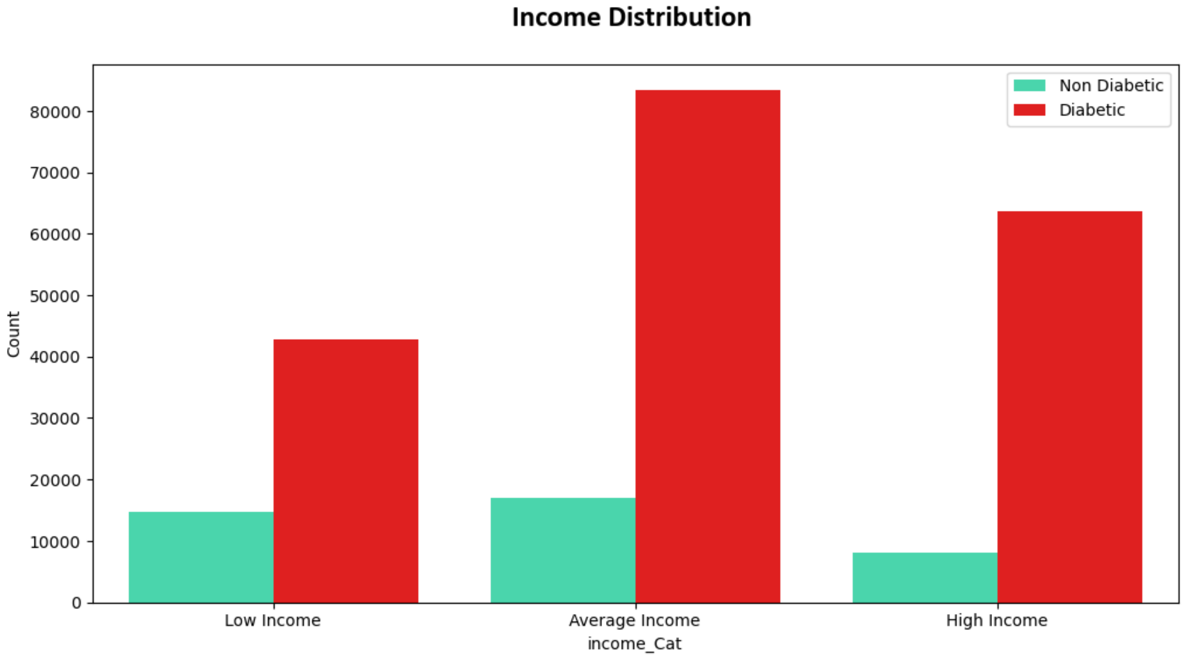
Figure5: Investigation of Income Risk Factor

In [30]:
```
breakdownbyincome = db_data.groupby(['income_Cat','DiabeticS']).size().reset_index(
breakdownbyincome
```

Out[30]:

	income_Cat	DiabeticS	Count
0	Low Income	Diabetic	14630
1	Low Income	Non-Diabetic	42798
2	Average Income	Diabetic	17043
3	Average Income	Non-Diabetic	83492
4	High Income	Diabetic	8053
5	High Income	Non-Diabetic	63765

In [31]:
```
plt.figure(figsize = (12,6))
labels=["Non Diabetic","Diabetic"]
g = sns.barplot(x ='income_Cat', y = 'Count', hue = 'DiabeticS', data = breakdownby
h, l = g.get_legend_handles_labels()
g.legend(h, labels, title="")
plt.title("Income Distribution \n",size=18, **hfont)
plt.savefig("IncomeDistribution.png")
plt.show()
```



Research Question\ What are the primary risk factors that contribute to the development and progression of diabetes, and how do these risk factors vary across different populations and demographics?

Conclusion\ Income is identified as a primary risk factor and It is observed that Diabetics is most prevalent among average income earners followed by high income earners However, how this factor varies across different populations and demographics cannot be established with the dataset.

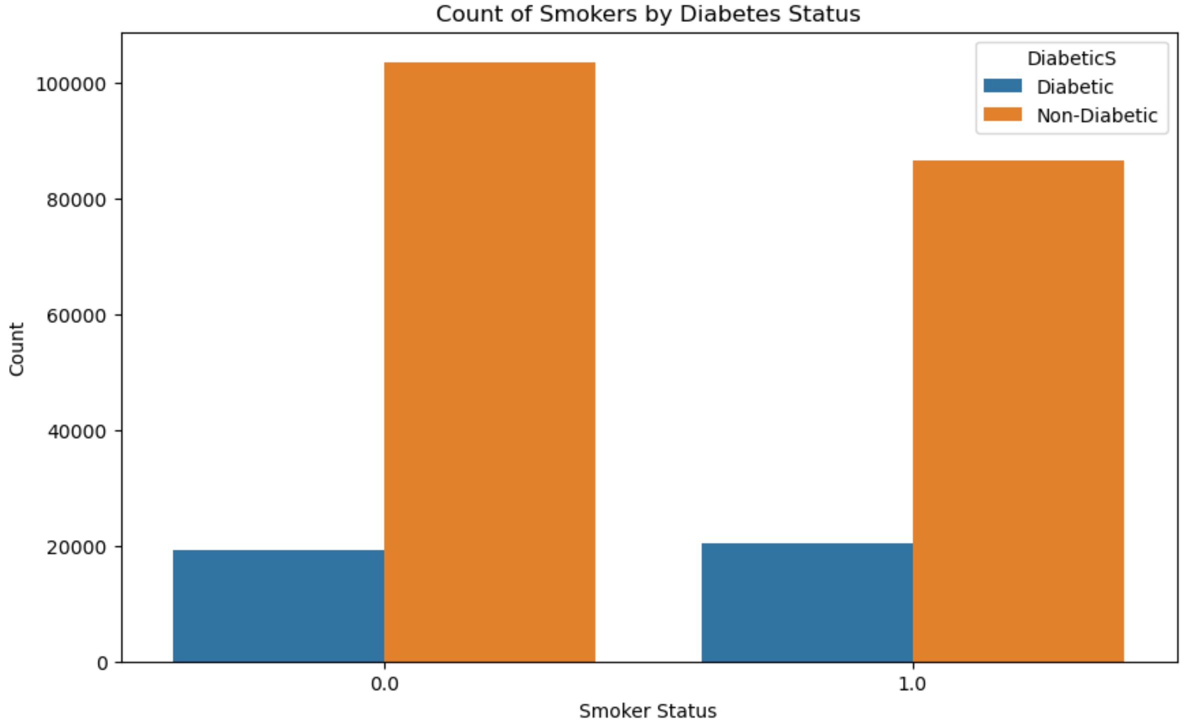
Figure6: Investigation of impact of smoking as a Diabetes Risk Factor

In [32]:
```
breakdownbystroke = db_data.groupby(['Smoker','DiabeticS']).size().reset_index(name
breakdownbystroke
```

Out[32]:

	Smoker	DiabeticS	Count
0	0.0	Diabetic	19222
1	0.0	Non-Diabetic	103559
2	1.0	Diabetic	20504
3	1.0	Non-Diabetic	86496

In [33]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming db_data is your DataFrame and the necessary grouping has been done
breakdownbystroke = db_data.groupby(['Smoker','DiabeticS']).size().reset_index(name

# Now, let's create a bar plot to visualize the relationship
plt.figure(figsize=(10, 6))
sns.barplot(x='Smoker', y='Count', hue='DiabeticS', data=breakdownbystroke)

# Add a title and labels to the plot
plt.title('Count of Smokers by Diabetes Status')
plt.xlabel('Smoker Status')
plt.ylabel('Count')

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

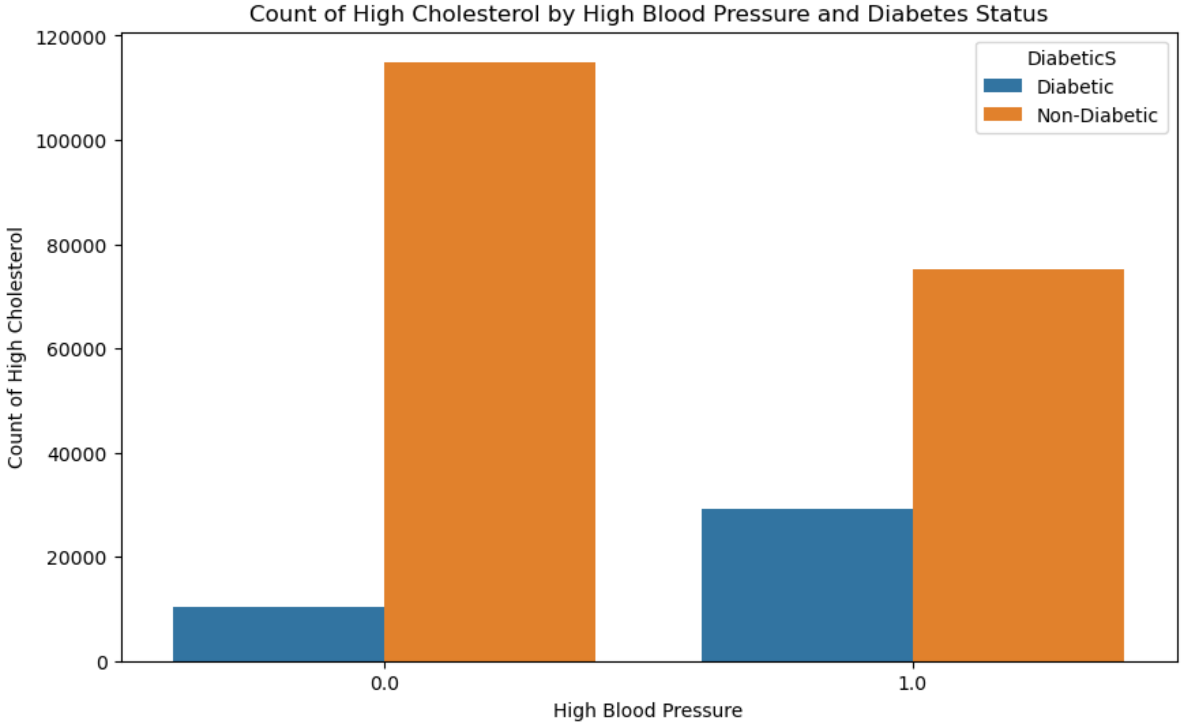


Research Question\ What are the primary risk factors that contribute to the development and progression of diabetes, and how do these risk factors vary across different populations and demographics?

Conclusion\ Smoking is identified as a primary risk factor and It is observed that Diabetics is more prevalent among smokers and less frequent among non smokers However, how this factor varies across different populations and demographics cannot be established with the dataset.

Figure7: Investigation of High BP as Diabetics Risk Factor

In [34]:
```
# db_data is the pandas DataFrame with the columns HighBP, DiabeticS, and HighChol
# Group by 'HighBP' and 'DiabeticS', and count the number of occurrences of 'HighCh
grouped_data = db_data.groupby(['HighBP', 'DiabeticS']).count().reset_index()[['Hig

# Now let's visualize this data with a bar chart using seaborn
import seaborn as sns
import matplotlib.pyplot as plt

# Create a bar plot
plt.figure(figsize=(10, 6))
sns.barplot(x='HighBP', y='HighChol', hue='DiabeticS', data=grouped_data)

# Set the title and labels
plt.title('Count of High Cholesterol by High Blood Pressure and Diabetes Status')
plt.xlabel('High Blood Pressure')
plt.ylabel('Count of High Cholesterol')

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



Research Question\ What are the primary risk factors that contribute to the development and progression of diabetes, and how do these risk factors vary across different populations and demographics?

Conclusion\ High Blood pressure is identified as a primary risk factor and It is observed that\

1. Non diabetics patients are mostly individuals with Low Blood Pressure and Low Cholesterol.
2. Conversely, Diabetics is higher among individuals with high Blood Pressure and high cholesterol. However, how this factor varies across different populations and demographics cannot be established with the dataset.

In [35]:
```
db_data.head()
```

Out[35]:

	Diabetes_Status	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	Pl
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	

5 rows × 27 columns

In [36]:
```
get_data_info(db_data)
```

Out[36]:

	Data_type	Total Value	Null_count	Unique_count
Diabetes_Status	float64	229781	0	3
HighBP	float64	229781	0	2
HighChol	float64	229781	0	2
CholCheck	float64	229781	0	2
BMI	float64	229781	0	84
Smoker	float64	229781	0	2
Stroke	float64	229781	0	2
HeartDiseaseorAttack	float64	229781	0	2
PhysActivity	float64	229781	0	2
Fruits	float64	229781	0	2
Veggies	float64	229781	0	2
HvyAlcoholConsump	float64	229781	0	2
AnyHealthcare	float64	229781	0	2
NoDocbcCost	float64	229781	0	2
GenHlth	float64	229781	0	5
MentHlth	float64	229781	0	31
PhysHlth	float64	229781	0	31
DiffWalk	float64	229781	0	2
Sex	float64	229781	0	2
Age	float64	229781	0	13
Education	float64	229781	0	6
Income	float64	229781	0	8
DiabeticS	object	229781	0	2
age_cat	category	229781	0	3
bmi_cat	category	227606	2175	4
income_Cat	category	229781	0	3
BMI Category	object	229781	0	6

In [37]:
```
db_data.head()
```

Out[37]:

	Diabetes_Status	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	Pl
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	

5 rows × 27 columns

Data Splitting

In [38]:
```
X = db_data.drop(['Diabetes_Status'],axis='columns')
X.head()
```

Out[38]:

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruit
0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.(
1	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.(
2	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.(
3	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.(
4	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.(

5 rows × 26 columns

In [39]:
```
db_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 229781 entries, 0 to 253679
Data columns (total 27 columns):
 #   Column                Non-Null Count   Dtype   
---  ------                --------------   -----   
 0   Diabetes_Status       229781 non-null  float64 
 1   HighBP                229781 non-null  float64 
 2   HighChol              229781 non-null  float64 
 3   CholCheck             229781 non-null  float64 
 4   BMI                   229781 non-null  float64 
 5   Smoker                229781 non-null  float64 
 6   Stroke                229781 non-null  float64 
 7   HeartDiseaseorAttack  229781 non-null  float64 
 8   PhysActivity          229781 non-null  float64 
 9   Fruits                229781 non-null  float64 
 10  Veggies               229781 non-null  float64 
 11  HvyAlcoholConsump     229781 non-null  float64 
 12  AnyHealthcare         229781 non-null  float64 
 13  NoDocbcCost           229781 non-null  float64 
 14  GenHlth               229781 non-null  float64 
 15  MentHlth              229781 non-null  float64 
 16  PhysHlth              229781 non-null  float64 
 17  DiffWalk              229781 non-null  float64 
 18  Sex                   229781 non-null  float64 
 19  Age                   229781 non-null  float64 
 20  Education             229781 non-null  float64 
 21  Income                229781 non-null  float64 
 22  DiabeticS             229781 non-null  object  
 23  age_cat               229781 non-null  category
 24  bmi_cat               227606 non-null  category
 25  income_Cat            229781 non-null  category
 26  BMI Category          229781 non-null  object  
dtypes: category(3), float64(22), object(2)
memory usage: 44.5+ MB
```

In [40]:
```
#Drop the object and category features
db_data.drop(columns=['DiabeticS','bmi_cat','income_Cat','age_cat','BMI Category'],
```

In [41]:
```
X = db_data.drop(['Diabetes_Status'],axis='columns')
X.head()
```

Out[41]:

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruit
0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.(
1	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.(
2	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.(
3	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.(
4	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.(

5 rows × 21 columns

In [42]:
```
y = db_data.Diabetes_Status
y[:5]
```

Out[42]:
```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: Diabetes_Status, dtype: float64
```

In [43]:
```
X.shape,y.shape
```

Out[43]:
```
((229781, 21), (229781,))
```

HYPERPARAMETER TUNING

In [44]:
```
#Python Dictionary with 5 supervised models and parameters to choose the best Model

model_params = {
    'Decision Tree': {
        'model' : DecisionTreeClassifier(),
        'params' : {
            'criterion':['gini','entropy'],
            'splitter': ['best','random'],
            'max_depth': [1,2,5,10,50,100],
            'random_state': [1,2,5,10]
        }
    },
    'Random_forest':{
        'model' : RandomForestClassifier(),
        'params' : {
            'n_estimators': [1,5,10],
            'n_jobs': [1,10,20],
         }
    },
    'Logistic_regression' :{
        'model' : LogisticRegression(),
        'params': {
            'C': [1,5,10],
            'solver':['liblinear','saga'],
            'multi_class':['auto'],
            'random_state': [1,2,10],
            'penalty': ['l1','l2','elasticnet','none']
        }
    },
    'K_Nearest_Neighbour' :{
        'model' : KNeighborsClassifier(),
        'params' :{
            'n_neighbors': [1,5,10],
            'algorithm': ["auto", "brute", "kd_tree", "ball_tree"],
            'weights': ['uniform','distance'],
            'n_jobs' : [1,10,20]
        }
    },
    'Gradient_Boost': {
        'model': GradientBoostingClassifier(),
        'params' :{
            'learning_rate': [0.01],
            'loss': ['exponential'],
            'max_depth': [50,70],
            'max_features': [1,2],
            'n_estimators': [1,10]
        }
    }
}
```

In [ ]:
```
scores = [] #check list comprehension

for model_name,mp in model_params.items():
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state
    sm = SMOTE(random_state=0)
    X_train,y_train = sm.fit_resample(X_train,y_train)
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    rs = GridSearchCV(mp['model'],mp['params'],cv=5,return_train_score=False)
    rs.fit(X_train,y_train)
    scores.append({
        'Model': model_name,
        'Best_Score': rs.best_score_,
```

```
        'Best_Parameters':rs.best_params_
    })
```

In [ ]:
```
pd.options.display.max_colwidth = 200
scoresdf = pd.DataFrame(scores,columns=['Model','Best_Score','Best_Parameters'])
scoresdf.sort_values(by='Best_Score',ascending=False, inplace=True)
scoresdf
```

Model Development

In [ ]:
```
models_results = {}

def show_model_results(X,y,model_name,model,rand_state,Datasplit=0.2,**kwargs):
    print(f'The model {model_name} with parameters : {kwargs}')
    # Create an object m of the model with parameters entered into the function
    m = model(**kwargs)
    # Split data into training and testing set
    X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=Datasplit,randon
    # Create an object of a SMOTE (Oversampling library)
    sm = SMOTE(random_state=0)
    # Performing oversampling on our train set
    X_train,y_train = sm.fit_resample(X_train,y_train)
    # Create an object of our scaling class
    scaler = StandardScaler()
    # Scale our X set
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
    # Model training
    m.fit(X_train,y_train)
    # Get model score
    score = m.score(X_test,y_test)
    # Get model predictions
    prediction = m.predict(X_test)
    # Get model precision score
    model_precision = precision_score(y_test,prediction)
    # Get model recall score
    recall = recall_score(y_test,prediction)
    # Get model F1 score
    F1 = f1_score(y_test,prediction)
    print('')
    print('****************************************************************')
    print(f'Model name:        \t {model_name}')
    print(f'Model Parameters:  \t {kwargs}')
    print(f'Model Score:       \t {score}')
    print(f'Model Precision Score:   {model_precision}')
    print(f'Model Recall Score:  \t {recall}')
    print(f'Model F1 Score:  \t {F1}')
    print('****************************************************************')
    # Call function plot_confusion matrix
    plot_confusion_matrics(m,X_test,y_test,model_name)
    return score,model_name,F1,model_precision,recall
# Function to plot our models confusion matrix
def plot_confusion_matrics(model, X_test, y_test,model_name):
    # Get model prediction
    y_pred = model.predict(X_test)
    # confusion matrix
    matrix = confusion_matrix(y_test, y_pred)
    # Dataframe to store values
    df_cm = pd.DataFrame(matrix, index = ['Diabetic', 'Healthy'],
                            columns = ['Diabetic', 'Healthy'])
    plt.figure(figsize = (12,8))
    #plot confusion matrix
```

```
    sns.heatmap(df_cm,
                annot=True,
                cmap='Greens',
                fmt='.5g',
                annot_kws={"size": 20}).set_title('Confusion matrix', fontsize = 35
    plt.xlabel('Predicted values', fontsize = 20)
    plt.ylabel('True values', fontsize = 20)
    plt.savefig(f"{model_name}.png")
    plt.show()
```

1. RANDOM FOREST CLASSIFIER

In [ ]:
```
#'n_estimators': 10, 'n_jobs': 20
rnd_state = 10
# Call show model result and pass parameters from Hyperparameter tuning
output = show_model_results(X,y,'Random Forest',RandomForestClassifier,n_estimators
# Save outputs in a dictionary
models_resultsRC = ({
    'Model Name': output[1],
    'Model Score': output[0],
    'F1 Score': output[2],
    'Precision Score': output[3],
    'Recall Score': output[4]
})
```

2. K NEAREST NEIGHBOURS

In [ ]:
```
rnd_state = 1
# Call show model result and pass parameters from Hyperparameter tuning
output = show_model_results(X,y,'KNN',KNeighborsClassifier,algorithm='auto',n_jobs=
# Save outputs in a dictionary
models_resultsKN = ({
    'Model Name': output[1],
    'Model Score': output[0],
    'F1 Score': output[2],
    'Precision Score': output[3],
    'Recall Score': output[4]
})
```

3. Gradient Boost

In [ ]:
```
rand_state = 0
# Call show model result and pass parameters from Hyperparameter tuning
output = show_model_results(X, y,'Gradient Boost',GradientBoostingClassifier,learni
# Save outputs in a dictionary
models_resultsGB = ({
    'Model Name': output[1],
    'Model Score': output[0],
    'F1 Score': output[2],
    'Precision Score': output[3],
    'Recall Score': output[4]
})
```

4. Decision Tree

In [ ]:
```
# Call show model result and pass parameters from Hyperparameter tuning

# 'criterion': 'entropy', 'max_depth': 100, 'random_state': 5, 'splitter': 'random'
output = show_model_results(X,y,'Decision Tree',DecisionTreeClassifier,10,criterion
# Save outputs in a dictionary
models_resultsDT = ({
```

```
    'Model Name': output[1],
    'Model Score': output[0],
    'F1 Score': output[2],
    'Precision Score': output[3],
    'Recall Score': output[4]
})
```

5. Logistic Regression

In [ ]:
```
# 'C': 5, 'multi_class': 'auto', 'penalty': 'l1', 'random_state': 2, 'solver': 'sag
# Call show model result and pass parameters from Hyperparameter tuning
output = show_model_results(X,y,'Logistic Regression',LogisticRegression,2,C=5,mult
# Save outputs in a dictionary
models_resultsLR = ({
    'Model Name': output[1],
    'Model Score': output[0],
    'F1 Score': output[2],
    'Precision Score': output[3],
    'Recall Score': output[4]
})
```

6. Artificial Neural Network

In [ ]:
```
from keras.callbacks import EarlyStopping
```

In [ ]:
```
sm = SMOTE(random_state=0)

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=0.2,random_state
X_train, y_train = sm.fit_resample(X_train, y_train)
y_train = keras.utils.to_categorical(y_train, 2)
y_test = keras.utils.to_categorical(y_test, 2)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
# Creating a Neural Network with 1 input layer and 3 hidden layers with activation
# Then one output layer with ***sigmoid*** function
model = keras.Sequential([
    keras.layers.Flatten(input_dim=X_train.shape[1]),
    keras.layers.Dense(500, activation='relu'),
    keras.layers.Dense(250, activation='relu'),
    keras.layers.Dense(125, activation='relu'),
    keras.layers.Dense(2, activation='sigmoid')
])
#Compile our model using Optimizer adam and loss function ,
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy']
# Model trains for 150 epochs and validates our model on X_test and y_test.
# Define early stopping criteria
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights

history = model.fit(X_train, y_train, epochs=100,validation_data=(X_test, y_test),
```

In [ ]:
```
y_predicted = model.predict(X_test)
y_predicted
```

In [ ]:
```
y_predicted_labels = [np.argmax(i) for i in y_predicted]
y_predicted_labels[:5]
```

In [ ]:
```
y_test_labels = [np.argmax(i) for i in y_test]
y_test_labels[:5]
```

In [ ]:
```
from sklearn.metrics import classification_report
# Print-Out our classification Report
print(classification_report(y_test_labels,y_predicted_labels))
```

In [ ]:
```
cm = tf.math.confusion_matrix(labels=y_test_labels,predictions=y_predicted_labels)
plt.figure(figsize=(10,7))
sns.heatmap(cm,annot=True,fmt='d',cmap='Greens')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

In [ ]:
```
preScore = precision_score(y_test_labels,y_predicted_labels, average="macro")
recScore = recall_score(y_test_labels,y_predicted_labels, average="macro")
f1Score = f1_score(y_test_labels,y_predicted_labels, average="macro")
```

In [ ]:
```
## Plot History
fig = plt.figure(figsize=(12, 8))
plt.title('ANN Accuracy', size=20)
plt.plot(history.history['accuracy'], label="Train Accuracy")
plt.plot(history.history['val_accuracy'], label="Val Accuracy")
plt.legend()
plt.savefig("ANNAccuracyGraph.png")
plt.show()
model.evaluate(X_test,y_test)
score = np.round(model.evaluate(X_test, y_test, verbose=0)[1], 3)
print(f'Neural Network score     ========>>> {score}')
models_resultsANN = ({
    'Model Name': 'Artificial Neural Network',
    'Model Score': score,
    'F1 Score': f1Score,
    'Precision Score': preScore,
    'Recall Score': recScore
})
```

Results

In [ ]:
```
myalgorithms = {
    'Logistic Regression':models_resultsLR,
    'Gradient Boost': models_resultsGB,
    'Decision Tree': models_resultsDT,
    'Random Forest Classifier': models_resultsRC,
    'K Nearest Neighbour': models_resultsKN,
    'Artificial Neural Network': models_resultsANN
}
myalgorithms
```

In [ ]:
```
myalgorithmsdf = pd.DataFrame(myalgorithms.values(),columns=['Model Name','Model Sc
myalgorithmsdf.sort_values(by='Model Score',ascending=False, inplace=True)
myalgorithmsdf
```

In [ ]:
```
g = sns.catplot(x='Model Name', y='Model Score', data=myalgorithmsdf,
                height=6, aspect=3, kind='bar', legend=True)
g.fig.suptitle('Model Accuracy Scores', size=25, y=1.1)
ax = g.facet_axis(0,0)
ax.tick_params(axis='x', which='major', labelsize=15)
for p in ax.patches:
    ax.text(p.get_x() + 0.27,
            p.get_height() * 1.02,
           '{0:.2f}%'.format(p.get_height()*100),
            color='black',
            rotation='horizontal',
```

```
        size='x-large')
plt.savefig("ModelAccuracy.png")
```

SHAP

In [ ]:
```
!pip install shap
```

In [ ]:
```
import shap #For model interpretability
```

In [ ]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=2)
sm = SMOTE(random_state=0)
X_train,y_train = sm.fit_resample(X_train,y_train)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Feature Importance

In [ ]:
```
RFModel = RandomForestClassifier(n_estimators=10,n_jobs=20)
RFModel.fit(X_train,y_train)
feature_importance_df = pd.DataFrame()
feature_importance_df['feature'] = X.columns
feature_importance_df['importance'] = RFModel.feature_importances_

feature_importance_df = feature_importance_df.sort_values('importance',ascending=Fa
print('***Random Forest***')
plt.figure(figsize=(10,10))
sns.barplot(x='importance',y='feature',data=feature_importance_df[:15])
plt.savefig("ShapValueRandomForest.png")
plt.show()
```

In [ ]:
```
# Gradient Boost Feature Importance
```

In [ ]:
```
GBModel = GradientBoostingClassifier(learning_rate=0.01,loss='exponential',max_dept
GBModel.fit(X_train,y_train)
feature_importance_df = pd.DataFrame()
feature_importance_df['feature'] = X.columns
feature_importance_df['importance'] = GBModel.feature_importances_

feature_importance_df = feature_importance_df.sort_values('importance',ascending=Fa
print('***Gradient Boost***')
plt.figure(figsize=(10,10))
sns.barplot(x='importance',y='feature',data=feature_importance_df[:15])
plt.savefig("ShapValueGradientBoost.png")
plt.show()
```

In [ ]:
```
```