

SMART BRIDGE INTERNSHIP TRAINING
VELLORE INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Campus: Vellore

Date: 30.06.2023

Project Title: Diabetes Prediction

Team Members:

S. No	Name	Reg No	Mail ID
1.	GURUARUNACHALAM G	20MID0068	guruarunachalam.g2020@vitstudent.ac.in
2.	PRABHATH SAI G B	20MID0137	prabhathsai.gb2020a@vitstudent.ac.in
3.	SELVA MANOOJ M	20MID0189	selvamanooj.m2020@vitstudent.ac.in
4.	GAGAN SAI G B	20MID0192	gagansai.gb2020@vitstudent.ac.in

Overview:

The Diabetes Prediction project utilizes machine learning to predict the likelihood of an individual developing diabetes. It involves collecting a dataset with relevant information about individuals, preprocessing the data, selecting or engineering informative features, training a suitable machine learning model, and evaluating its performance. The deployed model can be used to provide risk predictions for individuals, aiding in early intervention and prevention strategies.

Purpose:

The purpose of a diabetes prediction project is to develop a system or model that can accurately predict the likelihood of an individual developing diabetes. By leveraging machine learning algorithms and analyzing various factors associated with diabetes risk, this project aims to provide early detection and intervention opportunities. The ultimate goal is to improve healthcare outcomes by enabling proactive measures, personalized interventions, and targeted resource allocation to individuals at higher risk of developing diabetes.

Problem Statement and Problem Understanding (Literature Survey):

- **Problem:** The healthcare organization wants to address the increasing prevalence of diabetes and its associated health risks. Early detection and intervention are crucial in managing diabetes and reducing its complications. This Project aims to develop a diabetes prediction system that can identify individuals at high risk of developing diabetes.
- **Objective:** The objective of this project is to build a predictive model that accurately identifies individuals who are at risk of developing diabetes in the future. The model should utilize various data points such as BMI, Age, Physical Health, Blood Pressure and so on to make reliable predictions.

- **Target Users:** The primary users of the diabetes prediction system will be healthcare professionals, including physicians, endocrinologists, and other specialists. The system can also be utilized by public health agencies, insurance companies, and wellness programs to identify individuals who may benefit from targeted interventions and preventive measures.
- **Data Requirements:** This project requires a comprehensive and representative dataset containing a diverse range of individuals, including both diabetic and non-diabetic cases. The dataset should include relevant features such as Age, Gender, BMI, Blood Pressure, Cholesterol, and Physical Health.
- **Benefits:** The diabetes prediction system can bring several benefits to the healthcare organization and the individuals it serves. It can facilitate early detection and intervention, enabling healthcare professionals to provide targeted care and preventive measures. By identifying high-risk individuals, resources can be allocated more efficiently, and personalized interventions can be implemented, ultimately improving health outcomes and reducing the burden of diabetes-related complications.

Business Requirements:

- A Model that predicts the presence Diabetes, Pre-Diabetes and the absence of Diabetes in an individual with a good accuracy using the required data.
- An easily understandable Interface for the users.

Literature Survey:

"Machine Learning for Diabetes Prediction: A Review" by E. ?ahin et al. This paper provides a comprehensive review of the latest research on machine learning for diabetes prediction. The authors discuss the challenges, approaches, and evaluation metrics used in various studies.

"Predicting Type 2 Diabetes Mellitus Using Machine Learning Techniques" by S. Chakraborty et al. This paper proposes a machine learning approach for predicting the risk of developing type 2 diabetes mellitus based on demographic and clinical data. The authors compare various algorithms and feature selection techniques and evaluate their performance.

"Deep Learning for Diabetes Prediction: A Review" by Y. Zhao et al. This paper provides a review of the latest research on deep learning for diabetes prediction, with a focus on the use of convolutional neural networks (CNNs) and recurrent neural networks (RNNs).

"Diabetes Prediction Using Machine Learning Techniques: A Comparative Study" by S. B. Gawali and R. K. Kamat. This paper compares the performance of various machine learning algorithms, including logistic regression, decision trees, and neural networks, for diabetes prediction using clinical and demographic data.

"Machine Learning for Early Detection of Diabetic Retinopathy" by A. Gulshan et al. This paper proposes a deep learning approach for the early detection of diabetic retinopathy based on retinal images. The authors use a CNN to classify images into different stages of the disease and achieve high accuracy.

Social Or Business Impact:

- **Early Intervention and Prevention:** By accurately predicting the likelihood of individuals developing diabetes, the project enables early intervention and preventive measures. This can lead to improved health outcomes, reduced complications, and better management of the disease. Early detection allows healthcare professionals to implement lifestyle interventions, recommend medication, and provide personalized care plans to mitigate the risk of diabetes.
- **Resource Allocation and Cost Savings:** The project helps optimize resource allocation by identifying high-risk individuals who require targeted interventions. Healthcare organizations and public health agencies can allocate resources more efficiently, focusing on prevention and early intervention strategies for individuals at the highest risk. This targeted approach can result in cost savings by reducing the need for expensive treatments and hospitalizations associated with advanced diabetes complications.
- **Personalized Healthcare:** The project enables personalized healthcare by providing risk assessments tailored to each individual. Healthcare professionals can utilize the risk predictions to offer personalized advice, education, and interventions to individuals based on their specific diabetes risk profiles. This personalized approach empowers individuals to make informed decisions about their health and engage in proactive self-care.
- **Insurance and Wellness Programs:** Insurance companies and wellness programs can leverage the diabetes prediction project to identify individuals at risk and offer proactive support and preventive services. By incentivizing healthy behaviours, offering wellness programs, and providing tailored insurance plans, these organizations can promote healthier lifestyles, reduce the risk of diabetes, and potentially lower insurance costs associated with diabetes-related claims.

Dataset Source: <https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset>

Approaches and Solutions:

i. Data Collection:

There are three datasets in the given kaggle dataset link.

- a) diabetes _ 012 _ health _ indicators _ BRFSS2015.csv
- b) diabetes _ binary _ 5050split _ health _ indicators _ BRFSS2015.csv
- c) diabetes _ binary _ health _ indicators _ BRFSS2015.csv

We took the first dataset which is having three features (No Diabetes, Pre-Diabetes, Diabetes) and totally 22 features including target variable.

ii. Data Pre-processing:

The checking and handling of null values is done and the outlier analysis is also done using plots. Encoding the binary values in the attributes into the string values for easy exploration (visual analysis) in the upcoming processes.

iii. Data Exploration:

Exploring the dataset using many visual analysis and commands like counting the values in each attribute. Visual analysis with Univariate, Bivariate and Multivariate explorations are done to understand more about the dataset's insights.

iv. Splitting the data:

Split the dataset into X and y. And making the X and y compatible for splitting into training and test data.

v. Model Building:

As our project is a classification based, we used several classification algorithms.

1. Logistic Regression.
2. Naive Bayes.
3. KNN
4. Decision Tree(cart)
5. Random Forest.
6. Gradient Boosting.
7. XgBoost

Out of this, we took the model with good accuracy. After fitting the model with training data, we perform model evaluation.

vi. Model Evaluation:

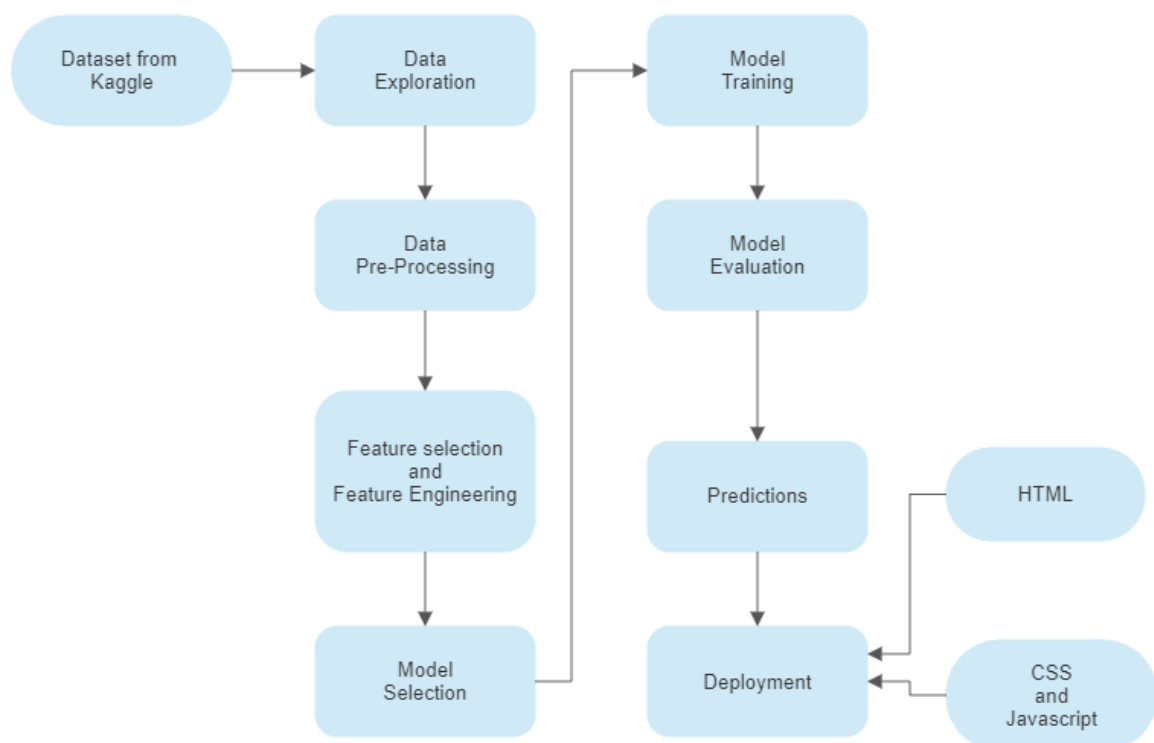
Then we test the model with test data that we split and then performing evaluation of the model. Model evaluation includes calculating accuracy, precision, recall and exploring which model has the more feature importance, everything is done.

vii. Model Deployment:

After that model is deployed using flask application in the format (.pkl).

Theoretical Analysis:

• **Block Diagram:**



Hardware and Software Requirements:

- **Hardware:**

- 1) **CPU:** A contemporary processor that can handle the computational demands of developing machine learning models. For speedier processing, multi-core CPUs are preferred.
- 2) **RAM:** The dataset and model training procedure will fit in enough RAM. Depending on the size and complexity of the dataset, a different quantity may be needed, although 8GB or more is typically advised.

- 3) **Storage:** Enough room to keep the dataset, the programme, and any intermediate files produced throughout the project.

- **Software:**

- 1) Jupyter Notebook Software
- 2) Spyder Software
- 3) Anaconda Software
- 4) Web Browser that supports HTML5

Experimental Investigation (Code with Explanation):

- Importing the libraries for data import, manipulation, EDA, model building, model evaluation and deployment.
 1. For Data import we are using the dataset with file format(.csv), so we used pandas.
 2. For Data manipulation we used pandas and numpy.
 3. For Exploratory Data Analysis, to make the visualisations good looking we used seaborn and matplotlib.
 4. For the ml models, the scikit library, xgboost, lightgbm are used. The necessary separate modules are imported for the specific particular models.
 5. For the model evaluation, we used the module from the sklearn.metrics which provides many evaluation techniques.
 6. For the model deployment, we have to save the model in pickle format, so the library pickle is used.

Importing all the necessary Libraries.

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
import xgboost as xgb
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, roc_auc_score, roc_curve
```

- Importing the dataset from the local directory and storing it in the 'data' variable

1. Importing the dataset

```
data = pd.read_csv("D:/Datasets/diabetes/diabetes_012_health_indicators_BRFSS2015.csv")
```

- Basic exploration about the dataset like knowing the column names, datatypes of the columns, shape, info, value counts.

Basic exploration about the dataset

```
data.columns
```

```
Index(['Diabetes_012', 'HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker',  
      'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies',  
      'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth',  
      'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age', 'Education',  
      'Income'],  
      dtype='object')
```

```
data.head()
```

	Diabetes_012	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	
0	0.0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	...	1.0	0.0	5.0	
1	0.0	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.0	...	0.0	1.0	3.0	
2	0.0	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.0	...	1.0	1.0	5.0	
3	0.0	1.0	0.0	1.0	27.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0	
4	0.0	1.0	1.0	1.0	24.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	2.0	

5 rows × 22 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 253680 entries, 0 to 253679  
Data columns (total 22 columns):  
#   Column                Non-Null Count  Dtype    
---  ---                  
0   Diabetes_012          253680 non-null float64  
1   HighBP                253680 non-null float64  
2   HighChol              253680 non-null float64  
3   CholCheck             253680 non-null float64  
4   BMI                   253680 non-null float64  
5   Smoker                253680 non-null float64  
6   Stroke                253680 non-null float64  
7   HeartDiseaseorAttack  253680 non-null float64  
8   PhysActivity          253680 non-null float64  
9   Fruits                253680 non-null float64  
10  Veggies               253680 non-null float64  
11  HvyAlcoholConsump     253680 non-null float64  
12  AnyHealthcare         253680 non-null float64  
13  NoDocbcCost           253680 non-null float64  
14  GenHlth               253680 non-null float64  
15  MentHlth              253680 non-null float64  
16  PhysHlth              253680 non-null float64  
17  DiffWalk              253680 non-null float64  
18  Sex                   253680 non-null float64  
19  Age                   253680 non-null float64  
20  Education              253680 non-null float64  
21  Income                253680 non-null float64  
dtypes: float64(22)  
memory usage: 42.6 MB
```

```
int64_col = data.select_dtypes(include = 'int64')  
print("Integer Columns: ", int64_col.columns.to_list())  
float64_col = data.select_dtypes(include = 'float64')  
print("Float Columns: ", float64_col.columns.to_list())  
object_col = data.select_dtypes(include = 'object')  
print("Object Columns: ", object_col.columns.to_list())  
numeric_col = data.select_dtypes('number')  
print("Numeric Columns: ", numeric_col.columns.to_list())
```

```
Integer Columns: []  
Float Columns: ['Diabetes_012', 'HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income']  
Object Columns: []  
Numeric Columns: ['Diabetes_012', 'HighBP', 'HighChol', 'CholCheck', 'BMI', 'Smoker', 'Stroke', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'HvyAlcoholConsump', 'AnyHealthcare', 'NoDocbcCost', 'GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk', 'Sex', 'Age', 'Education', 'Income']
```

```

for column in numeric_col:
    value_counts = data[column].value_counts(dropna = False).reset_index()
    value_counts.columns = ['Value', 'Count']
    print(f"Value counts for column '{column}':\n{value_counts.to_string(index=False)}\n")

```

Value counts for column 'Diabetes_012':

```

Value  Count
0.0    213703
2.0    35346
1.0     4631

```

Value counts for column 'HighBP':

```

Value  Count
0.0    144851
1.0    108829

```

Value counts for column 'HighChol':

```

Value  Count
0.0    146089
1.0    107591

```

Value counts for column 'CholCheck':

```

Value  Count
1.0    244210
0.0     6470

```

```

#Column : Diabetes_012:
    0 is for no diabetes
    1 is for pre-diabetes or borderline diabetes
    2 is for yes diabetes
#Column: HighBP:
    0 : It represents No high blood pressure
    1 : It represents high blood pressure
#Column: HighChol:
    0 : It represents No High Cholesterol
    1 : It represents high Cholesterol
#Column: CholCheck :
    0 : Cholesterol not checked in past 5 years
    1 : Cholesterol checked in past 5 years
#Column: Body Mass Index : Number
#Column : Smoker :
    Smoked at least 100 cigarettes in your entire life
    0 : Non - Smoker
    1 : Smoker
#Column: Stroke
    0 : No stroke
    1 : Stroke
#Column : Heart Disease or Attack
    0 : No Heart Disease or Attack
    1 : Heart Disease or Attack is there
#Column : PhysActivity :
    Adults who reported doing physical activity or exercise during the past 30 days other than their regular job
    0 : No physical Activity
    1 : Physical Activity
#Column : Fruits: Atleast ate a fruit a day
    0 : No
    1 : Yes
#Column : Veggies : Atleast ate a vegetable a day
    0 : No
    1 : Yes
#Column : HvyAlcoholConsump
    0 : No
    1 : Yes
#Column : 'AnyHealthcare'
    0 : No Health care access
    1 : Yes
#Column : 'NoDocbcCost'
    0 : No
    1 : Yes
#Column : 'GenHlth'
    Range 1 - 5(1 - Excellent, 5 - poor)
#Column : 'MentHlth'
    Range 0 - 30(0- No bad mental health days and then the range increasing if mental health is affected)
#Column : 'PhysHlth'
    Range 0 - 30
#Column : 'DiffWalk'
    0 - No difficulty in walking
    1 - Difficulty in Walking
#Column : 'Sex'
    0 - Female
    1 - Male
#Column : 'Age'
    1 is 18-24 all the way up to 13 is 80 and older. 5 year increments.
#Column : 'Education'
    with 1 being never attended school or kindergarten only up to 6 being college 4 years or more
    Scale here is 1-6
#Column : 'Income'
    with 1 being less than $10,000 all the way up to 8 being $75,000 or more

```

- Pre-processing the data by copying the original dataframe into another dataframe for exploring, the reason for this is to conveniently change the columns and their names and the categorical values encodings.

Data Preprocessing :

```
: import warnings
warnings.filterwarnings("ignore")
data2 = data.copy()
data2.Age[data2['Age'] == 1] = '18 to 24'
data2.Age[data2['Age'] == 2] = '25 to 29'
data2.Age[data2['Age'] == 3] = '30 to 34'
data2.Age[data2['Age'] == 4] = '35 to 39'
data2.Age[data2['Age'] == 5] = '40 to 44'
data2.Age[data2['Age'] == 6] = '45 to 49'
data2.Age[data2['Age'] == 7] = '50 to 54'
data2.Age[data2['Age'] == 8] = '55 to 59'
data2.Age[data2['Age'] == 9] = '60 to 64'
data2.Age[data2['Age'] == 10] = '65 to 69'
data2.Age[data2['Age'] == 11] = '70 to 74'
data2.Age[data2['Age'] == 12] = '75 to 79'
data2.Age[data2['Age'] == 13] = '80 or older'

data2.Diabetes_012[data2['Diabetes_012'] == 0] = 'No Diabetes'
data2.Diabetes_012[data2['Diabetes_012'] == 1] = 'Pre-Diabetes'
data2.Diabetes_012[data2['Diabetes_012'] == 2] = 'Diabetes'

data2.HighBP[data2['HighBP'] == 0] = 'No High'
data2.HighBP[data2['HighBP'] == 1] = 'High BP'

data2.HighChol[data2['HighChol'] == 0] = 'No High Cholesterol'
data2.HighChol[data2['HighChol'] == 1] = 'High Cholesterol'

data2.CholCheck[data2['CholCheck'] == 0] = 'No Cholesterol Check in 5 Years'
data2.CholCheck[data2['CholCheck'] == 1] = 'Cholesterol Check in 5 Years'

data2.Smoker[data2['Smoker'] == 0] = 'No'
data2.Smoker[data2['Smoker'] == 1] = 'Yes'

data2.Stroke[data2['Stroke'] == 0] = 'No'
data2.Stroke[data2['Stroke'] == 1] = 'Yes'

data2.HeartDiseaseorAttack[data2['HeartDiseaseorAttack'] == 0] = 'No'
data2.HeartDiseaseorAttack[data2['HeartDiseaseorAttack'] == 1] = 'Yes'

data2.PhysActivity[data2['PhysActivity'] == 0] = 'No'
data2.PhysActivity[data2['PhysActivity'] == 1] = 'Yes'

data2.Fruits[data2['Fruits'] == 0] = 'No'
data2.Fruits[data2['Fruits'] == 1] = 'Yes'

data2.Veggies[data2['Veggies'] == 0] = 'No'
data2.Veggies[data2['Veggies'] == 1] = 'Yes'

data2.HvyAlcoholConsump[data2['HvyAlcoholConsump'] == 0] = 'No'
data2.HvyAlcoholConsump[data2['HvyAlcoholConsump'] == 1] = 'Yes'

data2.AnyHealthcare[data2['AnyHealthcare'] == 0] = 'No'
data2.AnyHealthcare[data2['AnyHealthcare'] == 1] = 'Yes'

data2.NoDocbcCost[data2['NoDocbcCost'] == 0] = 'No'
data2.NoDocbcCost[data2['NoDocbcCost'] == 1] = 'Yes'

data2.GenHlth[data2['GenHlth'] == 5] = 'Excellent'
data2.GenHlth[data2['GenHlth'] == 4] = 'Very Good'
data2.GenHlth[data2['GenHlth'] == 3] = 'Good'
data2.GenHlth[data2['GenHlth'] == 2] = 'Fair'
data2.GenHlth[data2['GenHlth'] == 1] = 'Poor'

data2.DiffWalk[data2['DiffWalk'] == 0] = 'No'
data2.DiffWalk[data2['DiffWalk'] == 1] = 'Yes'

data2.Sex[data2['Sex'] == 0] = 'Female'
data2.Sex[data2['Sex'] == 1] = 'Male'

data2.Education[data2['Education'] == 1] = 'Never Attended School'
data2.Education[data2['Education'] == 2] = 'Elementary'
data2.Education[data2['Education'] == 3] = 'Junior High School'
data2.Education[data2['Education'] == 4] = 'Senior High School'
data2.Education[data2['Education'] == 5] = 'Undergraduate Degree'
data2.Education[data2['Education'] == 6] = 'Magister'

data2.Income[data2['Income'] == 1] = 'Less Than $10,000'
data2.Income[data2['Income'] == 2] = 'Less Than $10,000'
data2.Income[data2['Income'] == 3] = 'Less Than $10,000'
data2.Income[data2['Income'] == 4] = 'Less Than $10,000'
data2.Income[data2['Income'] == 5] = 'Less Than $35,000'
data2.Income[data2['Income'] == 6] = 'Less Than $35,000'
data2.Income[data2['Income'] == 7] = 'Less Than $35,000'
data2.Income[data2['Income'] == 8] = '$75,000 or More'
```


data2.head()

	Diabetes_012	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth
0	No Diabetes	High BP	High Cholesterol	Cholesterol Check in 5 Years	40.0	Yes	No	No	No	No	...	Yes	No	Excellent
1	No Diabetes	No High	No High Cholesterol	No Cholesterol Check in 5 Years	25.0	Yes	No	No	Yes	No	...	No	Yes	Good
2	No Diabetes	High BP	High Cholesterol	Cholesterol Check in 5 Years	28.0	No	No	No	No	Yes	...	Yes	Yes	Excellent
3	No Diabetes	High BP	No High Cholesterol	Cholesterol Check in 5 Years	27.0	No	No	No	Yes	Yes	...	Yes	No	Fair
4	No Diabetes	High BP	High Cholesterol	Cholesterol Check in 5 Years	24.0	No	No	No	Yes	Yes	...	Yes	No	Fair

5 rows × 22 columns

Checking Null Values and Handling it:

```

null_counts = data.isnull().sum()
total_counts = data.count()
dict_1 = {'Total Count' : total_counts, "Null Count" : null_counts}
null_table = pd.DataFrame(dict_1)
null_table.index.name = "Column Names"
null_table

```

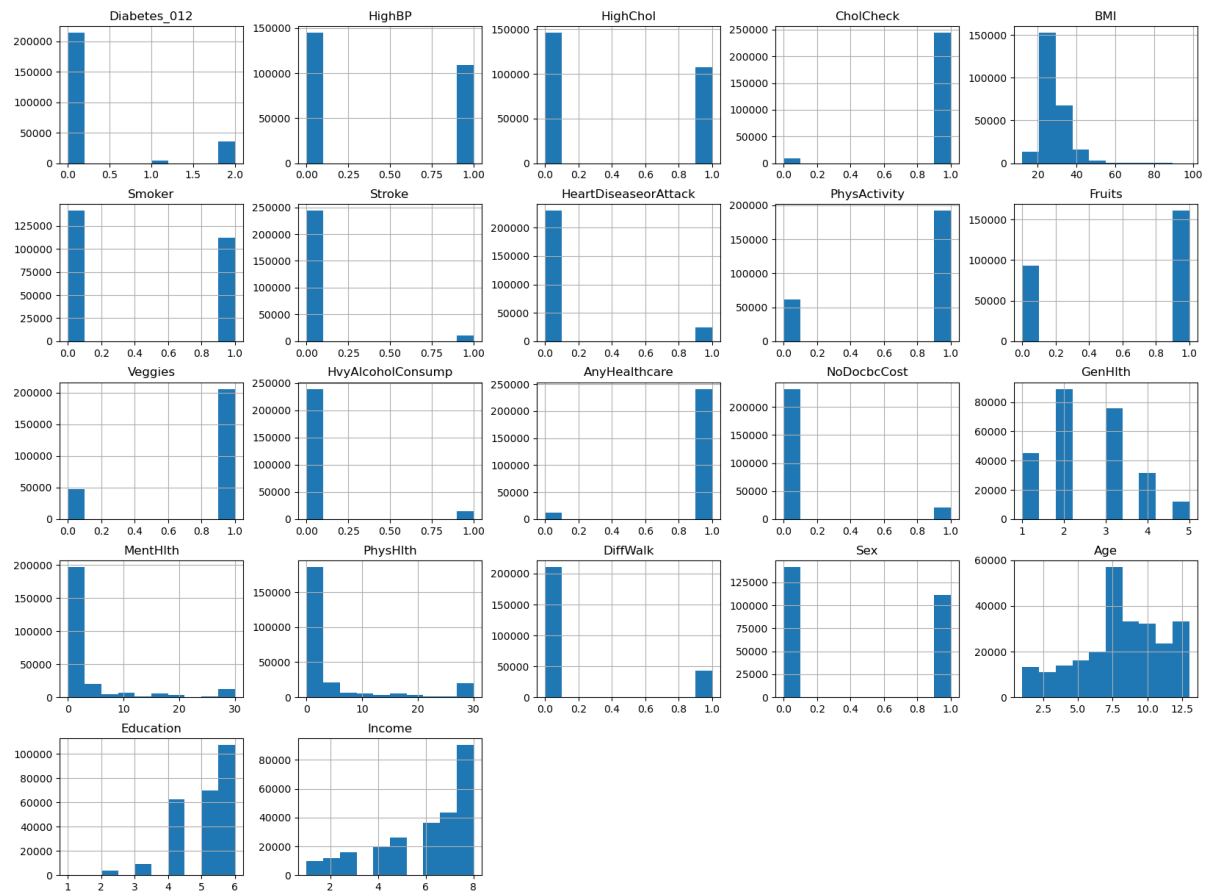
	Total Count	Null Count
Column Names		
Diabetes_012	253680	0
HighBP	253680	0
HighChol	253680	0
CholCheck	253680	0
BMI	253680	0
Smoker	253680	0
Stroke	253680	0
HeartDiseaseorAttack	253680	0
PhysActivity	253680	0
Fruits	253680	0
Veggies	253680	0
HvyAlcoholConsump	253680	0
AnyHealthcare	253680	0
NoDocbcCost	253680	0
GenHlth	253680	0
MentHlth	253680	0
PhysHlth	253680	0
DiffWalk	253680	0
Sex	253680	0
Age	253680	0
Education	253680	0
Income	253680	0

No Missing Values in the dataset, so no null value handling measures is done here.

➤ Data Visualization

Histogram:

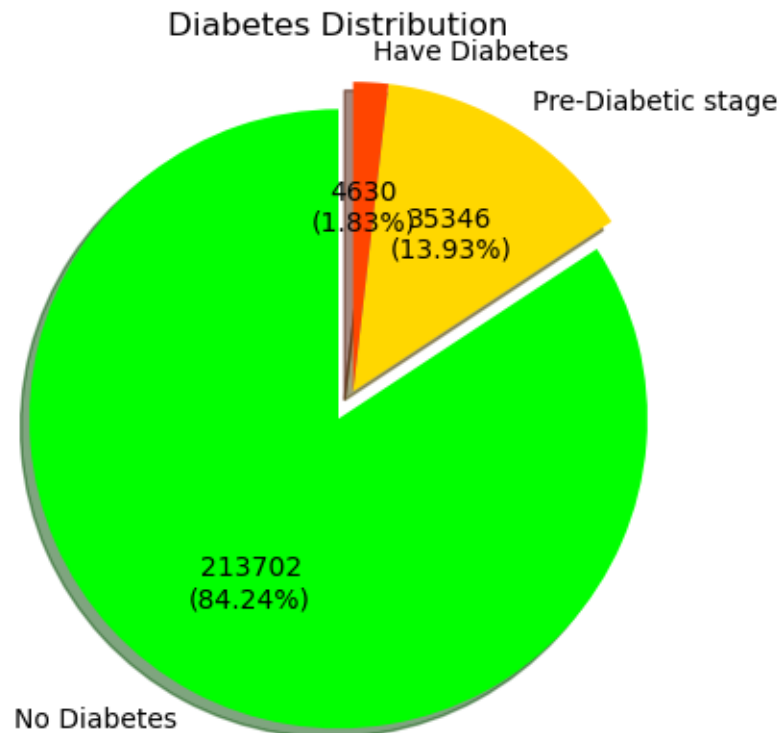
```
data.hist(figsize=(20,15))
```



Inference: The Histogram for all the dataset is done to explore the basic categorical values distributed in the dataset. The frequency of each column(attribute) is plotted and is help in the exploring the count of values

Pie Chart:

```
labels = ["No Diabetes", "Pre-Diabetic stage", "Have Diabetes"]
sizes = data["Diabetes_012"].value_counts()
colors = ['#00FF00', '#FFD700', '#FF4500']
explode = (0.1, 0, 0)
fig, ax = plt.subplots()
count_perc = lambda pct: f'{int(pct * sum(sizes) / 100)}\n({pct:.2f}%)'
ax.pie(sizes, labels=labels, colors=colors, explode=explode, autopct=count_perc,
      shadow=True, startangle=90)
ax.axis('equal')
ax.set_title('Diabetes Distribution')
plt.show()
```

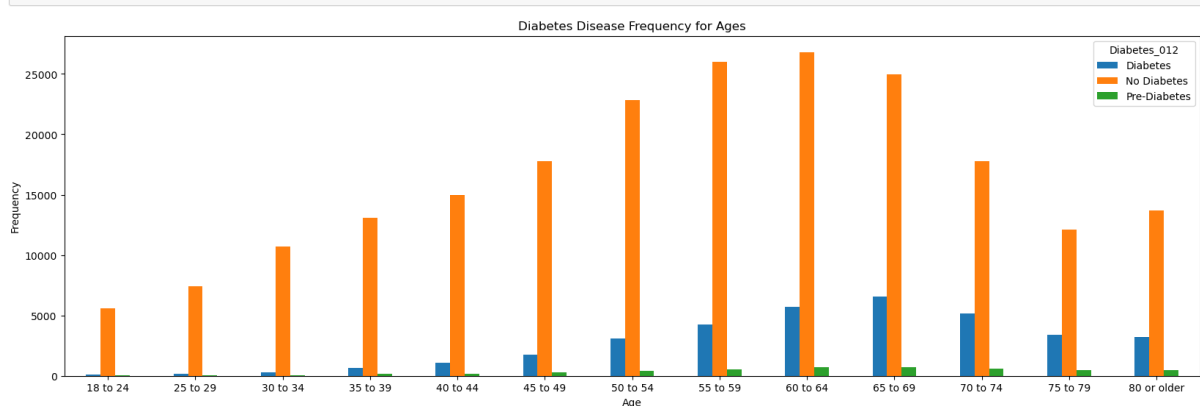


Inference:

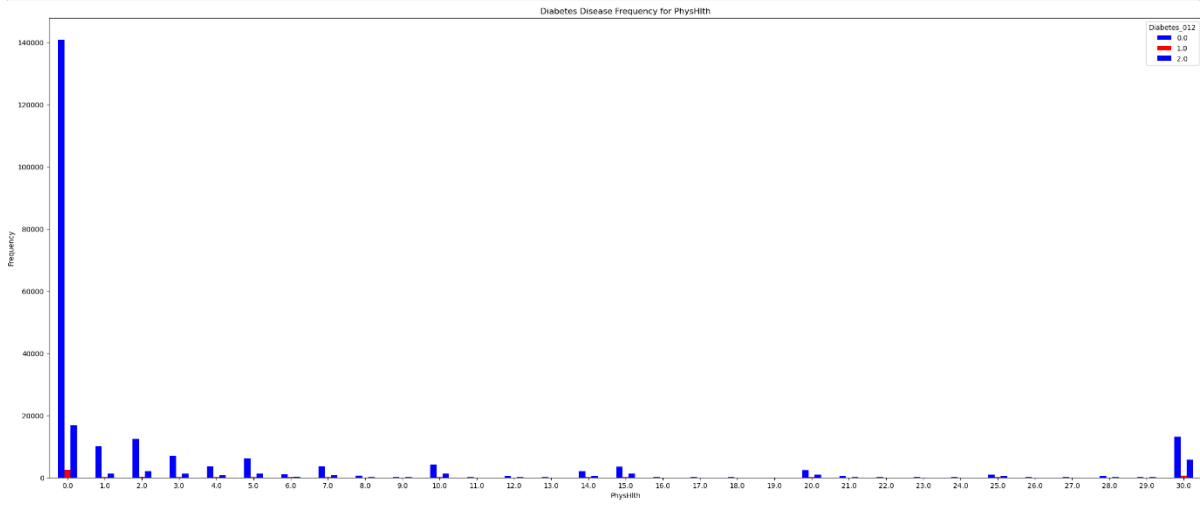
The pie-chart for the target column "Diabetes_012" to know how many persons are with Diabetes, Pre-Diabetes and Doesn't have diabetes and the plot visualise the value counts and the percentage of class split is visually depicted, as we can see the dataset contains around 84% of the persons with no diabetes and the diabetes count is significantly very low which will significantly affect the model's performance. And the inference from this plot is that out of 10 people, one person is having diabetes

Bivariate Analysis:

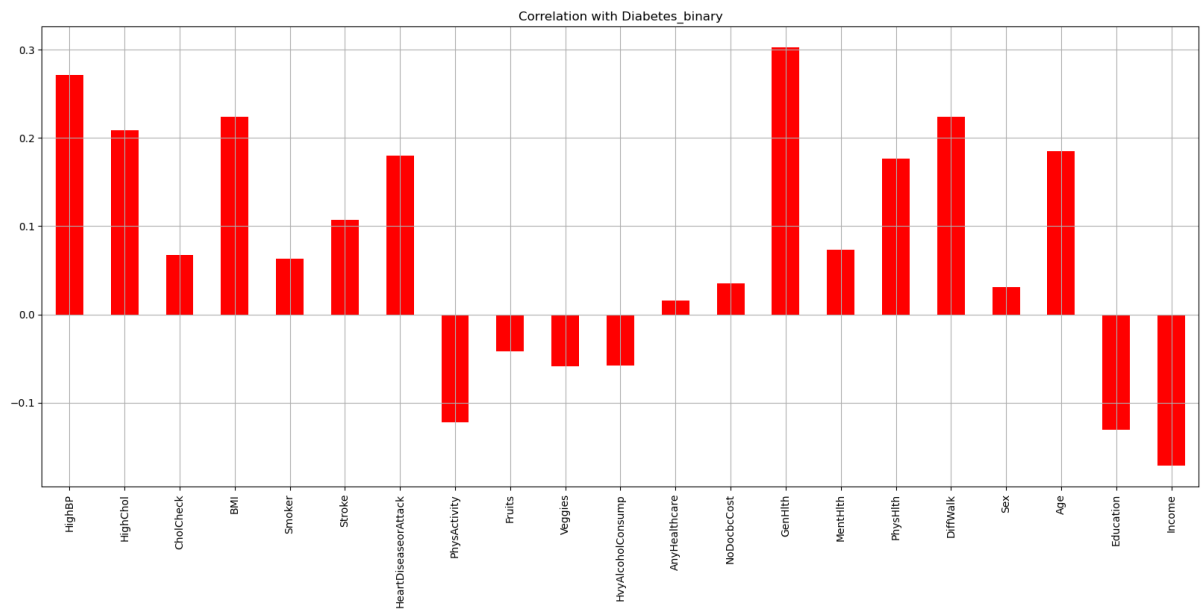
```
pd.crosstab(data2.Age,data2.Diabetes_012).plot(kind="bar",figsize=(20,6))
plt.title('Diabetes Disease Frequency for Ages')
plt.xlabel('Age')
plt.xticks(rotation=0)
plt.ylabel('Frequency')
plt.show()
```



```
pd.crosstab(data.PhysHlth,data.Diabetes_012).plot(kind="bar",figsize=(30,12),color=['Blue', 'Red' ])
plt.title('Diabetes Disease Frequency for PhysHlth')
plt.xlabel('PhysHlth')
plt.xticks(rotation=0)
plt.ylabel('Frequency')
plt.show()
```



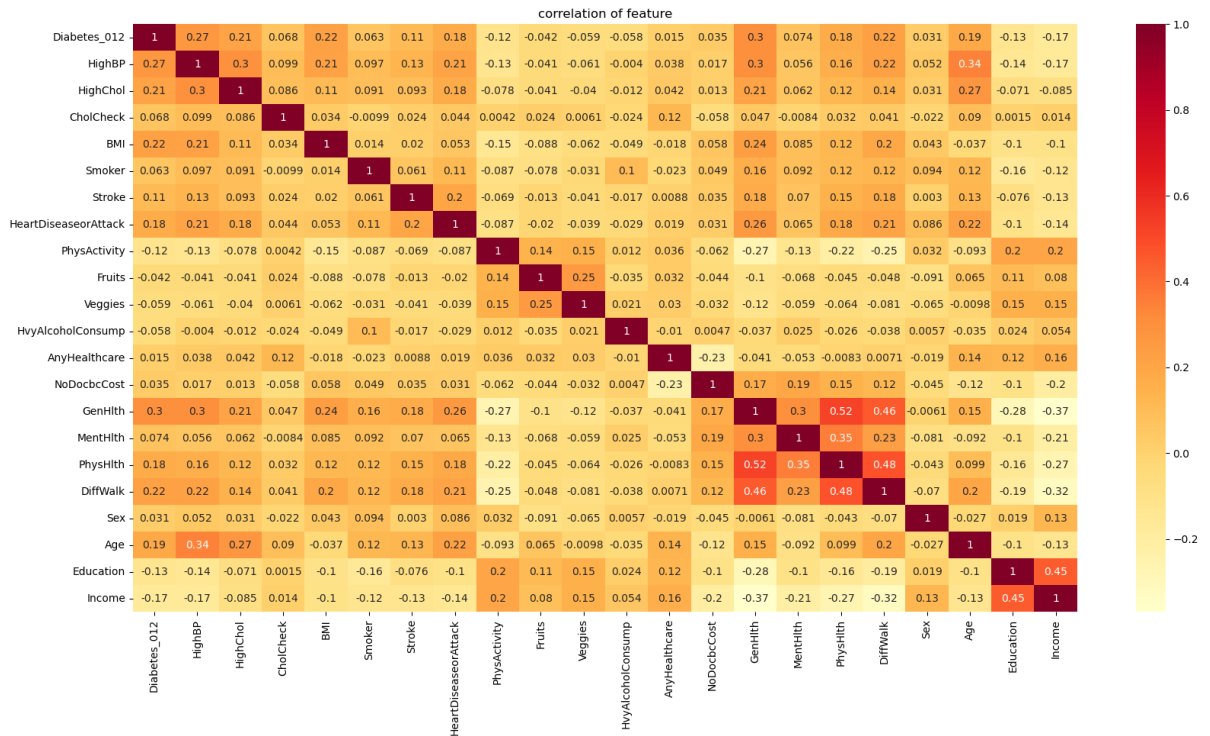
```
data.drop('Diabetes_012', axis=1).corrwith(data.Diabetes_012).plot(kind='bar', grid=True, figsize=(20, 8), title="Correlation with Diabetes_binary",color="red");
```



Multivariate Analysis:

Heat Map:

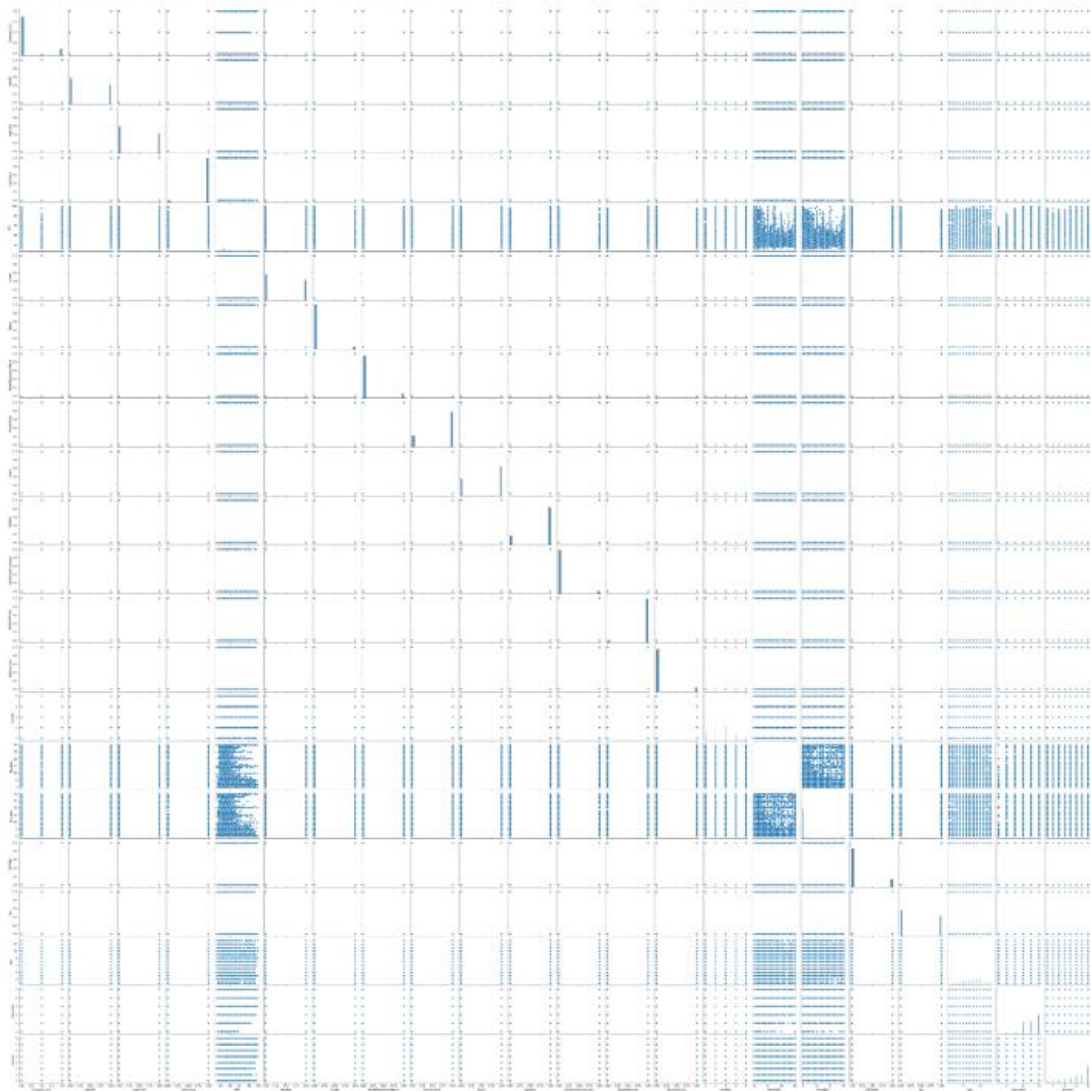
```
plt.figure(figsize = (20,10))
sns.heatmap(data.corr(),annot=True, cmap = 'v1orRd' )
plt.title("correlation of feature")
```



Pair Plot:

```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x1ca0838e730>
```

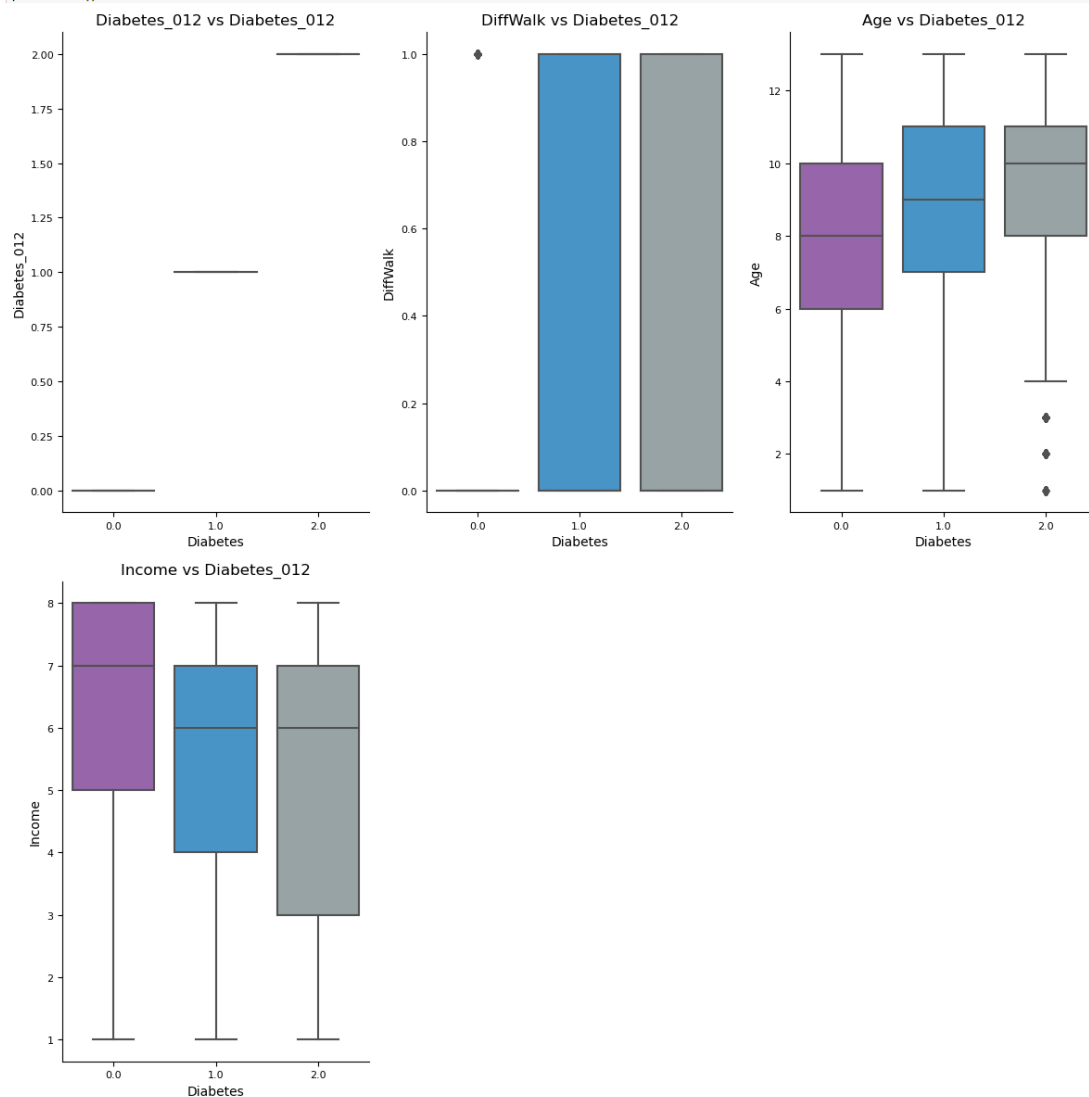


Box Plot:

```
features = ['Diabetes_012', 'DiffWalk', 'Age', 'Income']
subset_data = data[features]
custom_palette = ['#9b59b6', '#3498db', '#95a5a6', '#e74c3c', '#2ecc71', '#f1c40f', '#1abc9c', '#c0392b', '#2980b9', '#f39c12']
sns.set_palette(custom_palette)
num_columns = 3
num_rows = (len(features) + num_columns - 1) // num_columns
fig, axes = plt.subplots(nrows=num_rows, ncols=num_columns, figsize=(12, 6*num_rows))
axes = axes.flatten()
for i, feature in enumerate(features):
    sns.boxplot(x='Diabetes_012', y=feature, data=subset_data, ax=axes[i])
    axes[i].set_title(f'{feature} vs Diabetes_012', fontsize=12)
    axes[i].set_xlabel('Diabetes', fontsize=10)
    axes[i].set_ylabel(feature, fontsize=10)
    axes[i].tick_params(axis='both', labels=8)
    axes[i].spines['top'].set_visible(False)
    axes[i].spines['right'].set_visible(False)
if len(features) % num_columns != 0:
    for j in range(len(features), num_rows * num_columns):
        fig.delaxes(axes[j])

# Adjust the Layout
plt.tight_layout()

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



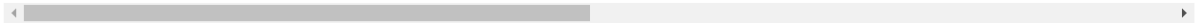
Inference: From the Heatmap, we can see the correlation values and the high values are Age, Income, Difficulty in walking. For that column only plotting the box plot to see the outlier presence, as the outliers are very less and some outliers are that 1 % Diabetic, so no-outlier handling procedures are done.

➤ Performing Statistical Analysis by getting the description of the data

```
data.describe()
```

	Diabetes_012	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	
count	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680.000000	253680
mean	0.296921	0.429001	0.424121	0.962670	28.382364	0.443169	0.040571	0.094186	0.756544	0
std	0.698160	0.494934	0.494210	0.189571	6.608694	0.496761	0.197294	0.292087	0.429169	0
min	0.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000	0
25%	0.000000	0.000000	0.000000	1.000000	24.000000	0.000000	0.000000	0.000000	1.000000	0
50%	0.000000	0.000000	0.000000	1.000000	27.000000	0.000000	0.000000	0.000000	1.000000	1
75%	0.000000	1.000000	1.000000	1.000000	31.000000	1.000000	0.000000	0.000000	1.000000	1
max	2.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	1.000000	1.000000	1

8 rows × 22 columns



➤ Splitting the dataset into two variables to train the data and build a model.

Splitting the dataset into 'x' and 'y' variables (y - Target Variable)

```
X = data.drop("Diabetes_012", axis = 1)
y = data["Diabetes_012"]
```

Splitting the 'x' and 'y' into Training and Test Data

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.7, test_size = 0.3, random_state = 100)
```

➤ Scaling the data using StandardScaler to handle highly varying magnitudes or values or units.

Scaling the data using StandardScaler

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
X_train = scalar.fit_transform(X_train)
X_test = scalar.fit_transform(X_test)
```


- Building the model using various Classifiers.

Logistic Regression:

Logistic regression is a statistical modeling technique used for predicting the probability of an event or the likelihood of an outcome occurring based on a set of input variables. It is primarily used for binary classification problems, where the outcome variable takes on one of two possible values (e.g., true/false, yes/no). The logistic regression model estimates the relationship between the independent variables (also known as predictors or features) and the dependent variable (the binary outcome) by applying a logistic or sigmoid function to a linear combination of the predictor variables. This transformation allows the model to output probabilities ranging from 0 to 1, which can then be used to make predictions or classify new instances.

```
lj = LogisticRegression(solver="liblinear").fit(X_train, y_train)
predict_lj = lj.predict(X_test)
acc_lj = accuracy_score(y_test, predict_lj)
print("The accuracy of logistic regression is : ", acc_lj)
print(classification_report(y_test, predict_lj))
```

```
The accuracy of logistic regression is : 0.8448964574792389
      precision    recall  f1-score   support

    0.0         0.86     0.98     0.91     63926
    1.0         0.00     0.00     0.00      1381
    2.0         0.54     0.17     0.26     10797

 accuracy         0.84         0.84         0.84     76104
 macro avg        0.47        0.38        0.39     76104
 weighted avg     0.80        0.84        0.80     76104
```

Gaussian Naïve Bayes:

Gaussian Naive Bayes is a variant of the Naive Bayes algorithm that assumes the features follow a Gaussian (normal) distribution. It is a probabilistic classifier that uses Bayes' theorem to calculate the posterior probability of a class given the observed features. The "naive" assumption in Naive Bayes refers to the independence assumption, where it assumes that the features are conditionally independent of each other given the class label. Gaussian Naive Bayes models the likelihood of each feature given a class using a Gaussian distribution, allowing it to handle continuous numerical features. It is commonly used for classification tasks and is known for its simplicity, efficiency, and ability to handle high-dimensional data.

Gaussian Naive Bayes

```
gnb = GaussianNB().fit(X_train, y_train)
predict_gnb = gnb.predict(X_test)
acc_gnb = accuracy_score(y_test, predict_gnb)
acc_gnb
print("The accuracy of Gaussian Naive Bayes is : ", acc_gnb)
```

```
The accuracy of Gaussian Naive Bayes is : 0.7584489645747924
```

K-Nearest Neighbour (KNN):

K-Nearest Neighbors (KNN) is a non-parametric machine learning algorithm used for both classification and regression tasks. It works based on the principle that similar instances tend to have similar labels or target values. In KNN, the algorithm stores the entire training dataset in memory, and when a new instance needs to be classified or predicted, it identifies the k nearest neighbors (the k closest instances) in the feature space. The class or value of the new instance is then determined by majority voting (for classification) or averaging (for regression) among its k nearest neighbors. The value of k, representing the number of neighbors, is a hyperparameter that needs to be predefined.

KNN

```
knnc = KNeighborsClassifier(n_neighbors = 3).fit(X_train, y_train)
predict_knnc = knnc.predict(X_test)
acc_knnc = accuracy_score(y_test, predict_knnc)
print("The accuracy of KNN Classifier is : ", acc_knnc)
```

The accuracy of KNN Classifier is : 0.8202328392725744

Decision Tree (CART):

A Decision Tree, specifically the Classification and Regression Tree (CART) algorithm, is a popular supervised machine learning technique used for both classification and regression tasks. It builds a tree-like model that makes predictions based on a series of binary decisions or conditions applied to the input features. The CART algorithm recursively partitions the feature space by selecting the best split based on a chosen criterion, such as Gini impurity for classification or mean squared error for regression. The goal is to create homogeneous subgroups within each branch of the tree that minimize the impurity or error. The resulting tree can be used to make predictions by traversing the branches from the root to a leaf node, where the predicted class or value is assigned. Decision Trees offer interpretability, handle both categorical and numerical features, and can capture non-linear relationships in the data.

Decision Tree (CART)

```
cartc = DecisionTreeClassifier(random_state=42).fit(X_train, y_train)
predict_cartc = cartc.predict(X_test)
acc_cartc = accuracy_score(y_test, predict_cartc)
print("The accuracy of Decision Tree Classifier is : ", acc_cartc)
```

The accuracy of Decision Tree Classifier is : 0.7653342794071271

Random Forest:

Random Forest, each decision tree is trained on a random subset of the training data and a random subset of features. This random sampling and feature selection introduce diversity among the trees. During prediction, each tree in the forest independently produces an outcome, and the final prediction is determined by combining the individual predictions through voting (for classification).

Random Forest

```
rfc = RandomForestClassifier(random_state=42,verbose=False).fit(X_train, y_train)
predict_rfc = rfc.predict(X_test)
acc_rfc = accuracy_score(y_test, predict_rfc)
print("The accuracy of Random Forest Classifier is : ", acc_rfc)
```

The accuracy of Random Forest Classifier is : 0.8405997056659308

Gradient Boosting:

Gradient Boosting is a powerful machine learning technique used for both regression and classification tasks. It is an ensemble method that combines multiple weak predictive models, typically decision trees, to create a strong predictive model.

Gradient Boosting

```
gbmc = GradientBoostingClassifier(verbose=False).fit(X_train, y_train)
predict_gbmc = gbmc.predict(X_test)
acc_gbmc = accuracy_score(y_test, predict_gbmc)
print("The accuracy of Gradient Boosting is : ", acc_gbmc)
```

The accuracy of Gradient Boosting is : 0.8484310942920215

XG Boosting:

XGBoost (Extreme Gradient Boosting) is an optimized and highly efficient implementation of the Gradient Boosting framework. It is a popular machine learning algorithm used for regression, classification, and ranking tasks. XGBoost improves upon traditional gradient boosting methods by introducing several enhancements and optimization techniques.

XG Boosting

```
xgbc = XGBClassifier().fit(X_train, y_train)
predict_xgbc = xgbc.predict(X_test)
acc_xgbc = accuracy_score(y_test, predict_xgbc)
print("The accuracy of XG Boost is ", acc_xgbc)
```

The accuracy of XG Boost is 0.8479317775675391

Light Gradient Boosting:

Light Gradient Boosting (LightGBM) is a gradient boosting framework that is designed to be lightweight and efficient. It is an optimized implementation of the gradient boosting algorithm that aims to provide faster training speed and lower memory usage while maintaining high prediction accuracy.

Light Gradient Boosting

```
lgbmc = LGBMClassifier().fit(X_train, y_train)
predict_lgbmc = lgbmc.predict(X_test)
acc_lgbmc = accuracy_score(y_test, predict_lgbmc)
print("The accuracy of Light Gradient Boosting is ", acc_lgbmc)
```

The accuracy of Light Gradient Boosting is 0.8484310942920215

- Exporting the Logistic Regression model to pickle file as it gives the highest accuracy among all the model deployed.

Exporting the Logistic Regression model to Pickle File

```
import pickle
pickle.dump(lj, open("diabetes_lj_model.pkl", "wb"))
```

HTML Code:

Index:

<!DOCTYPE html>

<html>

<head>

<style>

body {

background-color: #f2f2f2;

font-family: "Times New Roman", Times, serif;

}

h1 {

text-align: center;

margin-top: 0;

margin-bottom: 20px;

color: #333333;

}

.container {

```
max-width: 400px;
margin: 0 auto;
padding: 80px;
background-color: #E5F1F9;
border-radius: 10px;
box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

```
.form-row {
  margin-bottom: 10px;
}
```

```
.form-row label {
  display: block;
  font-weight: bold;
  margin-bottom: 5px;
}
```

```
div {
  width: 90%;
  padding: 20px;
  border: none;
  border-radius: 10px;
  background-color: #f8f8f8;
  box-shadow: inset 0 1px 2px rgba(0, 0, 0, 0.1);
}
```

```
.form-row input[type="radio"] {
  margin-right: 10px;
}
```

```
.form-row input[type="submit"] {  
  padding: 10px 20px;  
  background-color: #4CAF50;  
  color: #fff;  
  cursor: pointer;  
  transition: background-color 0.3s ease;  
}
```

```
.form-row input[type="submit"]:hover {  
  background-color: #45a049;  
}
```

```
.submit-btn {  
  text-align: center;  
}
```

```
.range-value {  
  font-weight: bold;  
  margin-left: 10px;  
}
```

```
.required-label::after {  
  content: "*";  
  color: red;  
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<div class="container">
```

<h1>Diabetes Prediction</h1>

<h4>This page works based on a trained model to predict that the person is having diabetes or not.</h4>

<h4>We would like to know some details about your health.</h4>

<form action="/predict" method="post">

<div class="form-row">

<label for="Name" class="required-label">Tell us your name</label>

<input type="text" name="Name" required>

</div>

<div class="form-row">

<label for="HighBp" class="required-label">Do you have high Blood Pressure[Bp of 140/80 and above] ?</label>

<input type="radio" name="HighBp" value="yes" required>Yes

<input type="radio" name="HighBp" value="no" required>No

</div>

<div class="form-row">

<label for="HighChol" class="required-label">Do you have high Cholesterol[240mg/dL and above] ?</label>

<input type="radio" name="HighChol" value="yes" required>Yes

<input type="radio" name="HighChol" value="no" required>No

</div>

<div class="form-row">

<label for="CholCheck" class="required-label">Have you checked your cholesterol in the past five years ?</label>

<input type="radio" name="CholCheck" value="yes" required>Yes

<input type="radio" name="CholCheck" value="no" required>No

</div>

<div class="form-row">

<label for="BMI" class="required-label">What is your Body Mass Index(BMI) value ?</label>

<input type="number" name="BMI" placeholder = "Enter BMI value here" required>

</div>

```
<div class="form-row">

  <label for="Smoker" class="required-label">Have you smoked(a minimum of 100
cigars) ?</label>

  <input type="radio" name="Smoker" value="yes" required>Yes

  <input type="radio" name="Smoker" value="no" required>No

</div>

<div class="form-row">

  <label for="Stroke" class="required-label">Do you have any history of getting stroke
?</label>

  <input type="radio" name="Stroke" value="yes" required>Yes

  <input type="radio" name="Stroke" value="no" required>No

</div>

<div class="form-row">

  <label for="HeartDiseaseorAttack" class="required-label">Do you have any Heart
diseases or Heart Attack ?</label>

  <input type="radio" name="HeartDiseaseorAttack" value="yes" required>Yes

  <input type="radio" name="HeartDiseaseorAttack" value="no" required>No

</div>

<div class="form-row">

  <label for="PhysActivity" class="required-label">Do you workout(exercise) daily or
weekly three times ?</label>

  <input type="radio" name="PhysActivity" value="yes" required>Yes

  <input type="radio" name="PhysActivity" value="no" required>No

</div>

<div class="form-row">

  <label for="Fruits" class="required-label">Do you eat fruits daily(atleast 1 per day)
?</label>

  <input type="radio" name="Fruits" value="yes" required>Yes

  <input type="radio" name="Fruits" value="no" required>No

</div>

<div class="form-row">
```


<label for="Veggies" class="required-label">Do you eat vegetables daily(atleast 1 per day) ?</label>

<input type="radio" name="Veggies" value="yes" required>Yes

<input type="radio" name="Veggies" value="no" required>No

</div>

<div class="form-row">

<label for="HvyAlcoholConsump" class="required-label">Are you a heavy alcohol consumer ?(For women, heavy drinking is 8 drinks or more per week. For men, heavy drinking is 15 drinks or more per week)</label>

<input type="radio" name="HvyAlcoholConsump" value="yes" required>Yes

<input type="radio" name="HvyAlcoholConsump" value="no" required>No

</div>

<div class="form-row">

<label for="AnyHealthcare" class="required-label">Do you enough health care access(like insurances and coverages) ?</label>

<input type="radio" name="AnyHealthcare" value="yes" required>Yes

<input type="radio" name="AnyHealthcare" value="no" required>No

</div>

<div class="form-row">

<label for="NoDocbcCost" class="required-label"> Was there a time in the past 12 months when you needed to see a doctor but could not because of cost?</label>

<input type="radio" name="NoDocbcCost" value="yes" required>Yes

<input type="radio" name="NoDocbcCost" value="no" required>No

</div>

<div class="form-row">

<label for="GenHlth" class="required-label">Would you say that in general your health ?
(On a scale of 1-5: 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor)</label>

<input type="range" name="GenHlth" min="1" max="5" required
oninput="updateRangeValue(this)">

Value:

</div>

<div class="form-row">


```
<label for="Education" class="required-label">Tell about your education level  
?<br>with 1 being never attended school or kindergarten only up to 6 being college 4 years  
or more(Scale here is 1-6)</label>
```

```
<input type="range" name="Education" min="1" max="6" required  
oninput="updateRangeValue(this)">
```

```
<span class="range-value">Value: <span></span></span>
```

```
</div>
```

```
<div class="form-row">
```

```
<label for="Income" class="required-label">Annual Household income from all sources  
?<br> On a scale of 1 to 8(with 1 = $10000 and 8 being $80000 and more, with $10000  
increment)</label>
```

```
<input type="range" name="Income" min="1" max="8" required  
oninput="updateRangeValue(this)">
```

```
<span class="range-value">Value: <span></span></span>
```

```
</div>
```

```
<div class="submit-btn">
```

```
<input type="submit" value="Submit">
```

```
</div>
```

```
</form>
```

```
</div>
```

```
<script>
```

```
function updateRangeValue(rangeInput) {  
    rangeInput.nextElementSibling.firstElementChild.textContent = rangeInput.value;  
}
```

```
</script>
```

```
</body>
```

```
</html>
```

Result

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>

.container {
  max-width: 400px;
  margin: 0 auto;
  padding: 20px;
  background-color: #f4f4f4;
  border: 1px solid #ccc;
  border-radius: 5px;
  animation: fade-in 1s ease-in-out;
}

@keyframes fade-in {
  0% { opacity: 0; transform: translateY(10px); }
  100% { opacity: 1; transform: translateY(0); }
}

h2 {
  text-align: center;
  color: #333;
  animation: scale-up 0.8s ease-in-out;
}

@keyframes scale-up {
  0% { transform: scale(0.9); }
  100% { transform: scale(1); }
}

p {
  margin-bottom: 10px;
  animation: slide-up 0.8s ease-in-out;
```

```
}
```

```
@keyframes slide-up {  
  0% { opacity: 0; transform: translateY(10px); }  
  100% { opacity: 1; transform: translateY(0); }  
}
```

```
ul {  
  padding-left: 20px;  
  animation: slide-up 0.8s ease-in-out;  
}
```

```
.tip-list li {  
  list-style-type: disc;  
  margin-bottom: 5px;  
}
```

```
.message {  
  font-weight: bold;  
  margin-bottom: 15px;  
  animation: slide-up 0.8s ease-in-out;  
}
```

```
.message.no-diabetes {  
  color: #006400;  
}
```

```
.message.pre-diabetic {  
  color: #ff9900;  
}
```

```
.message.diabetes {
    color: #cc0000;
}

</style>
</head>
<body>
    <div class="container">
        <h2>Diabetes Prediction Result</h2>
        {% if prediction == 0 %}
            <p class="message no-diabetes">Hi! {{ name }}, the predicted outcome is that you don't
            have diabetes.</p>
            <p>Here are some tips to avoid diabetes:</p>
            <ul class="tip-list">
                <li>Adopt a healthy diet with fruits and vegetables.</li>
                <li>Engage in regular physical activity.</li>
                <li>Maintain a healthy weight.</li>
                <li>Get regular check-ups with a healthcare professional.</li>
            </ul>
            {% elif prediction == 1 %}
                <p class="message pre-diabetic">Hi! {{ name }}, the predicted outcome is that you are in
                a pre-diabetic stage.</p>
                <p>Here are some steps to prevent diabetes:</p>
                <ul class="tip-list">
                    <li>Follow a balanced diet.</li>
                    <li>Increase physical activity.</li>
                    <li>Maintain a healthy weight.</li>
                    <li>Monitor blood sugar levels regularly.</li>
                </ul>
                <p>You may also consult with healthcare professionals for personalized guidance.</p>
            {% elif prediction == 2 %}
```

```
<p class="message diabetes">Hi! {{ name }}, the predicted outcome is that you have diabetes.</p>
```

```
<p>Here are some tips for managing diabetes:</p>
```

```
<ul class="tip-list">
```

```
<li>Follow a healthy meal plan.</li>
```

```
<li>Monitor blood sugar levels regularly.</li>
```

```
<li>Take prescribed medications as directed.</li>
```

```
<li>Engage in regular exercise.</li>
```

```
<li>Visit healthcare professionals for regular check-ups and guidance.</li>
```

```
</ul>
```

```
<p>You may also consult with diabetes specialists for personalized care.</p>
```

```
{ % endif % }
```

```
</div>
```

```
</body>
```

```
</html>
```

Flask Code:

```
from flask import Flask, render_template, request
```

```
import pandas as pd
```

```
import pickle
```

```
app = Flask(__name__)
```

```
# Load the model from the pickle file
```

```
with open('diabetes_lj_model.pkl', 'rb') as file:
```

```
    model = pickle.load(file)
```

```
# Flask route for the home page
```

```
@app.route('/')
```

```
def home():
```

```

return render_template('index.html')

# Flask route for form submission
@app.route('/predict', methods=['POST'])
def predict():
    # Get user input from the form
    name = request.form['Name']
    input_data = {
        'HighBp': 1 if request.form['HighBp'] == 'yes' else 0,
        'HighChol': 1 if request.form['HighChol'] == 'yes' else 0,
        'CholCheck': 1 if request.form['CholCheck'] == 'yes' else 0,
        'BMI': float(request.form['BMI']),
        'Smoker': 1 if request.form['Smoker'] == 'yes' else 0,
        'Stroke': 1 if request.form['Stroke'] == 'yes' else 0,
        'HeartDiseaseorAttack': 1 if request.form['HeartDiseaseorAttack'] == 'yes' else 0,
        'PhysActivity': 1 if request.form['PhysActivity'] == 'yes' else 0,
        'Fruits': 1 if request.form['Fruits'] == 'yes' else 0,
        'Veggies': 1 if request.form['Veggies'] == 'yes' else 0,
        'HvyAlcoholConsump': 1 if request.form['HvyAlcoholConsump'] == 'yes' else 0,
        'AnyHealthcare': 1 if request.form['AnyHealthcare'] == 'yes' else 0,
        'NoDocbcCost': 1 if request.form['NoDocbcCost'] == 'yes' else 0,
        'GenHlth': int(request.form['GenHlth']),
        'MentHlth': int(request.form['MentHlth']),
        'PhysHlth': int(request.form['PhysHlth']),
        'DiffWalk': 1 if request.form['DiffWalk'] == 'yes' else 0,
        'Sex': int(request.form['Sex']),
        'Age': int(request.form['Age']),
        'Education': int(request.form['Education']),
        'Income': int(request.form['Income'])
    }

```



```

print("Incoming form response:")

for key, value in input_data.items():
    print(key + ": " + str(value))

# Create a dataframe from the user input
input_df = pd.DataFrame(input_data, index=[0])

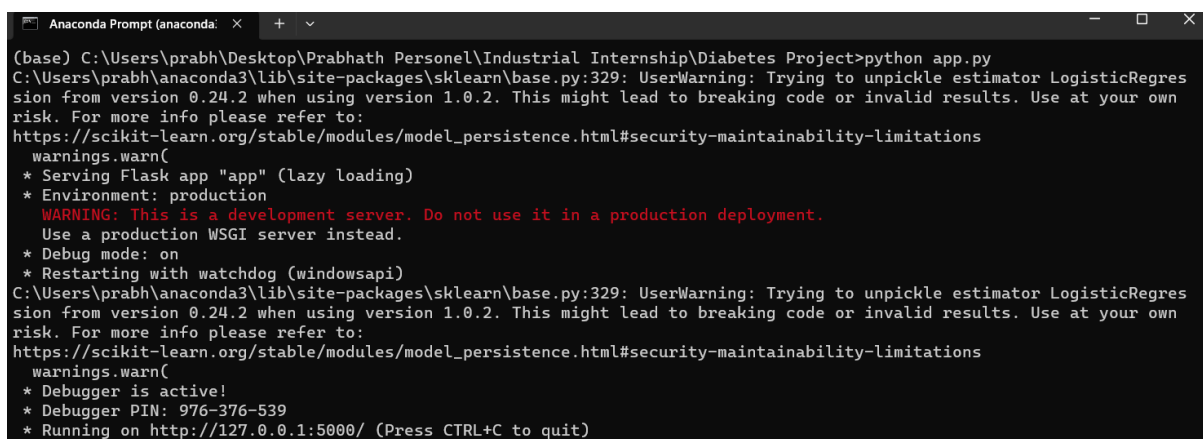
# Make a prediction using the loaded model
prediction = model.predict(input_df)[0]

# Return the prediction as a response
return render_template('result.html', prediction=prediction, name = name)

if __name__ == '__main__':
    app.run(debug=True)

```

Executing app.py

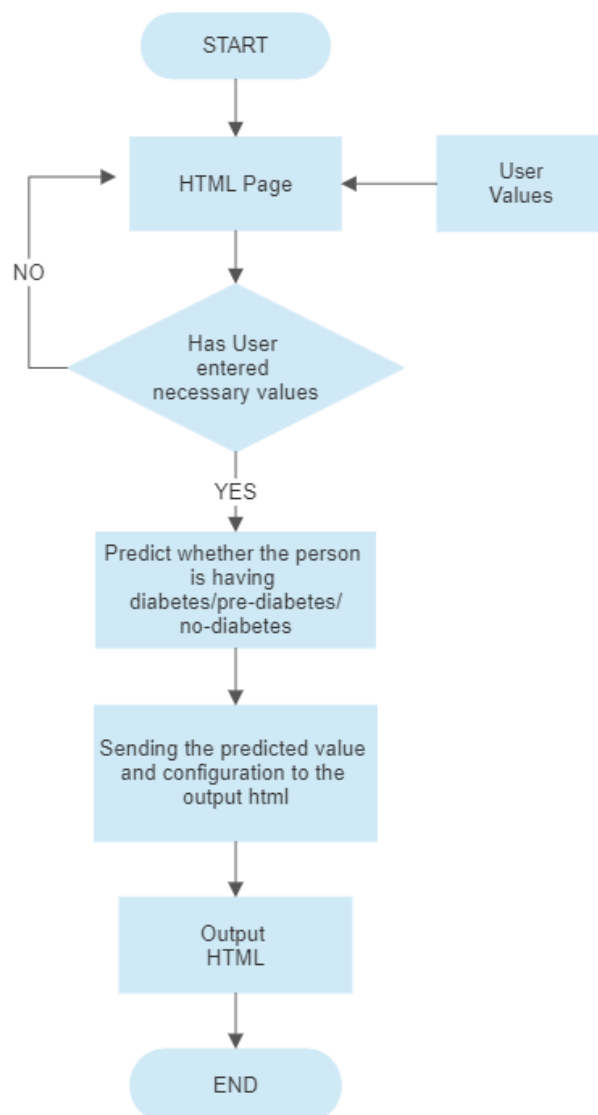


```

Anaconda Prompt (anaconda: x + v
(base) C:\Users\prabh\Desktop\Prabhath Personel\Industrial Internship\Diabetes Project>python app.py
C:\Users\prabh\anaconda3\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 0.24.2 when using version 1.0.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
  warnings.warn(
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with watchdog (windowsapi)
C:\Users\prabh\anaconda3\lib\site-packages\sklearn\base.py:329: UserWarning: Trying to unpickle estimator LogisticRegression from version 0.24.2 when using version 1.0.2. This might lead to breaking code or invalid results. Use at your own risk. For more info please refer to: https://scikit-learn.org/stable/modules/model_persistence.html#security-maintainability-limitations
  warnings.warn(
* Debugger is active!
* Debugger PIN: 976-376-539
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

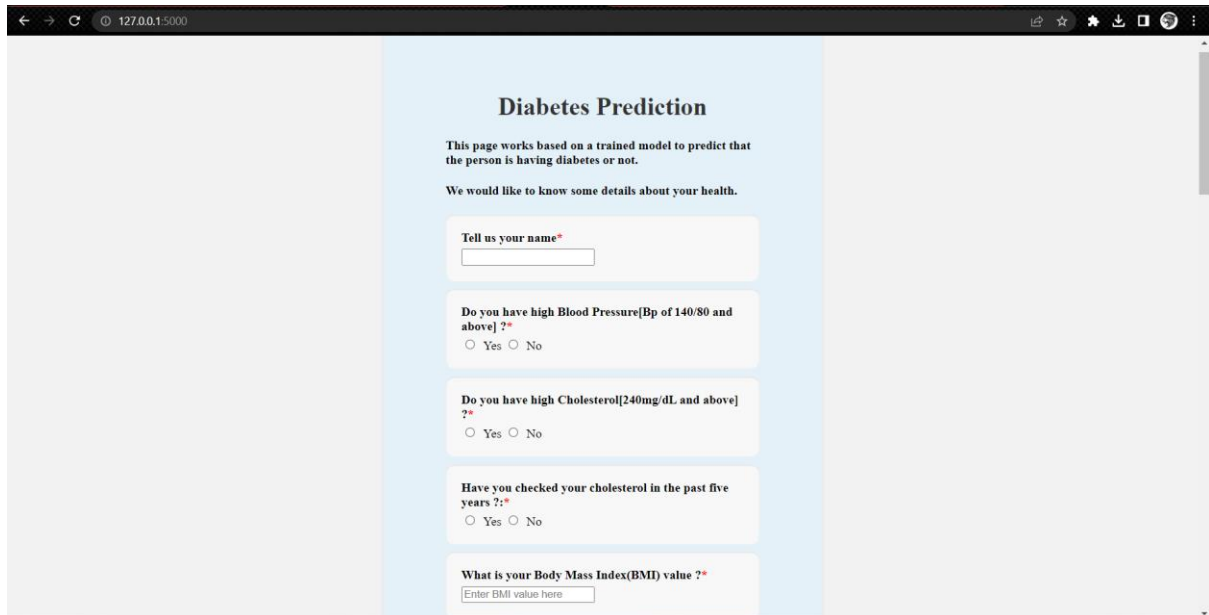
```

Flow Chart of the Solution:



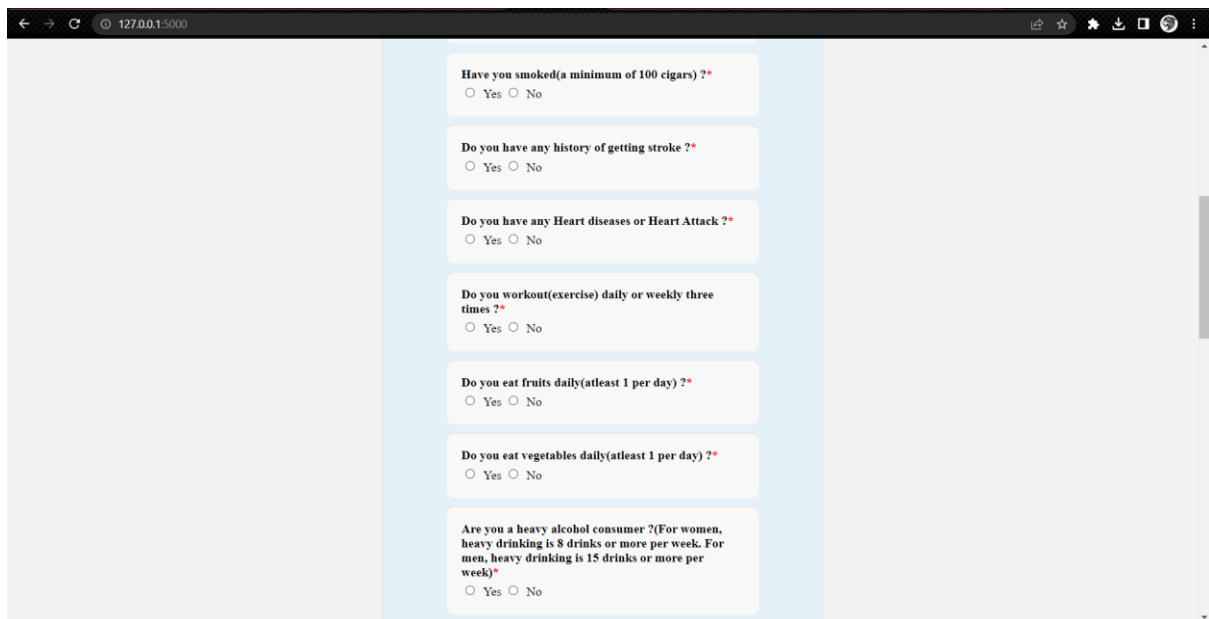
Result:

Opening Page of the Site, here the data from the user is received.



A screenshot of a web browser displaying the first part of a "Diabetes Prediction" form. The browser's address bar shows "127.0.0.1:5000". The form has a light blue background and is titled "Diabetes Prediction". Below the title, a paragraph states: "This page works based on a trained model to predict that the person is having diabetes or not." followed by "We would like to know some details about your health." The form contains five input fields, each with a red asterisk indicating a required field:

- Tell us your name***: A text input field with the placeholder "Enter name here".
- Do you have high Blood Pressure[Bp of 140/80 and above] ?***: Radio buttons for "Yes" and "No".
- Do you have high Cholesterol[240mg/dL and above] ?***: Radio buttons for "Yes" and "No".
- Have you checked your cholesterol in the past five years ?***: Radio buttons for "Yes" and "No".
- What is your Body Mass Index(BMI) value ?***: A text input field with the placeholder "Enter BMI value here".



A screenshot of the same web browser displaying the second part of the "Diabetes Prediction" form. The form continues with seven more input fields, each with a red asterisk indicating a required field:

- Have you smoked(a minimum of 100 cigars) ?***: Radio buttons for "Yes" and "No".
- Do you have any history of getting stroke ?***: Radio buttons for "Yes" and "No".
- Do you have any Heart diseases or Heart Attack ?***: Radio buttons for "Yes" and "No".
- Do you workout(exercise) daily or weekly three times ?***: Radio buttons for "Yes" and "No".
- Do you eat fruits daily(atleast 1 per day) ?***: Radio buttons for "Yes" and "No".
- Do you eat vegetables daily(atleast 1 per day) ?***: Radio buttons for "Yes" and "No".
- Are you a heavy alcohol consumer ?(For women, heavy drinking is 8 drinks or more per week. For men, heavy drinking is 15 drinks or more per week)***: Radio buttons for "Yes" and "No".

← → ↻ 127.0.0.1:5000

Do you enough health care access(like insurances and coverages) ?*

☐ Yes ☐ No

Was there a time in the past 12 months when you needed to see a doctor but could not because of cost? *

☐ Yes ☐ No

Would you say that in general your health ?
(On a scale of 1-5: 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor)*

Value:

Now thinking about your mental health, which includes stress, depression, and problems with emotions, for how many days during the past 30 days was your mental health not good? *

Value:

Now thinking about your physical health, which includes physical illness and injury, for how many days during the past 30 days? *

Value:

← → ↻ 127.0.0.1:5000

Do you have serious difficulty walking or climbing stairs?*

☐ Yes ☐ No

Sex:*

☐ Female ☐ Male

Age:
1 is 18-24 all the way up to 13 with 80 and older. 5 year increments.*

Value:

Tell about your education level ?
with 1 being never attended school or kindergarten only up to 6 being college 4 years or more(Scale here is 1-6)*

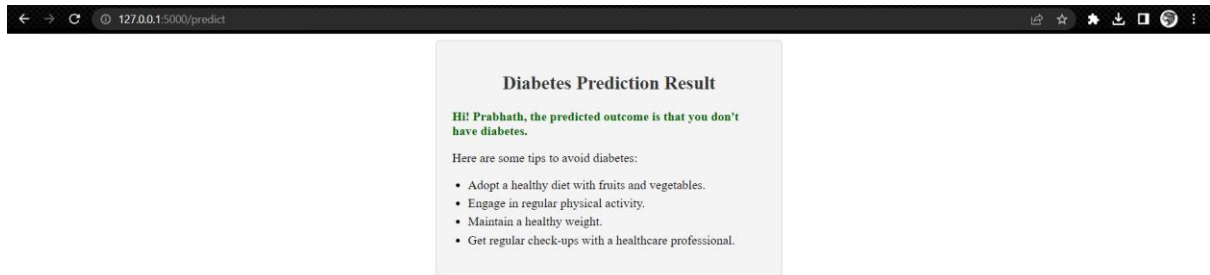
Value:

Annual Household income from all sources ?
On a scale of 1 to 8(with 1 = \$10000 and 8 being \$80000 and more, with \$10000 increment)*

Value:

Output-1:

The page displays that the user does not have diabetes from the date input by them.



Output-2:

The page displays that the user has diabetes from the date input by them.

