Diabetes Prediction Using Machine Learning

Project manual (project development phase)

Date	6 November 2023
Team ID	PNT2022TMID592372
Project Name	Project- Diabetes Prediction Using Machine Learning
Maximum Marks	15 Marks

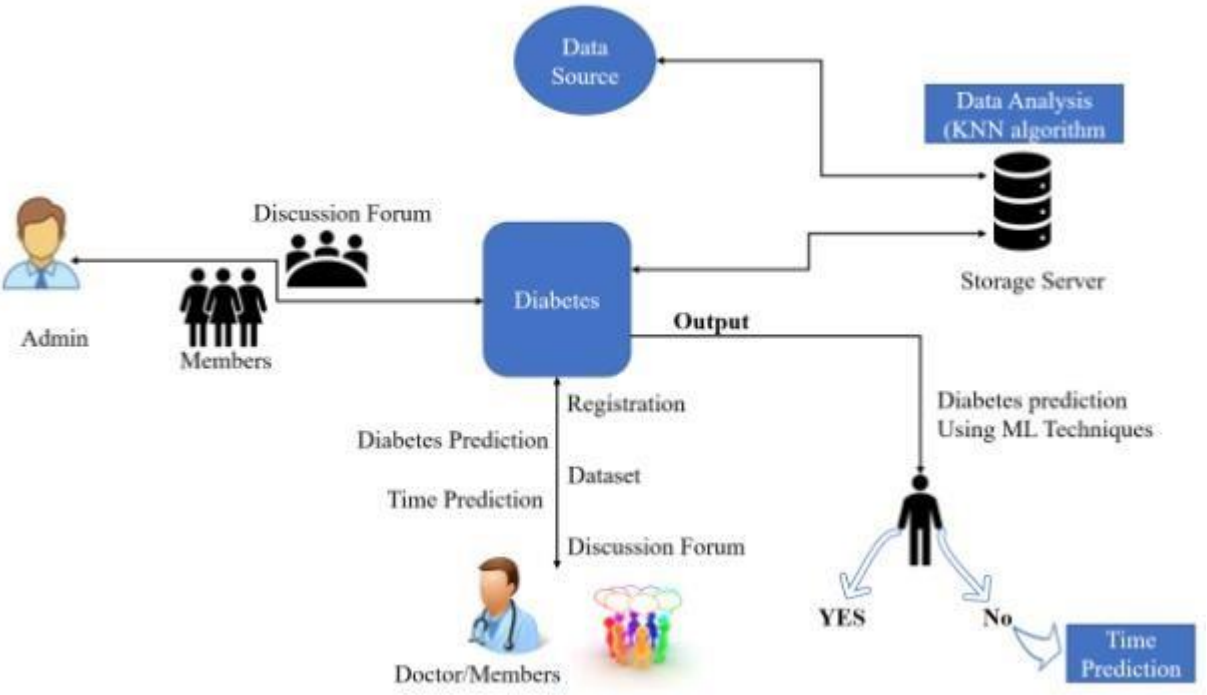
Diabetes Prediction Using Machine Learning

In this project, our objective is to harness the power of machine learning algorithms to forecast the likelihood of diabetes onset in individuals based on their medical records and pertinent factors like age, body mass index (BMI), family medical history, and lifestyle choices. Our dataset will encompass a range of clinical parameters, including blood pressure, BMI, heart conditions, and cholesterol levels.

Our primary aim is to construct a predictive model capable of pinpointing individuals with a heightened risk of developing diabetes. This predictive capability will enable early intervention and proactive measures to prevent the onset of this chronic disease. By leveraging machine learning techniques to scrutinize extensive datasets, we aim to uncover hidden patterns and make precise predictions, which could potentially have a profound impact on saving lives.

In essence, this project holds the potential to make significant contributions to the healthcare domain by enhancing the early detection and prevention of diabetes. The ultimate outcome would be improved health outcomes for both individuals and entire communities.

Technical Architecture:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding ○ Specify the business problem ○ Business requirements ○ Literature Survey ○ Social or Business Impact.

- Data Collection & Preparation ○ Collect the dataset ○ Data preprocessing

- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis

- Model Building ○ Train the machine learning model using different algorithms.
  - Test and evaluate the model's performance to ensure it can make accurate predictions.

- Performance Testing ○ Test the model using various evaluation metrics to measure its effectiveness.

- Model Deployment ○ Save the best model ○ Integrate with Web Framework to create a user interface (UI).

Prior Knowledge:

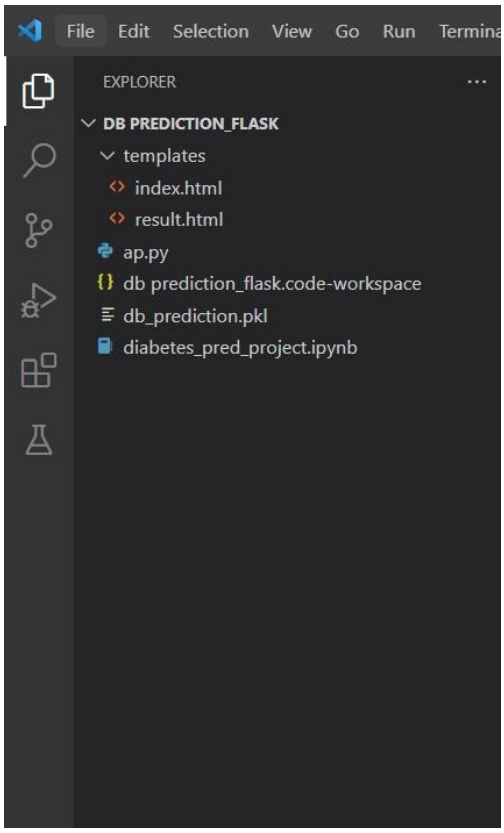
You must have prior knowledge of following topics to complete this project.

- ML Concepts

Supervised learning: https://www.javatpoint.com/supervised-machine-learning

Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning ● Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- KNN: https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning ● Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model- evaluation-error-metrics/
- NLP:-https://www.tutorialspoint.com/natural_language_processing/natural_language_processing_python.htm
- Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0 **Project Structure:**



Create the Project folder which contains files as shown below:

- We are building a flask application which needs HTML pages stored in the templates folder and a python script ap.py for scripting.
- db_prediction.pkl is our saved model. Further we will use this model for flask integration. ● Training folder contains a model training file.

-

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

The core business challenge tackled in this project revolves around the early identification and anticipation of diabetes using advanced machine learning techniques. Our primary objective is to construct a predictive model that can effectively pinpoint individuals with an elevated likelihood of developing diabetes, leveraging their health records and pertinent data. The early detection and management of diabetes carry the potential to enhance healthcare results, curtail expenses, and deliver substantial advantages to healthcare providers and insurance firms. Therefore, developing an accurate and reliable predictive model for diabetes detection can have a significant impact on healthcare outcomes and costs.

Activity 2: Business requirements

The business requirements in the context of the diabetes prediction project are the specific needs and expectations of the business stakeholders concerning the desired project outcomes. For this project, the primary business requirements are as follows:

- Precision in Predictions: The predictive model must exhibit a high degree of accuracy in its ability to identify individuals at a heightened risk of developing diabetes based on their health records and related factors.
- Efficiency: The model should operate efficiently, swiftly analyzing extensive datasets to deliver timely predictions.
- Scalability: It is crucial that the model can adapt to accommodate substantial datasets and future increases in data volume without compromising performance.
- Adaptability: The model should be flexible and amenable to adjustments to account for changes in data sources or input parameters.
- User-Friendly: It should be designed with user-friendliness in mind, ensuring that healthcare providers and insurance companies can readily use and comprehend the model.
- Integration: The model should seamlessly integrate with existing healthcare systems and procedures, streamlining its adoption into established workflows.
- Security: Safeguarding patient data privacy is paramount; the model should possess robust security measures to protect sensitive information.
- Compliance: The model must adhere to pertinent healthcare regulations and standards, ensuring legal and ethical compliance throughout its use.

Activity 3: Literature Survey

A literature survey is an essential step in any diabetes prediction using machine learning:

- **"Machine Learning for Diabetes Prediction: A Review" by E. Şahin et al.**:

This paper presents an extensive review of the most recent research concerning machine learning in the context of diabetes prediction. The authors delve into the challenges faced, the various methodologies applied, and the evaluation metrics used in different studies.

- **"Predicting Type 2 Diabetes Mellitus Using Machine Learning Techniques" by S. Chakraborty et al.**:

  This paper introduces a machine learning-based approach for forecasting the risk of developing type 2 diabetes mellitus. It relies on demographic and clinical data. The authors assess and compare diverse algorithms and feature selection techniques, evaluating their effectiveness.

- **"Deep Learning for Diabetes Prediction: A Review" by Y. Zhao et al.**:

  This review focuses on the application of deep learning in diabetes prediction, with a special emphasis on 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 conducts a comparative analysis of various machine learning algorithms, such as 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.**:

  The authors propose a deep learning methodology for the early identification of diabetic retinopathy by analyzing retinal images. Their approach employs a convolutional neural network (CNN) to categorize images into different disease stages, achieving high accuracy.

Activity 4: Social or Business Impact

- Precise diabetes prediction through machine learning carries substantial implications both socially and for businesses. It enables the early identification of individuals with an elevated risk of diabetes, paving the way for timely interventions and prevention measures. From a business standpoint, this accuracy aids healthcare providers and insurers in optimizing resource allocation and managing healthcare expenses more effectively. Furthermore, this predictive capability fosters a more individualized approach to healthcare, thereby enhancing patient outcomes and adherence to treatment regimens. In summary, accurate diabetes prediction using machine learning has the potential to enhance patient well-being, cut down on healthcare expenditures, and usher in a new era of personalized healthcare services.

Milestone 2: Data Collection & Preprocessing

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Collect the dataset

In our project, we sourced our dataset from a well-known platform, Kaggle.com, and obtained it in a CSV format. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.
Link: Diabetes Health Indicators Dataset | Kaggle
As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Activity 1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
```

Activity 1.2: Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas. In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```
#load the dataset
df=pd.read_csv('/content/diabetes_012_health_indicators_BRFSS2015.csv')
df.head()
```

Activity 2: Data Preprocessing

As we have understood how the data is, let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results.
This activity includes the following steps.

- Handling missing values
- Handling categorical data

- Handling Outliers
- Handling imbalance data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Activity 2.1: Handling missing values

- Let's know the info and describe of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used

```
## can get the data types of columns of data set
df.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
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```
#checking if there are any null values or not
df.isnull().sum()
```

```
Diabetes_012            0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

Activity 2.2: Handling Categorical Values

As we can see our dataset has categorical data, we must convert the categorical data to integer encoding or binary encoding.
To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using Label encoding with the help of list comprehension.

- In our project, there are no categorical features are there . all are float data type only.

```
## can get the data types of columns of data set
df.info()
```

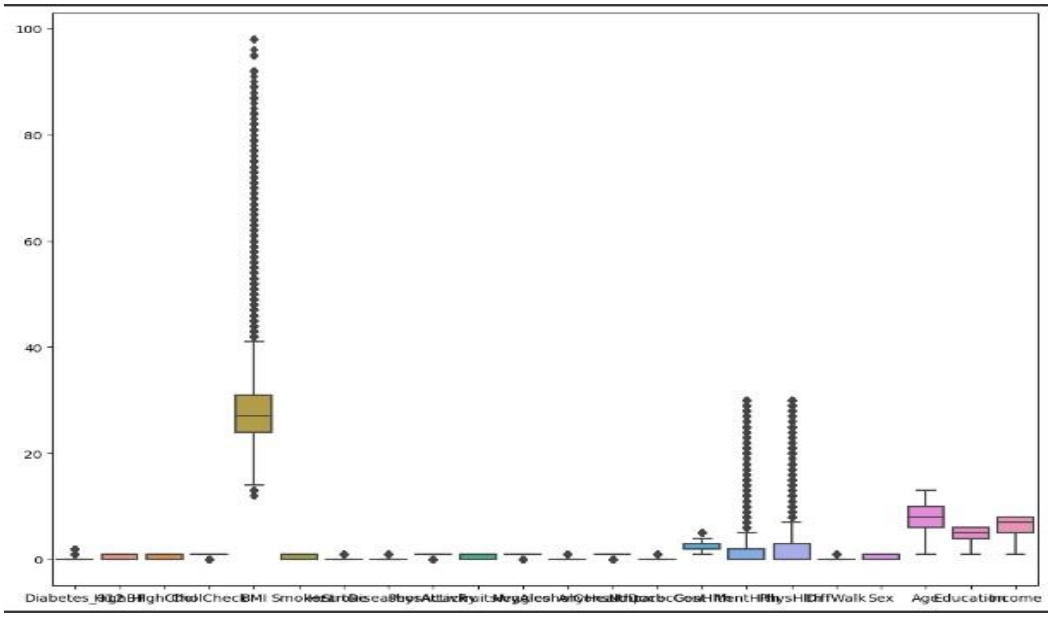
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 253680 entries, 0 to 253679
Data columns (total 22 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   Diabetes_012          253680 non-null  float64
 1   HighBP                253680 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
```

Activity 2.3: Handling outliers

With the help of boxplot, outliers are visualized. And here we are going to find upper bound and lower bound of some columns feature with some mathematical formula

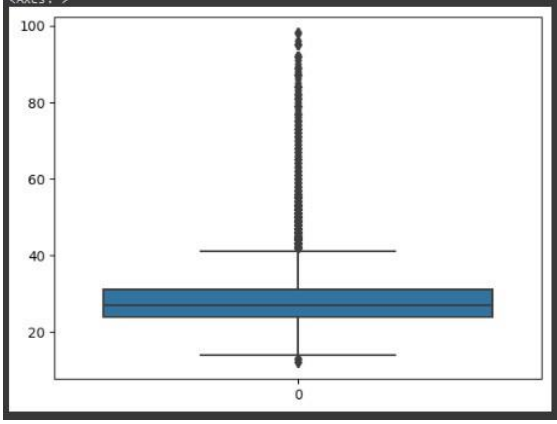
- From the below diagram, we could visualize that some of the feature has outliers Boxplot from seaborn library is used here.

```
#boxplot
sns.boxplot(df)
fig = plt.gcf()
fig.set_size_inches(12, 10)

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




```
up_outlier =['Stroke',  'HeartDiseaseorAttack','HvyAlcoholConsump','NoDocbcCost','GenHlth', 'MentHlth', 'PhysHlth', 'DiffWalk']

low_outlier = ['CholCheck','PhysActivity','Veggies','AnyHealthcare']

sns.boxplot(df.BMI)
```

```
<Axes: >
```



```
#outlier removal by replacement with median
q1=df.BMI.quantile(0.25) # q1
q3=df.BMI.quantile(0.75) # q3
IQR=q3-q1

upper_limit=q3+1.5*IQR  # upper linit value

lower_limit=q1-1.5*IQR
```

```
df['BMI'].median()
```

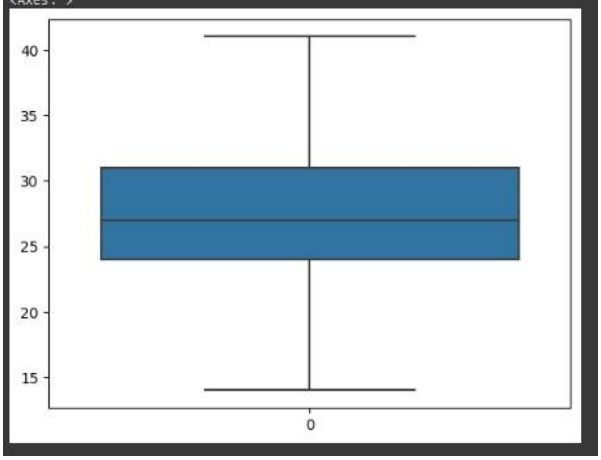
```
27.0
```

```
df['BMI']=np.where(df['BMI']>upper_limit,27,df['BMI'])
df['BMI']=np.where(df['BMI']<lower_limit,27,df['BMI'])


#outlier removal by replacement with median
for i in up_outlier:
  q1=df[i].quantile(0.25) # q1
  q3=df[i].quantile(0.75) # q3
  IQR=q3-q1
  upper_limit=q3+1.5*IQR  # upper linit value
  lower_limit=q1-1.5*IQR
  df[i]=np.where(df[i]>upper_limit,df[i].median(),df[i])
```

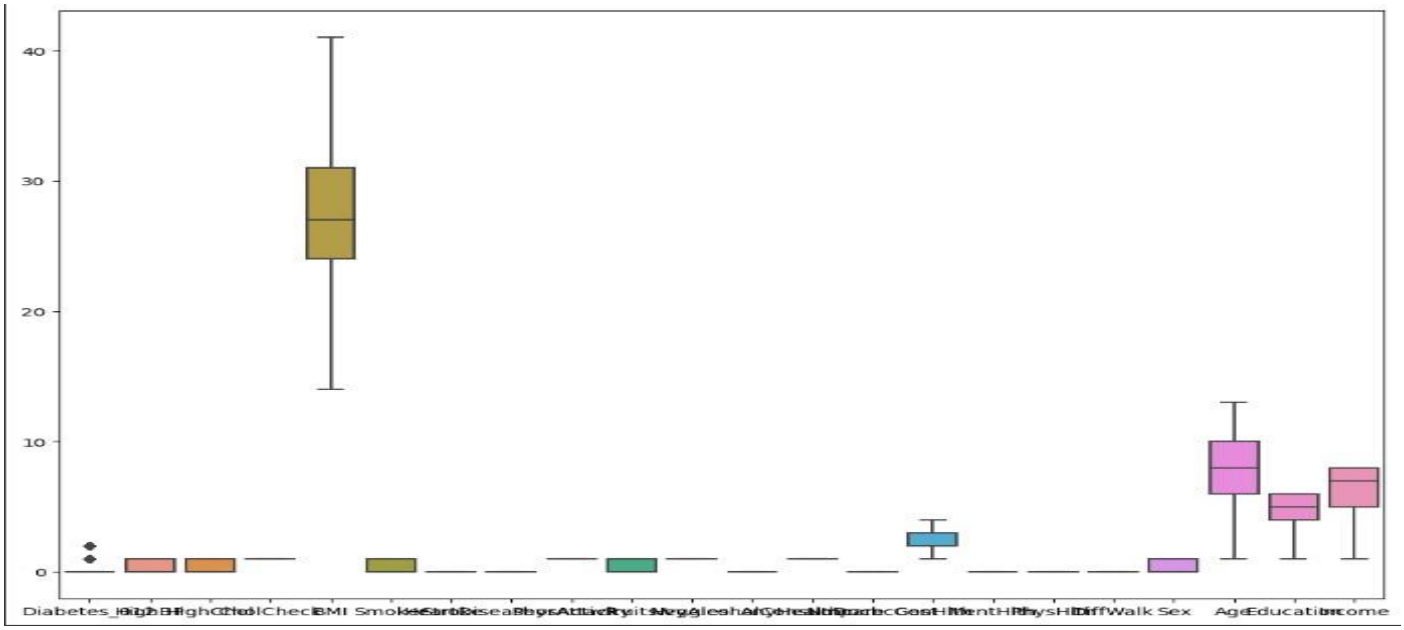
```
for i in low_outlier:
  q1=df[i].quantile(0.25) # q1
  q3=df[i].quantile(0.75) # q3
  IQR=q3-q1
  upper_limit=q3+1.5*IQR  # upper linit value
  lower_limit=q1-1.5*IQR
  df[i]=np.where(df[i]<lower_limit,df[i].median(),df[i])
```

```
sns.boxplot(df.BMI)
```

```
<Axes: >
```




```
sns.boxplot(df)
fig = plt.gcf()
fig.set_size_inches(12, 10)

plt.show()
```



Activity 2.4: Handling imbalance data

First split the data into dependent and independent variable:

x and y split

```
x=df.iloc[:,1:]
x.head()
```

[19]

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

5 rows × 21 columns

```
y=df.Diabetes_012
y.head()
```

[20]

```
0    0.0
1    0.0
2    0.0
3    0.0
4    0.0
Name: Diabetes_012, dtype: float64
```

```
y.value_counts()
```

21]

```
0.0    213703
2.0     35346
1.0      4631
Name: Diabetes_012, dtype: int64
```

We can see data imbalance in the data set. With the help of smote, we can balance the data

.

balancing the imbalance dataset

```
#balancing the data
smote=SMOTE()
```

2]

```
x_smote,y_smote = smote.fit_resample(x,y)
```

3]

```
y_smote.value_counts()
```

4]

```
0.0    213703
2.0    213703
1.0    213703
Name: Diabetes_012, dtype: int64
```

Activity 2.5 : Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set
For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x_smote, y_smote, test_size, random_state.

Scaling is done to standardize the range of independent variables or features of the input data.

```
# descriptive statistics
# statistical analysis of the data
df.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
mean	0.296921	0.429001	0.424121	0.962670	28.382364	0.443169	0.040571	0.094186	0.756544
std	0.698160	0.494934	0.494210	0.189571	6.608694	0.496761	0.197294	0.292087	0.429169
min	0.000000	0.000000	0.000000	0.000000	12.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	1.000000	24.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	0.000000	1.000000	27.000000	0.000000	0.000000	0.000000	1.000000
75%	0.000000	1.000000	1.000000	1.000000	31.000000	1.000000	0.000000	0.000000	1.000000
max	2.000000	1.000000	1.000000	1.000000	98.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 22 columns

```
# train test split
x_train,x_test,y_train,y_test = train_test_split(x_smote,y_smote,test_size=0.3,random_state=47)
```

```
x_train.shape
```

```
(448776, 21)
```

scaling

```
# scaling independent variables
scale = MinMaxScaler()
```

```
x_train_scaled= pd.DataFrame(scale.fit_transform(x_train),columns=x_train.columns)
x_test_scaled= pd.DataFrame(scale.fit_transform(x_test),columns=x_test.columns)
```

Milestone 3: Exploratory Data Analysis

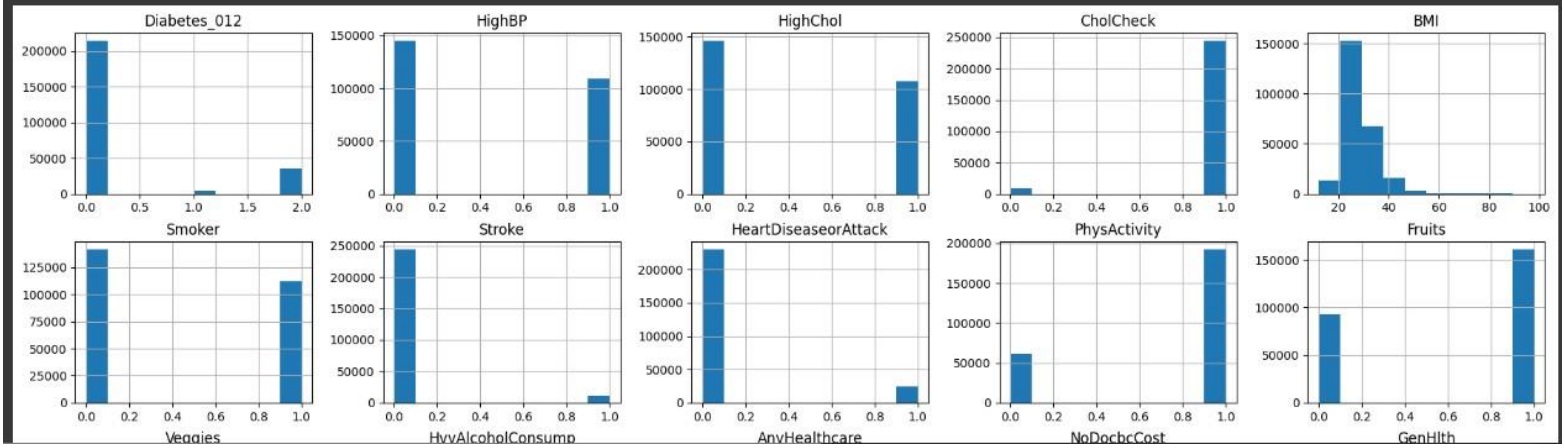
Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features

Activity 2: Visual analysis

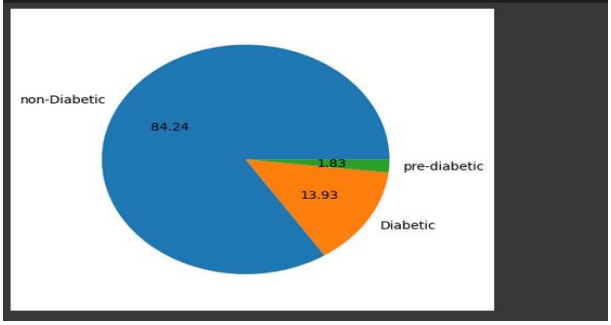
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

```
# Histogram for a numerical column

df.hist(figsize=(22,15))
plt.show()
```

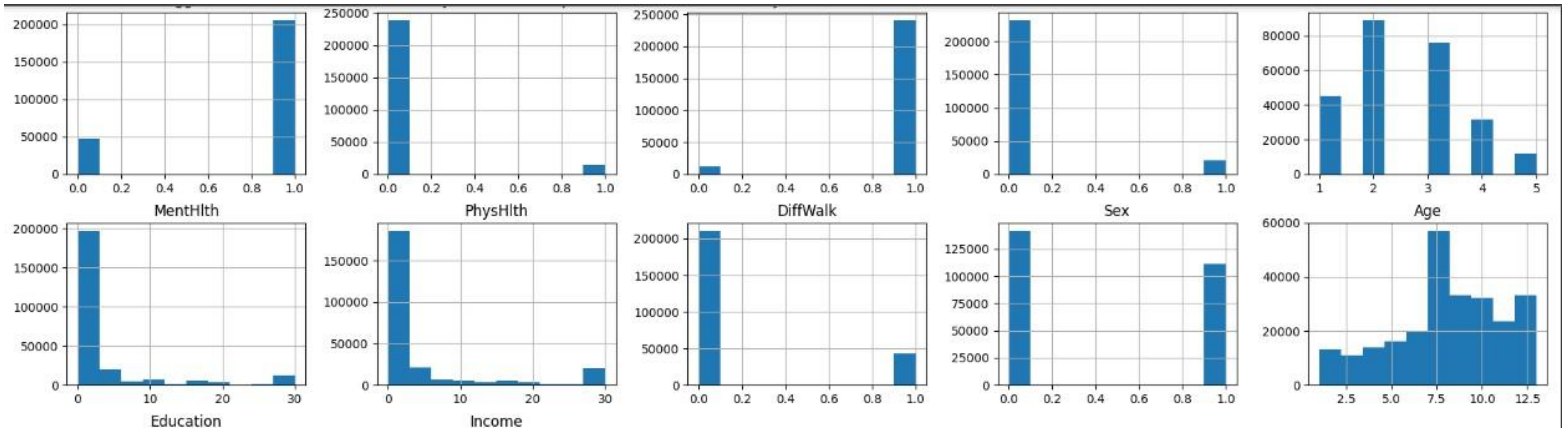


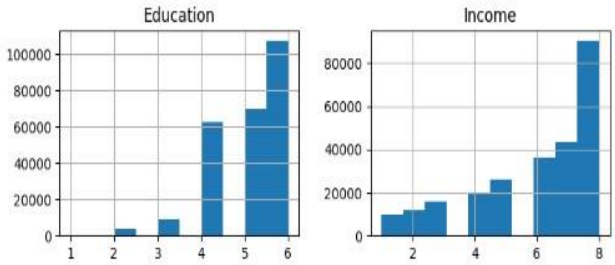
Pie Chart:

```
# pie chart

labels=["non-Diabetic","Diabetic","pre-diabetic"]
plt.pie(df["Diabetes_012"].value_counts() , labels =labels ,autopct='%.02f' )
plt.show()
```



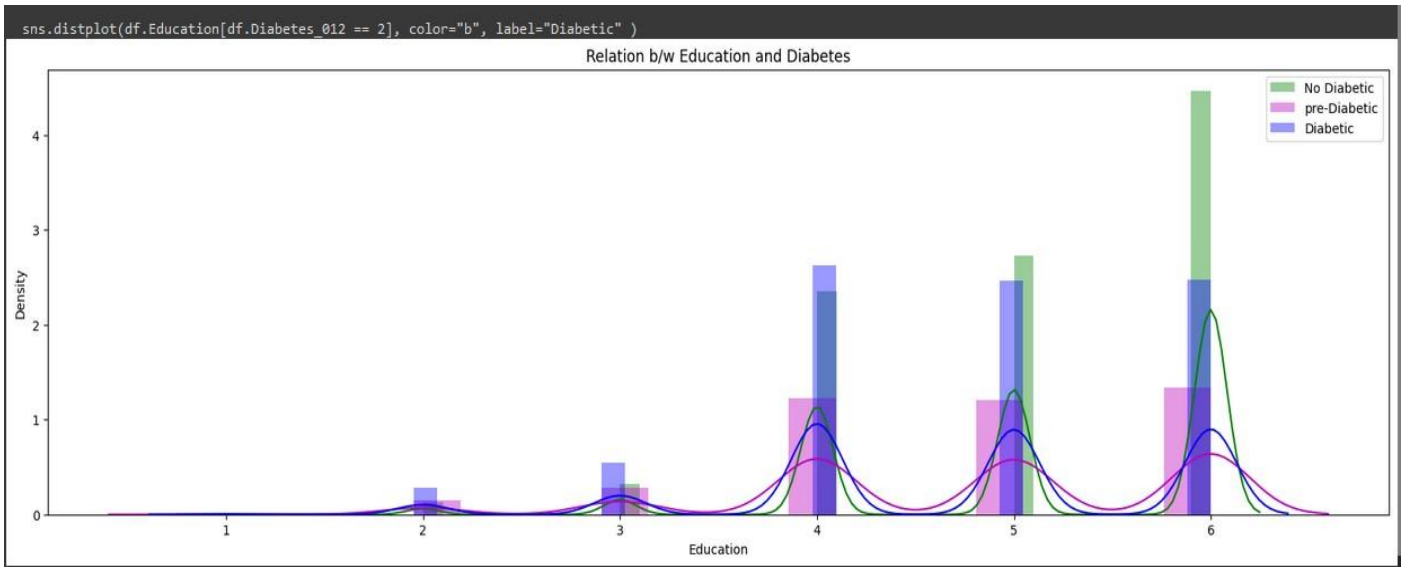
Distplot:





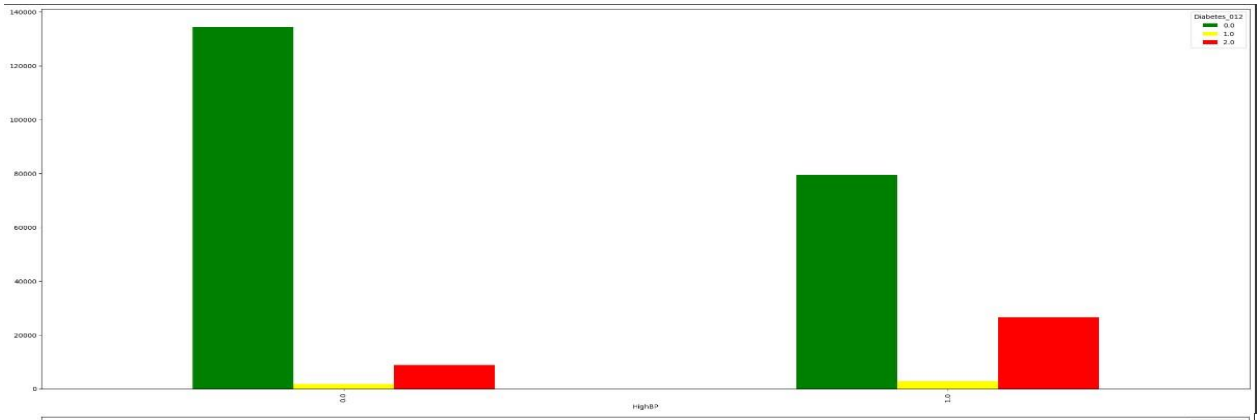

```
#distplot - seeing ther relation between education and diabetes
plt.figure(figsize=(20,6))


sns.distplot(df.Education[df.Diabetes_012 == 0], color="g", label="No Diabetic" )
sns.distplot(df.Education[df.Diabetes_012 == 1], color="m", label="pre-Diabetic" )
sns.distplot(df.Education[df.Diabetes_012 == 2], color="b", label="Diabetic" )

plt.title("Relation b/w Education and Diabetes")

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

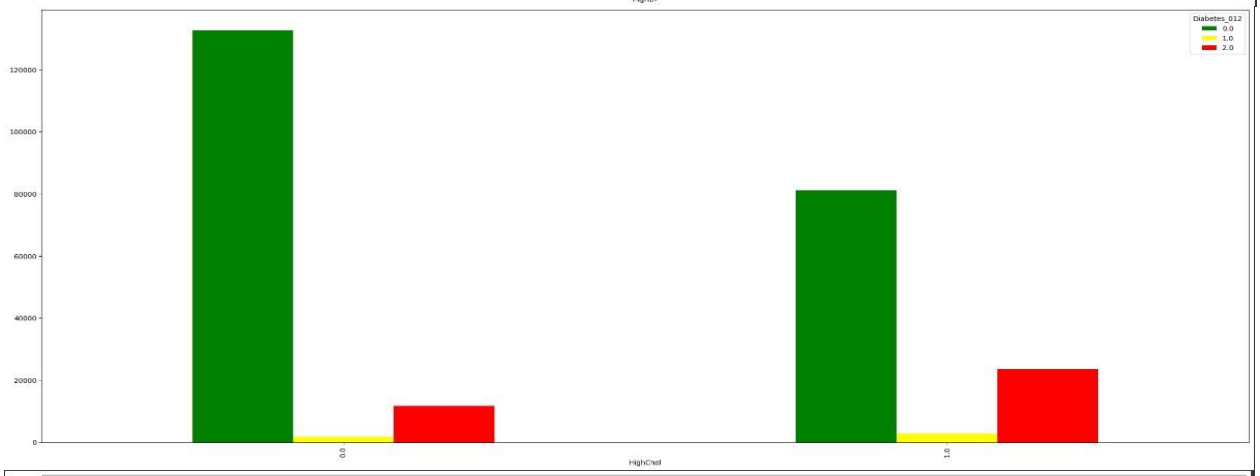
Plot:

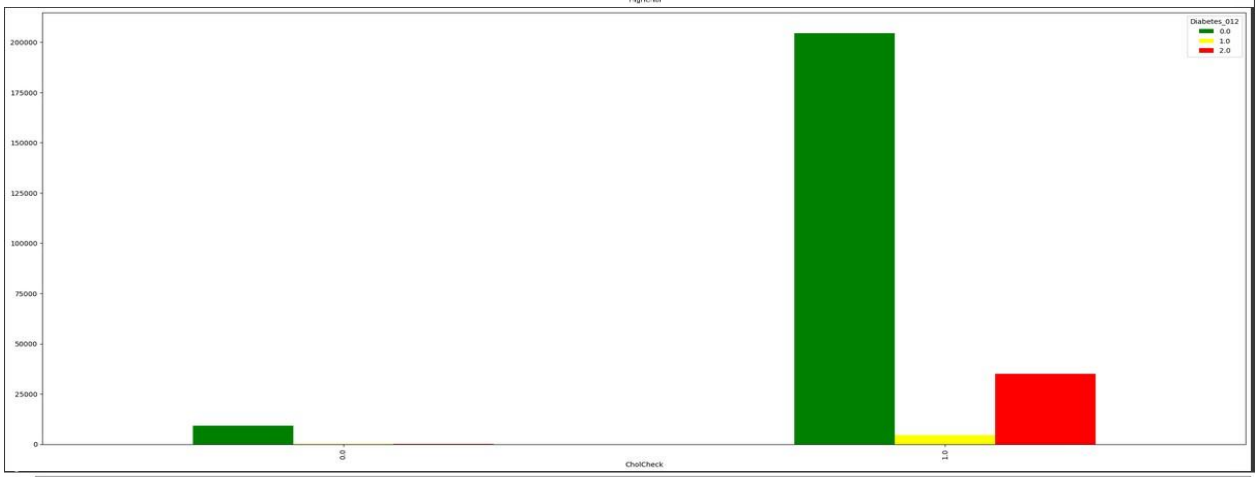


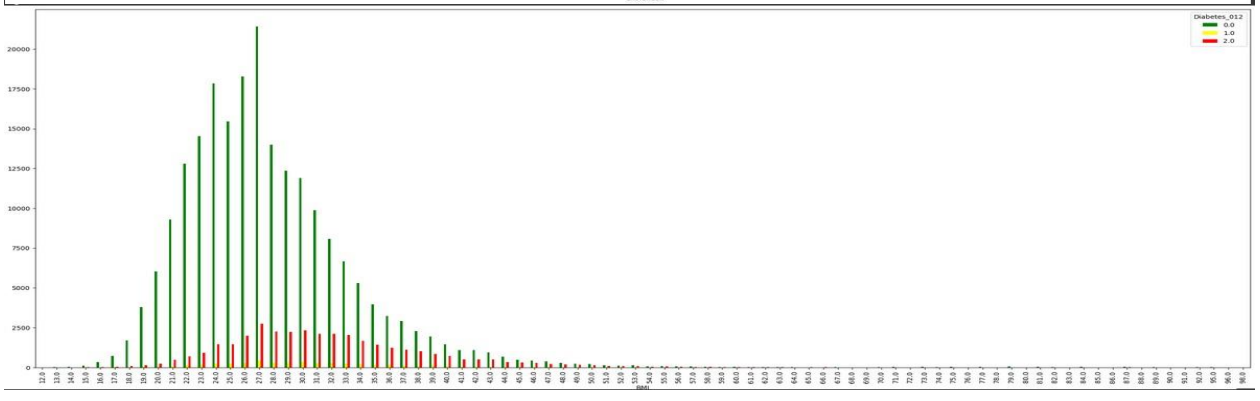
Comparing all Attributes:

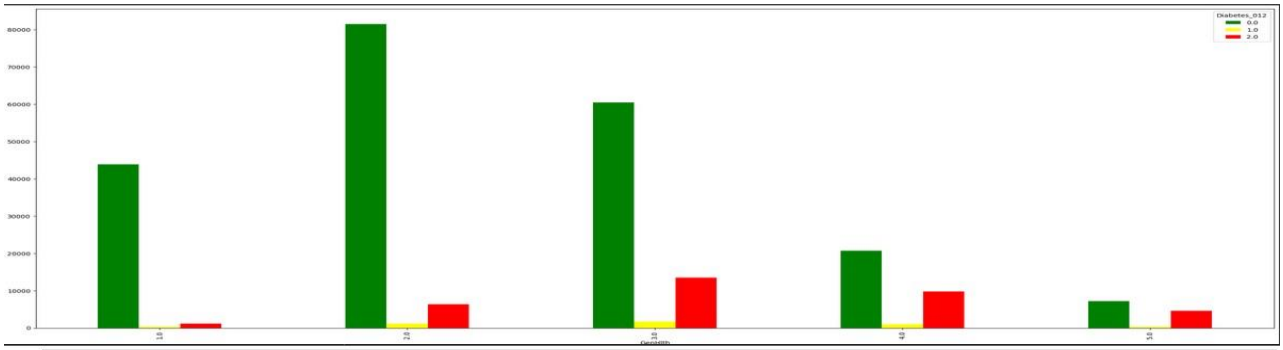
```
#comparing all attributes with Diabetes_012
for i in attributes:
  plt.figure(figsize=(5,3))
  pd.crosstab(df[i],df.Diabetes_012).plot(kind="bar",figsize=(30,12),color=['green', 'yellow','red' ])

  plt.show()
```

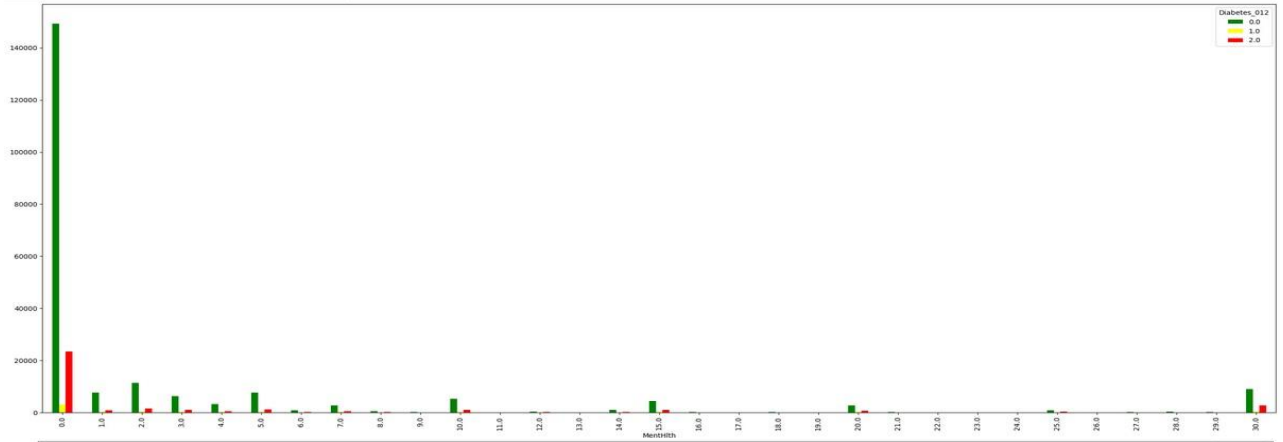

Diabetes_012
0.0
1.0
2.0
HighBP

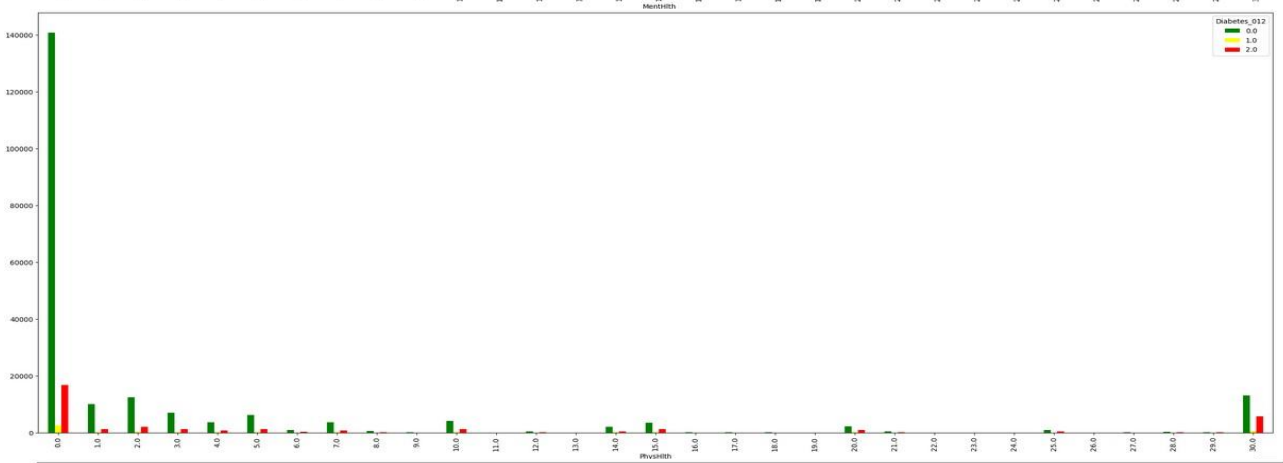

Diabetes_012
0.0
1.0
2.0
HighChol

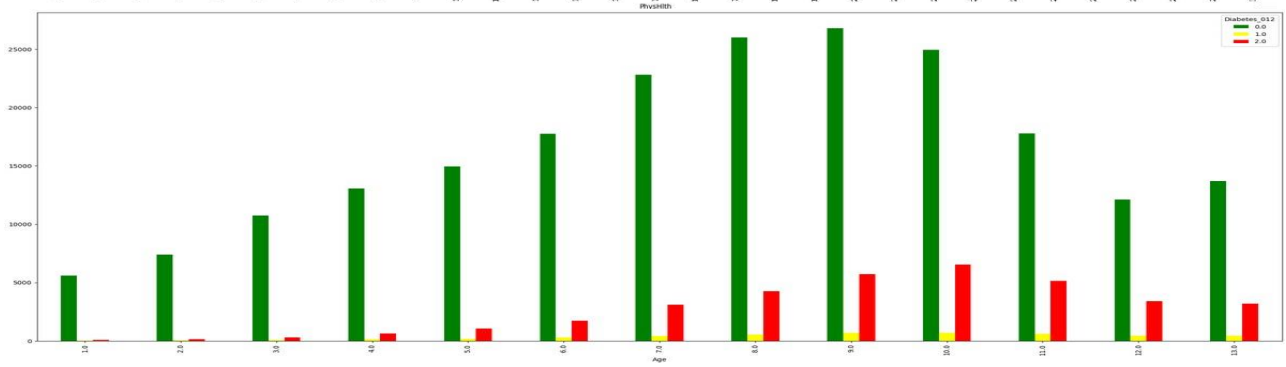

Diabetes_012
0.0
1.0
2.0
CholCheck

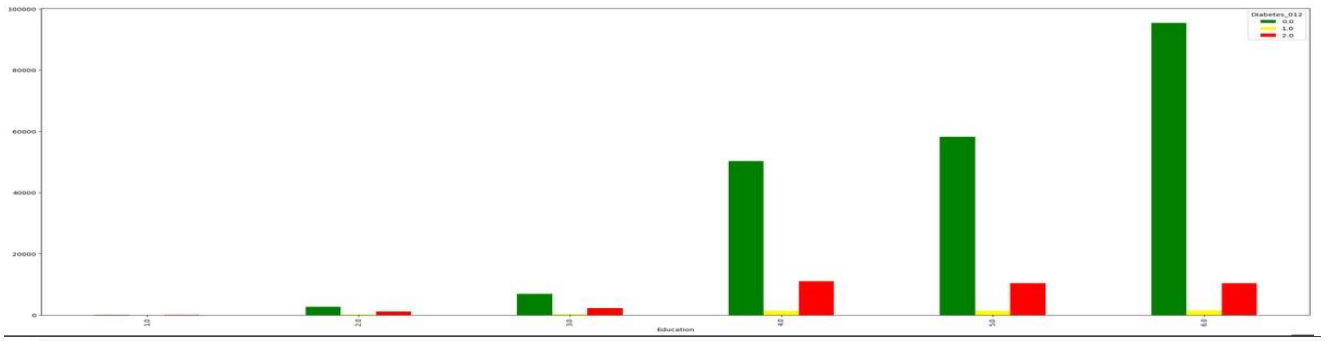

Diabetes_012
0.0
1.0
2.0

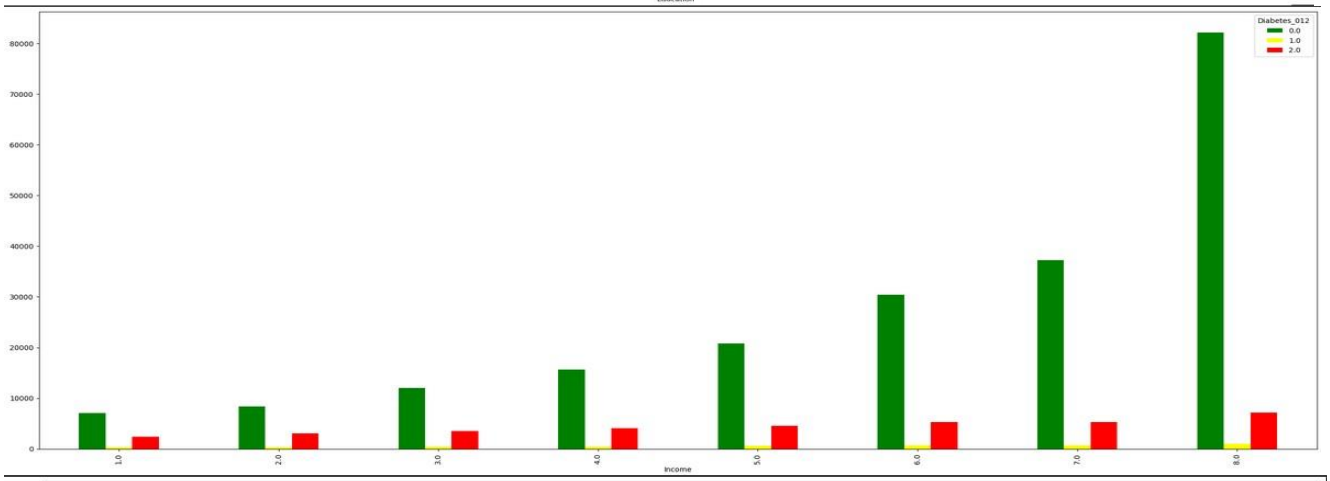

Diabetes_012
0.0
1.0
2.0
GenHlth

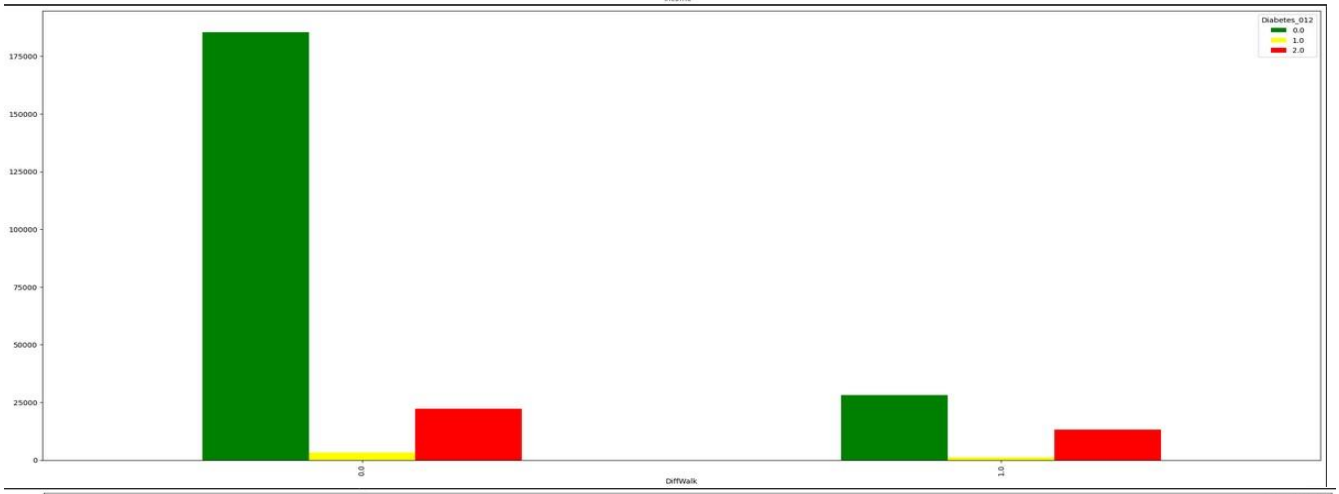

Diabetes_012
0.0
1.0
2.0
MentHlth

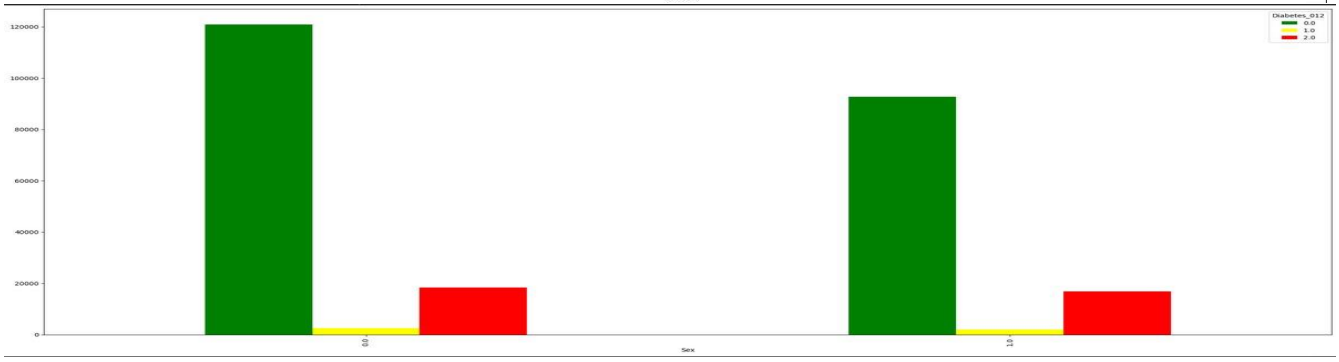

Diabetes_012
0.0
1.0
2.0
PhysHlth


Diabetes_012
0.0
1.0
2.0
Age







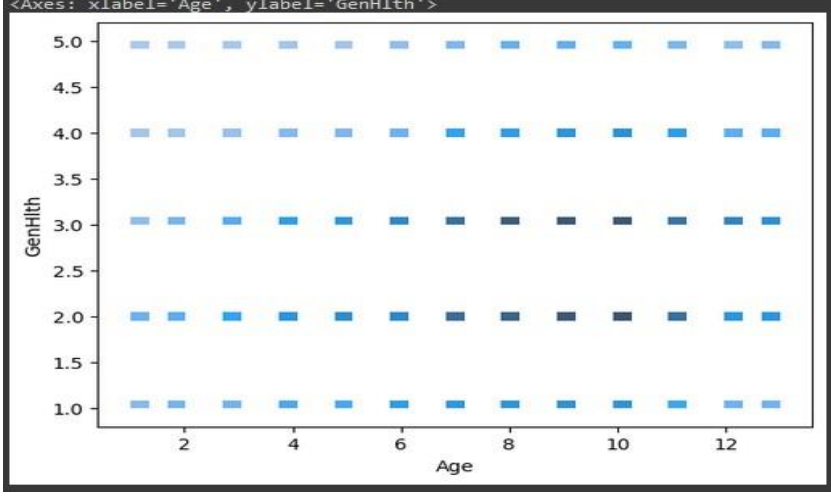


Activity 2.1: Univariate analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as distplot and count plot.
In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features.

```
#Univariate Analysis
sns.histplot(data=df,x='Age',y='GenHlth')
```

```
<Axes: xlabel='Age', ylabel='GenHlth'>
```

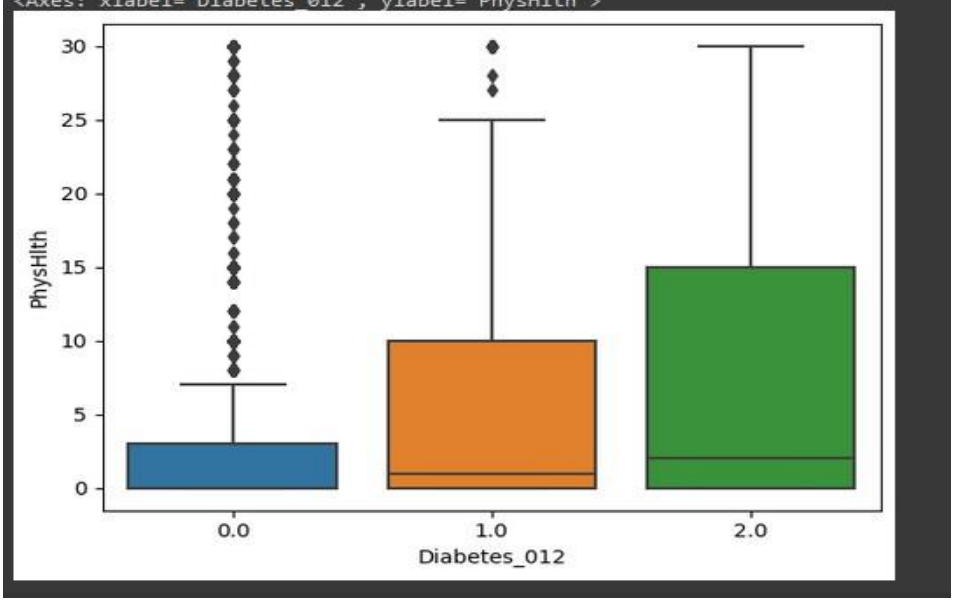


Activity 2.2: Bivariate analysis

To find the relation between two features we use bivariate analysis. Here we are visualizing. PhysHlth plot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

```
#Bivariate Analysis
sns.boxplot(data=df,x='Diabetes_012',y='PhysHlth')
```

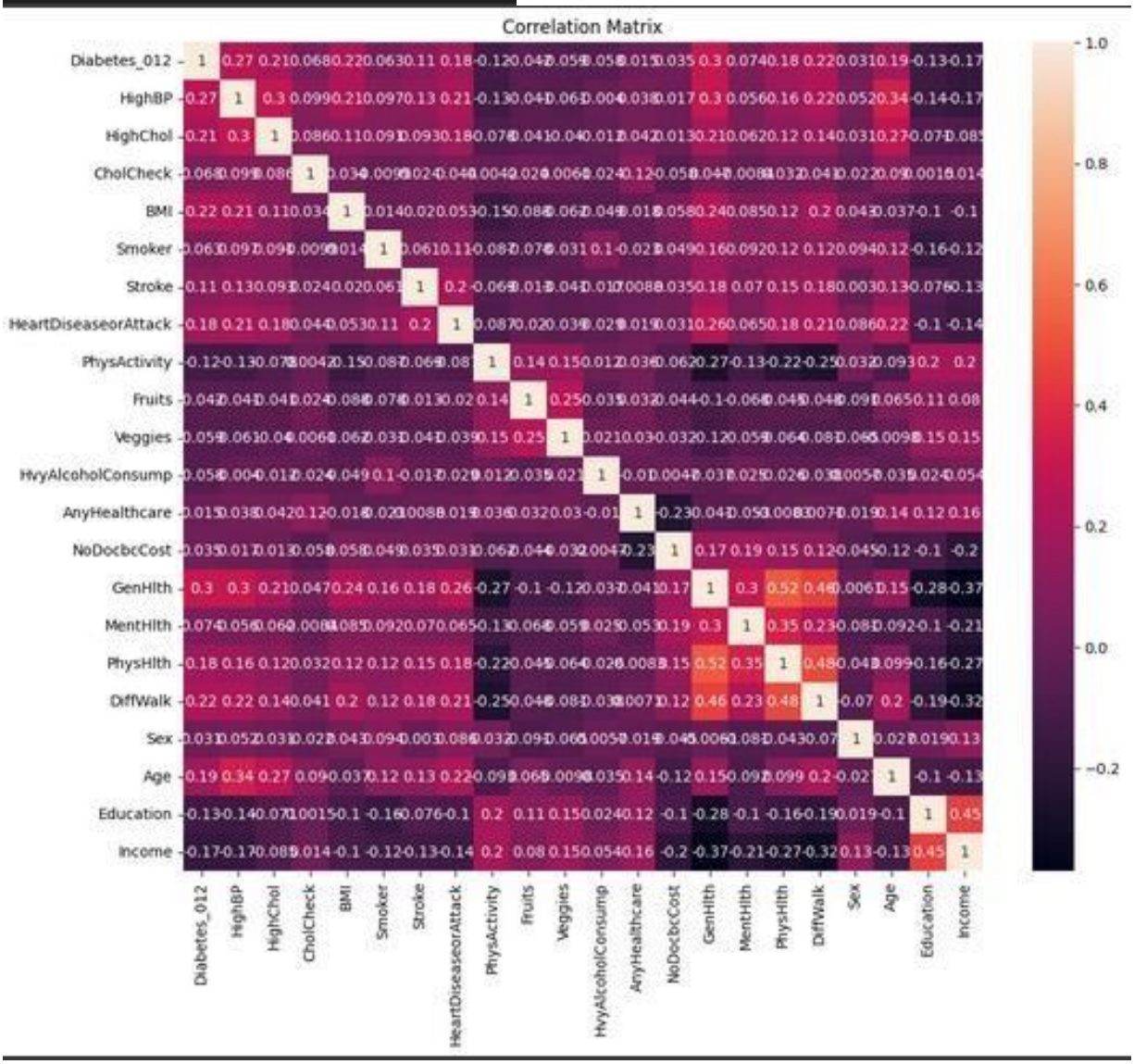
```
<Axes: xlabel='Diabetes_012', ylabel='PhysHlth'>
```



Activity 2.3: Multivariate analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used heatmap from seaborn package.

```
# Multivariate Analysis

corr_matrix = df.corr()
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True)
plt.title('Correlation Matrix')
plt.show()
```



Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project, we are applying three classification algorithms. The best model is saved based on its performance.

Activity 1.1: Random Forest Regressor

A function named random forest regressor is created and train and test data are passed as the parameters. Inside the function, random forest regressor algorithm is initialized and training data is passed to the model with the fit() function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model with R2_score.

```
model5 =RandomForestClassifier(max_depth = 12,n_estimators = 10, random_state=47)

model5.fit(x_train_scaled,y_train)

RandomForestClassifier(max_depth=12, n_estimators=10, random_state=47)

y_pred5 = model5.predict(x_test_scaled)

y_pred5_train = model5.predict(x_train_scaled)

print("Test accuracy", accuracy_score(y_test,y_pred5))
print("Train accuracy", accuracy_score(y_train,y_pred5_train))
```

```
Test accuracy 0.7291364456437532
Train accuracy 0.7413475765192434
```

```
pd.crosstab(y_test,y_pred5)
```

col_0	0.0	1.0	2.0
Diabetes_012			
0.0	50611	586	13020
1.0	2988	47276	13844
2.0	8231	13427	42350

```
print(classification_report(y_test,y_pred5))
```

	precision	recall	f1-score	support
0.0	0.82	0.79	0.80	64217
1.0	0.77	0.74	0.75	64108
2.0	0.61	0.66	0.64	64008
accuracy			0.73	192333
macro avg	0.73	0.73	0.73	192333
weighted avg	0.73	0.73	0.73	192333

```
from sklearn.metrics import confusion_matrix, classification_report
# Plot the confusion matrix using Seaborn's heatmap
cm5 = confusion_matrix(y_test,y_pred5)
sns.heatmap(cm5, annot=True, fmt='d', cmap='Blues', cbar=True)

plt.show()
```

Activity 1.2: Decision Tree Regressor

A function named decision Tree regressor is created and train and test data are passed as the parameters. Inside the function, decision Tree regressor algorithm is initialized and training data is passed to the model with fit() function. Test data is predicted with predict () function and saved in a new variable. For evaluating the model, For evaluating the model with R2_score

```
model4 = DecisionTreeClassifier(max_depth=12)

model4.fit(x_train_scaled,y_train)

DecisionTreeClassifier(max_depth=12)

y_pred4 = model4.predict(x_test_scaled)

y_pred4_train = model4.predict(x_train_scaled)

print("Test accuracy", accuracy_score(y_test,y_pred4))
print("Train accuracy", accuracy_score(y_train,y_pred4_train))
```

```
Test accuracy 0.6845471135998502
Train accuracy 0.6935910119970765
```

```
pd.crosstab(y_test,y_pred4)
```

col_0	0.0	1.0	2.0
Diabetes_012			
0.0	50924	1947	11346
1.0	5532	43746	14830
2.0	10585	16432	36991

```
print(classification_report(y_test,y_pred4))
```

	precision	recall	f1-score	support
0.0	0.76	0.79	0.78	64217
1.0	0.70	0.68	0.69	64108
2.0	0.59	0.58	0.58	64008
accuracy			0.68	192333
macro avg	0.68	0.68	0.68	192333
weighted avg	0.68	0.68	0.68	192333

```
from sklearn.metrics import confusion_matrix, classification_report
# Plot the confusion matrix using Seaborn's heatmap
cm4 = confusion_matrix(y_test,y_pred4)
sns.heatmap(cm4, annot=True, fmt='d', cmap='Blues', cbar=True)

plt.show()
```

Activity 1.3: Logistic Regressor

To evaluate the performance of a logistic regression model, we need to test it on a separate dataset that it has not seen during training. This separate dataset is called the test data. We typically split the available data into two parts, the training data and the test data. The model is trained on the training data, and then tested on the test data to see how well it generalizes to new, unseen data.

```
## Model building

model1 = LogisticRegression()

model1.fit(x_train_scaled,y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
y_pred1 = model1.predict(x_test_scaled)

y_pred1_train =  model1.predict(x_train_scaled)

print('Testing Accuracy = ', accuracy_score(y_test,y_pred1))
print('Training Accuracy = ', accuracy_score(y_train,y_pred1_train))
```

```
Testing Accuracy =  0.5076663911029309
Training Accuracy =  0.5078034475996934
```

```
pd.crosstab(y_test,y_pred1)
```

col_0	0.0	1.0	2.0
Diabetes_012			
0.0	42438	8602	13177
1.0	17822	14855	31431
2.0	11428	12232	40348

```
print(classification_report(y_test,y_pred1))
```

```
              precision    recall  f1-score   support

         0.0       0.59      0.66      0.62     64217
         1.0       0.42      0.23      0.30     64108
         2.0       0.47      0.63      0.54     64008

    accuracy                           0.51    192333
   macro avg       0.49      0.51      0.49    192333
weighted avg       0.49      0.51      0.49    192333
```

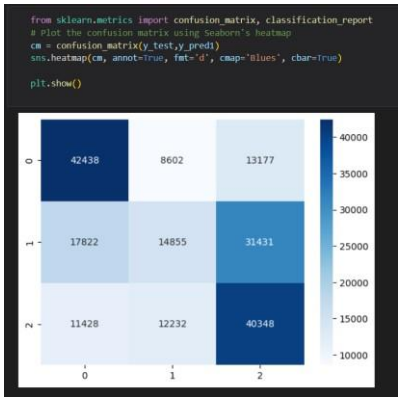
```
from sklearn.metrics import confusion_matrix, classification_report
# Plot the confusion matrix using Seaborn's heatmap
cm = confusion_matrix(y_test,y_pred1)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=True)

plt.show()
```

Milestone 5: Performance Testing

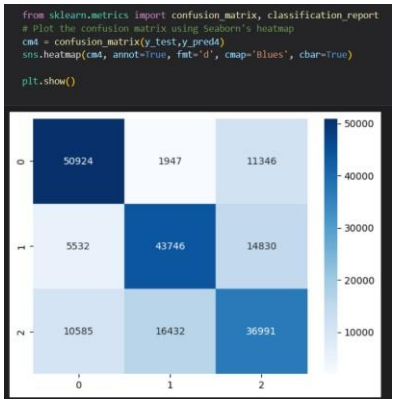
Activity 1: Testing model with multiple evaluation metrics Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more

comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

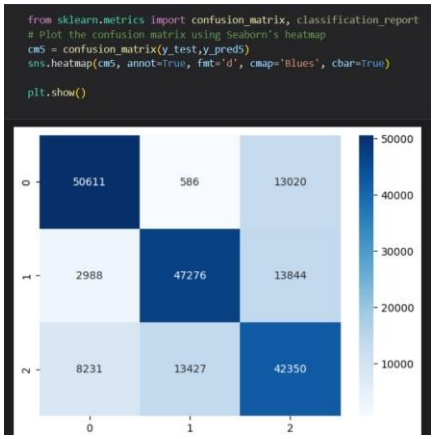
Logistic regression:



Decision tree:



Random forest:



Activity 1.1: Compare the model

on comparing the above three models, with their accuracy score on training and testing data and confusion metrics, From the above three models random forest regressor is performing well.

Milestone 6: Model Deployment

Activity 1: Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
import pickle


pickle.dump(model5,open('db_prediction.pkl','wb'))
```

Activity 2: Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

Activity 2.1: Building Html Pages:

For this project create two HTML files namely

· index.html ·
result.html

and save them in the templates folder.

Activity 2.2: Build Python code:

Import the libraries in python file

```
from flask import Flask, render_template, request
import pickle
import numpy as np
import pandas as pd
```

Render HTML page:

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (_name_) as argument.

```
app = Flask(__name__)

# Load the model from the pickle file
model = pickle.load(open('db_prediction.pkl','rb'))
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

Retrieves the value from UI:

```
@app.route('/')
def start():
    return render_template('index.html')
```

In the above example, '/' URL is bound with the index.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Main Function:

```
@app.route("/login", methods=['POST']) def
login():
    Sex =
request.form['Sex']     if
(Sex=='Male'):         Sex=1
else:
        Sex=0

    HighBP = request.form['HighBP']
if (HighBP=='yes'):
HighBP=1     else:
        HighBP=0

    Fruits = request.form['Fruits']
if (Fruits=='yes'):
Fruits=1     else:
        Fruits=0

    Veggies = request.form['Veggies']
if (Veggies=='yes'):
Veggies=1     else:
        Veggies=0

    AnyHealthcare =
request.form['AnyHealthcare']     if
(AnyHealthcare=='yes'):         AnyHealthcare=1
else:
        AnyHealthcare=0

    Highchol
=request.form['Highchol']     if
(Highchol=='yes'):         Highchol=1
else:
        Highchol=0

    Age=
request.form['Age']     if
(Age=='1'):         Age=1
elif (Age=='2'):
```

```
        Age=2
elif (Age=='3'):
Age=3     elif
(Age=='4'):
Age=4     elif
(Age=='5'):
Age=5     elif
(Age=='6'):
Age=6     elif
(Age=='7'):
Age=7     elif
(Age=='8'):
Age=8     elif
(Age=='9'):
Age=9     elif
(Age=='10'):
Age=10     elif
(Age=='11'):
Age=11     elif
(Age=='12'):
Age=12
else:
        Age=13

    BMI = request.form['BMI']

    Smoker = request.form['Smoker']
if (Smoker=='yes'):
Smoker=1     else:
        Smoker=0

    CholCheck =
request.form['CholCheck']     if
(CholCheck=='yes'):         CholCheck=1
else:
        CholCheck=0

    Stroke = request.form['Stroke']
if (Stroke=='yes'):
Stroke=1     else:
        Stroke=0

    HvyAlcoholConsump= request.form['HvyAlcoholConsump']
```

```
    if
(HvyAlcoholConsump=='yes'):
HvyAlcoholConsump=1     else:
        HvyAlcoholConsump=0

    Diffwalk=
request.form['Diffwalk']     if
(Diffwalk=='yes'):         Diffwalk=1
else:
        Diffwalk=0

    HeartDiseaseorAttack =
request.form['HeartDiseaseorAttack']     if
(HeartDiseaseorAttack=='yes'):         HeartDiseaseorAttack=1
else:
        HeartDiseaseorAttack=0

    PhysActivity=
request.form['PhysActivity']     if
(PhysActivity=='yes'):         PhysActivity=1
else:
        PhysActivity=0

    NoDocbcCost = request.form['NoDocbcCost']
if (NoDocbcCost=='yes'):
NoDocbcCost=1     else:
        NoDocbcCost=0

    GenHlth =request.form['GenHlth']
if (GenHlth=='1'):         GenHlth=1
elif (GenHlth=='2'):
GenHlth=2     elif (GenHlth=='3'):
GenHlth=3     elif (GenHlth=='4'):
GenHlth=4     else:
        GenHlth=5

    MentHlth = request.form['MentHlth']
    PhysHlth =request.form['PhysHlth']
    Education =request.form['Education']
```

```
    if (Education=='1'):
Education=1     elif
(Education=='2'):
Education=2     elif
(Education=='3'):
Education=3     elif
(Education=='4'):
Education=4     else:
        Education=5

    Income
=request.form['Income']     if
(Income=='1'):         Income=1
elif (Income=='2'):
Income=2     elif (Income=='3'):
Income=3     elif (Income=='4'):
Income=4     elif (Income=='5'):
Income=5     elif (Income=='6'):
Income=6     elif (Income=='7'):
Income=7                 else:
        Income=8


    input_data =[[float(HighBP) ,float(Highchol) ,
float(CholCheck),float(BMI),float(Smoker),float(Stroke),float(HeartDiseaseorAttack),float(PhysActivit
y),float(Fruits),float(Veggies), float(HvyAlcoholConsump), float(AnyHealthcare), float(NoDocbcCost),
float(GenHlth),float(PhysHlth),float(MentHlth),float(Diffwalk),float(Sex),float(Age),float(Education)
,float(Income) ]]
     output=
model.predict(input_data)
         if (output
== 0):
        return render_template('result.html', result='Good News, Diabetes not present',rec="According
to our prediction model, there is currently no indication that you have diabetes. However, it's
important to view this as a snapshot, and maintaining a healthy lifestyle is key to ongoing
wellbeing.",rec1="To continue minimizing the risk of diabetes, we recommend staying active, eating a
balanced diet rich in fruits and vegetables, and maintaining a healthy weight. Regular exercise and a
nutritious diet contribute to overall health and well-being.")
```

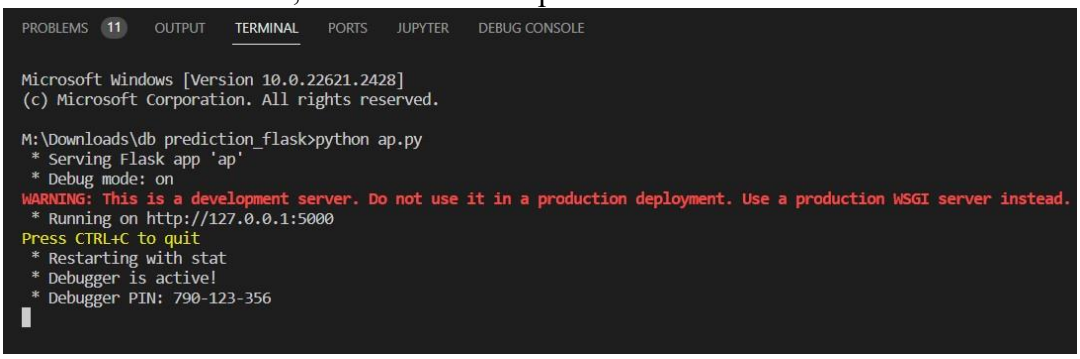
Here we are routing our app to prediction () function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model Predict() function. This function returns the prediction. And this prediction value will be

```
    elif (output == 1):
        return render_template('result.html', result='there is a risk of getting diabetes',rec="Our
analysis suggests that there is a potential risk for developing diabetes in the future. It's
essential to understand that this is not a definite prediction but an indication to be mindful of
your health.",rec1="To reduce the risk of developing diabetes, consider adopting preventive
measures. Focus on maintaining a balanced diet, engaging in regular physical activity, and
scheduling routine health check-ups. These lifestyle changes can play a significant role in
minimizing the risk.")     else:
        return render_template('result.html', result='Oh no! you are sufferning from
diabetes.',rec="Based on our analysis, it appears that you currently have diabetes. We understand
that this may be concerning, but it's important not to panic. This prediction is not a diagnosis, and
we strongly recommend consulting with a healthcare professional for a comprehensive
evaluation.",rec1="To address this concern promptly, we recommend scheduling a check-up with your
healthcare provider. They can provide personalized advice and guidance. Additionally, consider making
immediate lifestyle changes such as adjusting your diet and incorporating regular exercise into your
routine.")

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

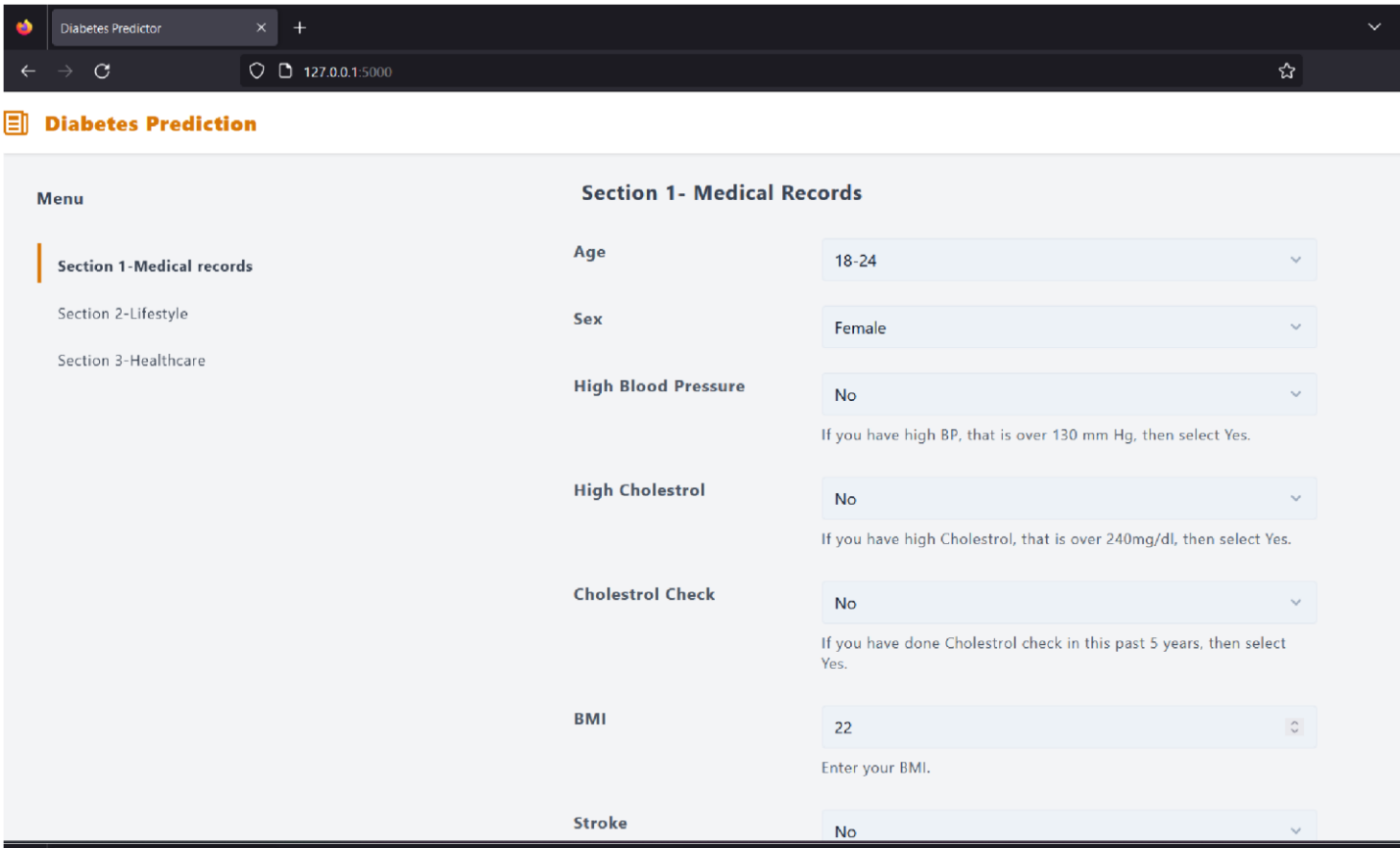
rendered to the text that we have mentioned in the submit.html page earlier.

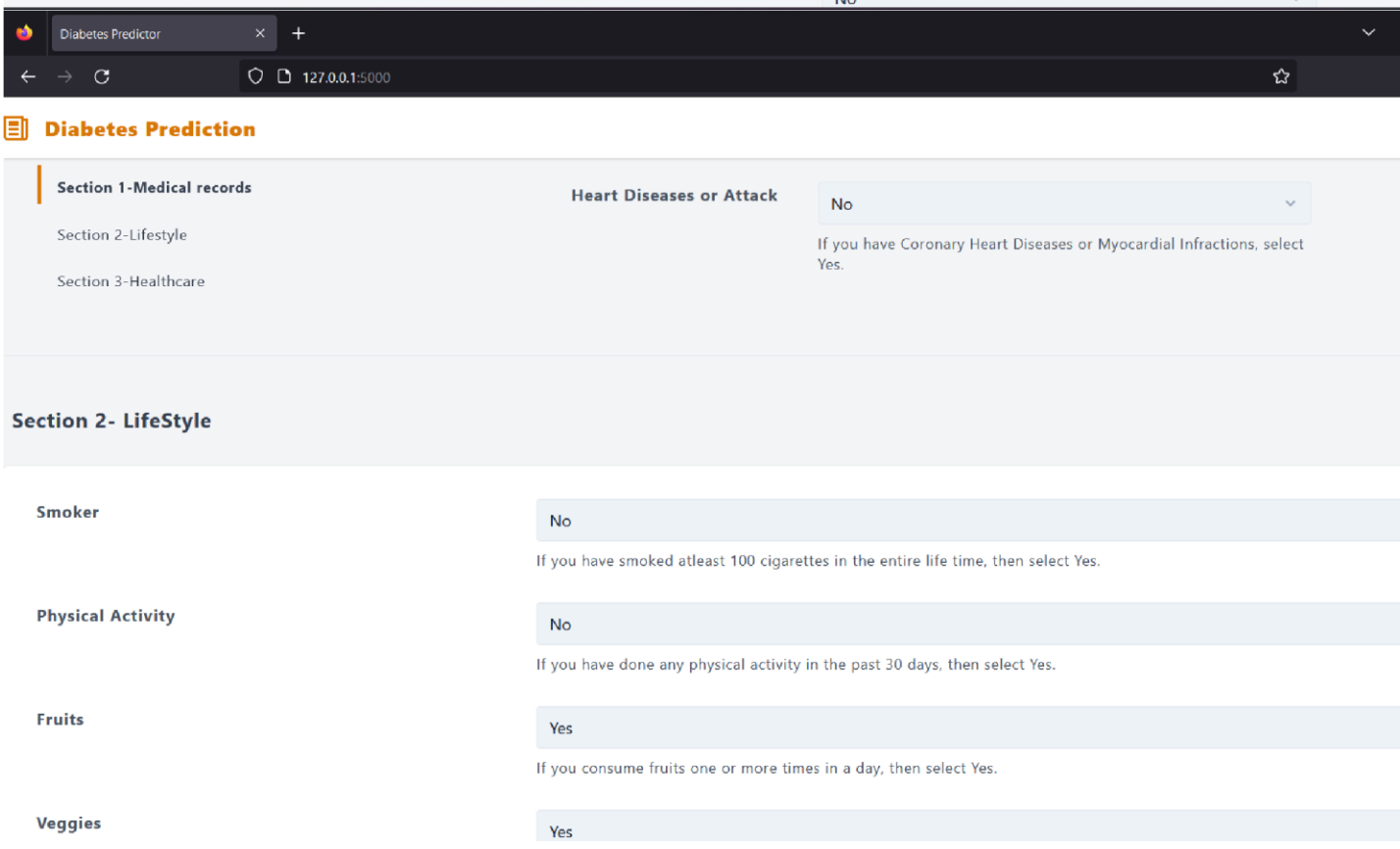
Activity 2.3: Run the web application

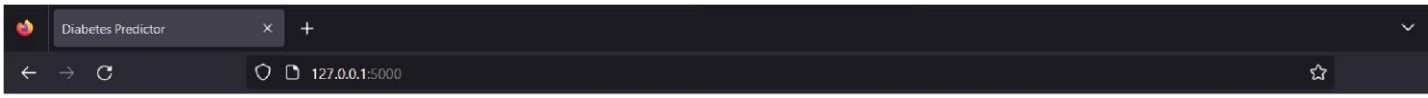
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python ap.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



Now, Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result


Diabetes Predictor
127.0.0.1:5000
Diabetes Prediction
Menu
Section 1-Medical records
Section 2-Lifestyle
Section 3-Healthcare
Section 1- Medical Records
Age
18-24
Sex
Female
High Blood Pressure
No
If you have high BP, that is over 130 mm Hg, then select Yes.
High Cholestrol
No
If you have high Cholestrol, that is over 240mg/dl, then select Yes.
Cholestrol Check
No
If you have done Cholestrol check in this past 5 years, then select Yes.
BMI
22
Enter your BMI.
Stroke
No


Diabetes Predictor
127.0.0.1:5000
Diabetes Prediction
Section 1-Medical records
Section 2-Lifestyle
Section 3-Healthcare
Heart Diseases or Attack
No
If you have Coronary Heart Diseases or Myocardial Infractions, select Yes.
Section 2- LifeStyle
Smoker
No
If you have smoked atleast 100 cigarettes in the entire life time, then select Yes.
Physical Activity
No
If you have done any physical activity in the past 30 days, then select Yes.
Fruits
Yes
If you consume fruits one or more times in a day, then select Yes.
Veggies
Yes



Diabetes Prediction

Veggies

Yes

If you consume vegetables one or more times in a day, then select Yes.

Heavy Alchohol Consumption

No

If you are a person who consumes Alchohol more than 14 drinks per week for men, and 7 drinks per week for women, th

Education level

College 1 year to 3 years (Some college or technical school)

Income

less than $10,000

Section 3- Healthcare

Any Health care

No

If you have any kind of health care coverage, then select Yes.

Diabetes Prediction

If you have any kind of health care coverage, then select Yes.

General Health care

Good

Mental Health care

2

How many days during the past 30 days, your mental health was not good?

Physical Health care

2

How many days during the past 30 days, your physical health was not good?

Walking difficulty

No

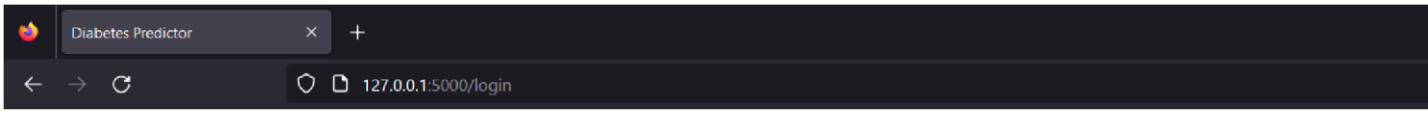
If you face any difficulties while climbing stairs or walking, then select Yes.

No doctor because of cost

No

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

Submit



Diabetes Prediction Result

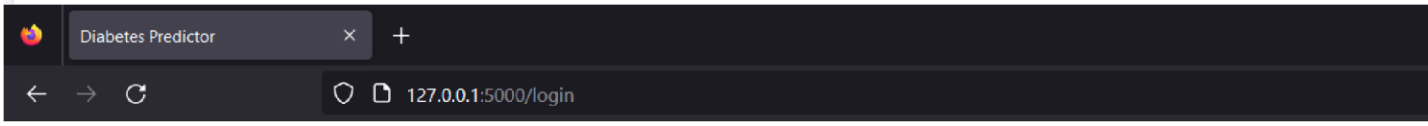
Menu

Result

Recommendations

Result

Good News, Diabetes not present



Diabetes Prediction Result

Recommendations

According to our prediction model, there is currently no indication that you have diabetes. However, it's important to view this as a snapshot, and maintaining a healthy lifestyle is key to ongoing well-being.

To continue minimizing the risk of diabetes, we recommend staying active, eating a balanced diet rich in fruits and vegetables, and maintaining a healthy weight. Regular exercise and a nutritious diet contribute to overall health and well-being.