Project Documentation

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Diabetes Prediction Using Machine Learning

By

Mudunuri Harsha Vardhan Varma(21BRS1634)

Pentapati Meher Baba(21BCE5948)

Konjerla Likhith(21BCE5887)

Challa Sai Phanindra(21BAI1730)

Introduction

Diabetes mellitus is a chronic metabolic disorder that affects millions of individuals worldwide. It is characterized by elevated blood sugar levels resulting from insulin resistance or insufficient insulin production. Early detection and accurate prediction of diabetes are crucial for effective disease management and prevention of complications. In recent years, machine learning techniques have emerged as powerful tools for analyzing large-scale medical data and making accurate predictions. This project aims to develop a robust and reliable machine learning model for diabetes prediction, leveraging a comprehensive approach that incorporates various data sources and advanced algorithms.

In the context of machine learning, we aim to build a model that can accurately predict whether a patient has diabetes based on diagnostic measurements. The dataset we work with includes several medical predictor variables and one target variable (Outcome). **These predictors encompass factors like theBlood Pressure, High cholesterol , Smoker, Stroke, Heavy Alcohol Consumption, body mass index (BMI), Sex, and age.**

Machine learning (ML) has emerged as a powerful tool for predicting various diseases, including diabetes. ML algorithms can analyze vast amounts of data to identify patterns and relationships that may not be readily apparent to humans. These patterns can then be used to develop predictive models that can identify individuals at risk of developing diabetes.

In summary, this project aims to leverage the power of machine learning and comprehensive data integration to develop an accurate and reliable model for diabetes prediction. By combining diverse data sources and utilizing advanced algorithms, this project seeks to make significant strides in early detection and prevention of diabetes, ultimately improving the quality of life for individuals affected by this chronic condition.

A study published in the journal Nature Medicine found that an ML model could predict diabetes with an accuracy of 90%.

LITERATURE SURVEY

A literature survey for "Diabetes Prediction Using Machine Learning" involves reviewing existing research and studies in the field to understand the methodologies, algorithms, and findings. Here's a concise literature survey highlighting key aspects of research in this area:

1. Introduction to Diabetes Prediction:

Diabetes prediction using machine learning has gained substantial attention in recent years due to its potential in early detection and prevention.

Early studies often focused on traditional risk factors, such as age, BMI, and family history.

2. Feature Selection and Extraction:

Many studies emphasize the importance of feature selection and extraction to identify the most relevant predictors for diabetes.

Common features include age, BMI, blood pressure, cholesterol levels, and lifestyle factors.

3. Machine Learning Algorithms:

Researchers have explored various machine learning algorithms for diabetes prediction, including logistic regression, decision trees, support vector machines, and ensemble methods like random forests.

The choice of algorithms often depends on the dataset characteristics and the desired balance between accuracy and interpretability.

4. Integration of Clinical and Genetic Data:

Some studies have integrated clinical data with genetic information to enhance prediction accuracy.

Genetic markers and polymorphisms associated with diabetes risk have been considered in more recent research.

5. Cross-Dataset Validation:

Validating models on diverse datasets is a critical aspect of robust diabetes

prediction research.

Studies often discuss the challenges of dataset heterogeneity and the need for generalized models.

6.Interpretability and Explainability:

As machine learning models become more complex, there is a growing emphasis on interpretability and explainability.

Researchers aim to develop models that not only predict but also provide insights into the factors influencing predictions.

7. Real-world Implementations:

Some studies explore the feasibility of implementing diabetes prediction models in real-world healthcare settings.

Integration with electronic health records and clinical workflows is a topic of interest.

8. Challenges and Future Directions:

Literature often highlights challenges such as data imbalances, interpretability issues, and the need for continuous model adaptation.

Future directions include exploring personalized medicine approaches and addressing ethical considerations.

9. Comparative Analyses:

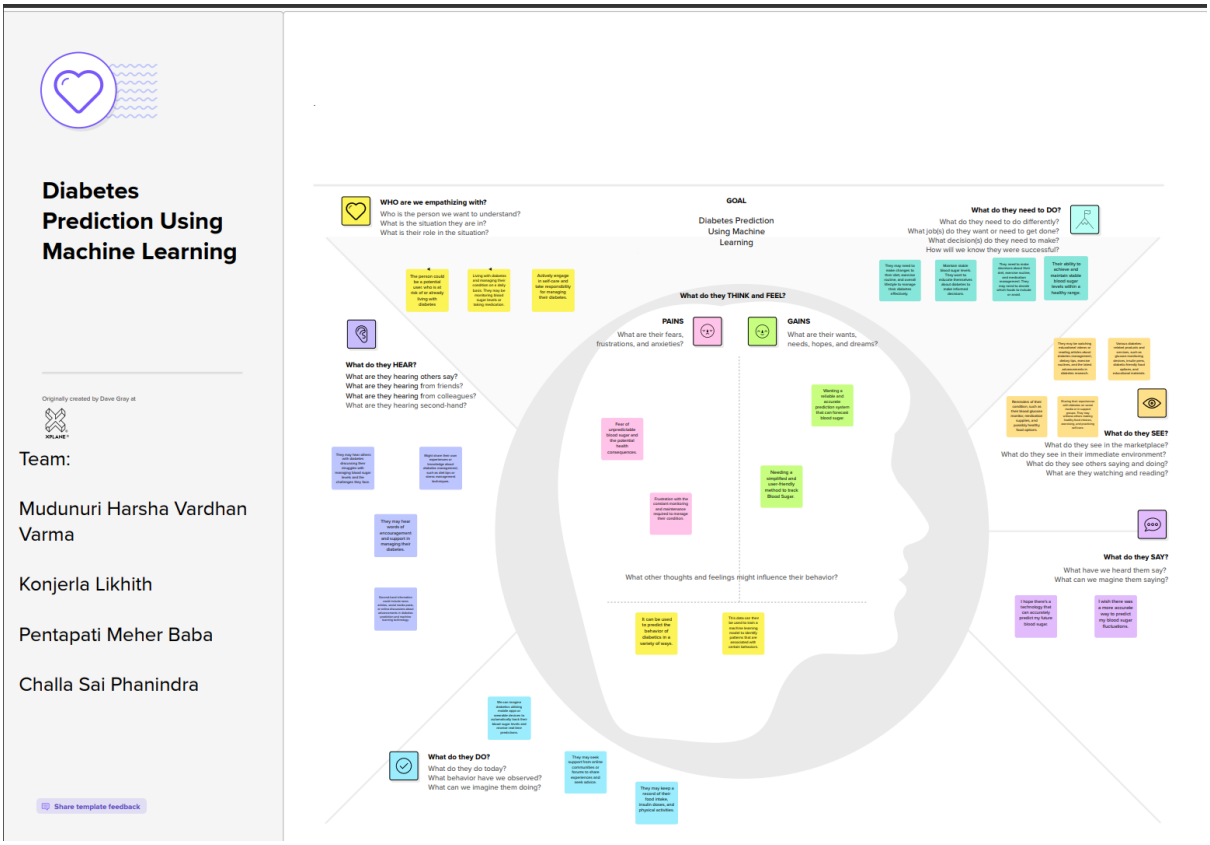
Comparative analyses between different machine learning models and traditional risk assessment tools are common in the literature.

Studies often benchmark the performance of machine learning models against established clinical risk assessment methods.

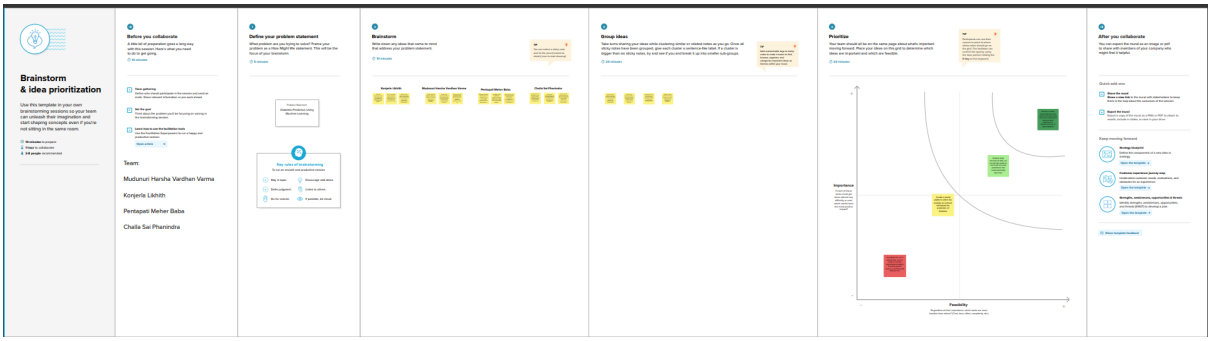
In summary, the literature on diabetes prediction using machine learning reflects a dynamic and evolving field, incorporating a range of algorithms, features, and data sources. The ongoing efforts in improving model accuracy, interpretability, and real-world applicability underscore the potential of machine learning in advancing diabetes prediction and preventive healthcare

IDEATION & PROPOSED SOLUTION

Empathy Map Canvas:



Ideation & Brainstorming:



REQUIREMENT ANALYSIS

Functional Requirements:

Data Collection:

The system should be able to collect and ingest relevant patient data, including age, gender, family history, BMI, blood pressure, and blood sugar levels.

Data Preprocessing:

The system should preprocess the data to handle missing values, outliers, and data normalization.

It should be able to clean and format the data for machine learning model input.

Machine Learning Model:

The system should implement a machine learning model that predicts the likelihood of a person developing diabetes.

The model should be able to learn from historical data to make accurate predictions.

Feature Selection:

The system should employ techniques for feature selection to identify the most relevant attributes for diabetes prediction.

It should allow for manual feature selection or automated methods like feature importance.

Training and Testing:

The system should train the machine learning model on a labeled dataset.It should also support testing the model's accuracy using a separate test dataset.

Prediction:

The system should provide the capability to input patient information and generate a prediction of the likelihood of diabetes development.

User Interface:

The system should have a user-friendly interface for users to interact with and input patient data.

It should display prediction results and visualizations to assist in decision-making.

Data Storage:

The system should store patient data securely and in compliance with data privacy regulations.

Security:

The system should implement security measures to protect patient data and prevent unauthorized access.

Non-Functional Requirements

Accuracy:

The system should achieve a high level of accuracy in diabetes prediction, minimizing false positives and false negatives.

Performance:

The system should provide predictions within a reasonable time frame to ensure timely decision-making.

Scalability:

The system should be able to handle a growing volume of patient data and adapt to changing requirements.

Reliability:

The system should be available and operational with minimal downtime.

Privacy and Compliance:

The system should comply with data protection regulations, such as GDPR or HIPAA, to ensure patient data privacy and security.

Explainability:

The machine learning model should be interpretable and provide explanations for its predictions to build trust with users and healthcare professionals.

User-Friendly Interface:

The user interface should be intuitive and user-friendly to accommodate users with varying levels of technical expertise.

Maintenance:

The system should be maintainable and allow for updates and improvements to the machine learning model as new data becomes available.

Integration:

The system should be capable of integrating with other healthcare systems or electronic health records (EHR) for data exchange and interoperability.

Ethical Considerations:

The system should adhere to ethical guidelines for the use of machine learning in healthcare, including bias mitigation and fair treatment of patients.

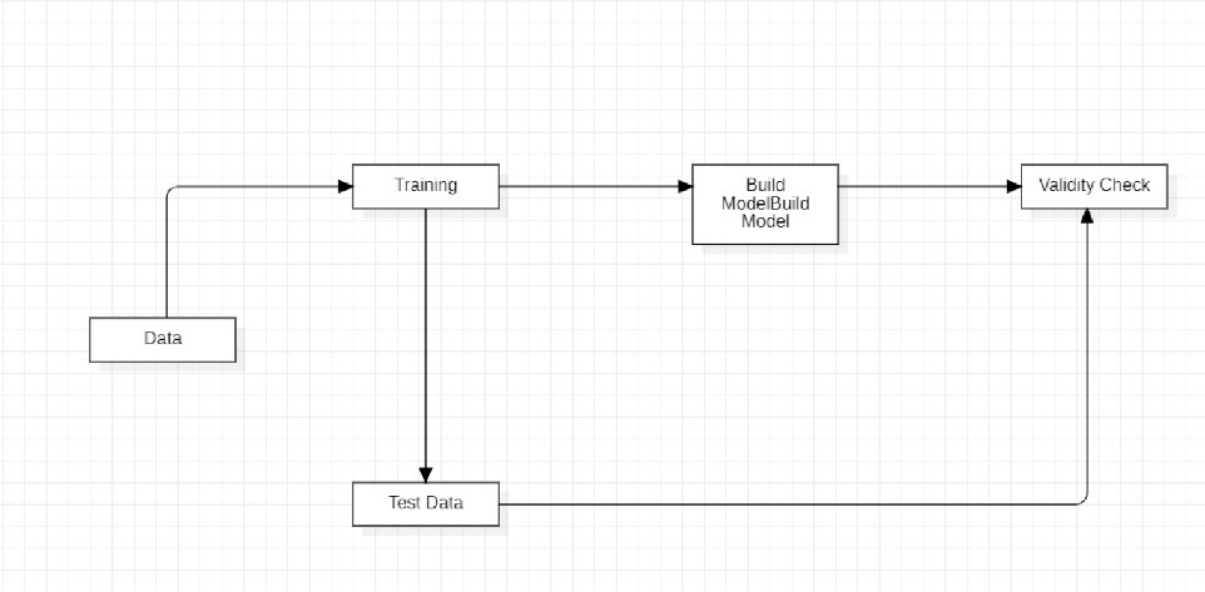
Documentation:

The project should include comprehensive documentation for users, administrators, and developers to facilitate understanding and support

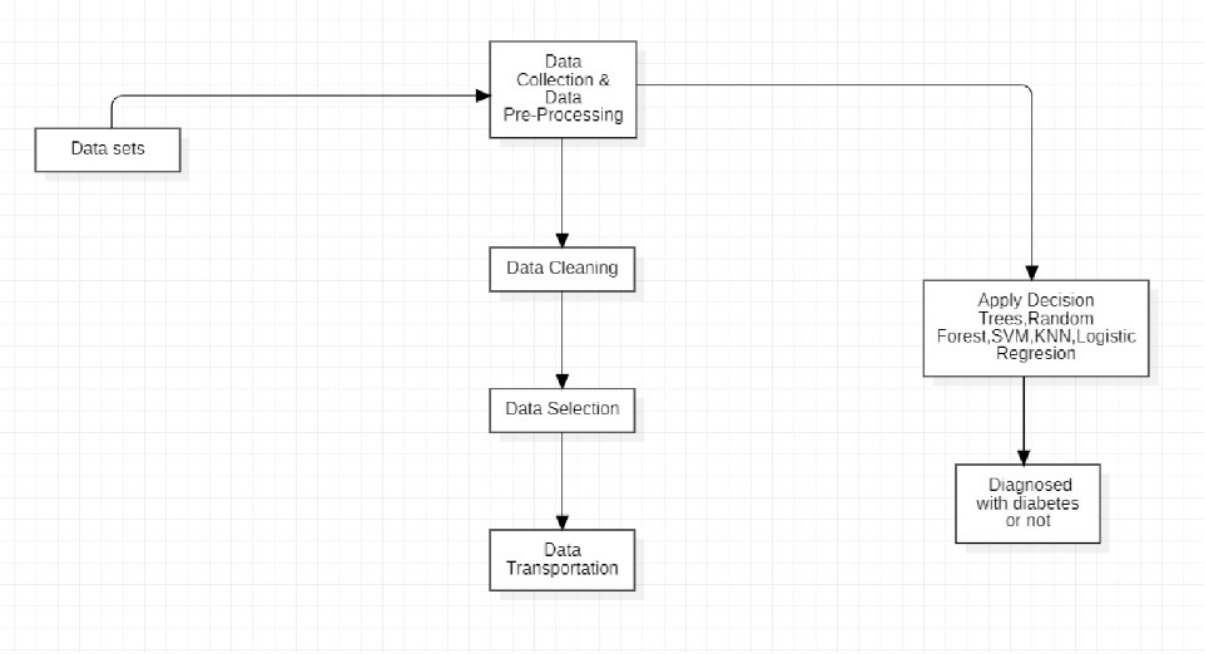
PROJECT DESIGN

Data Flow Diagrams

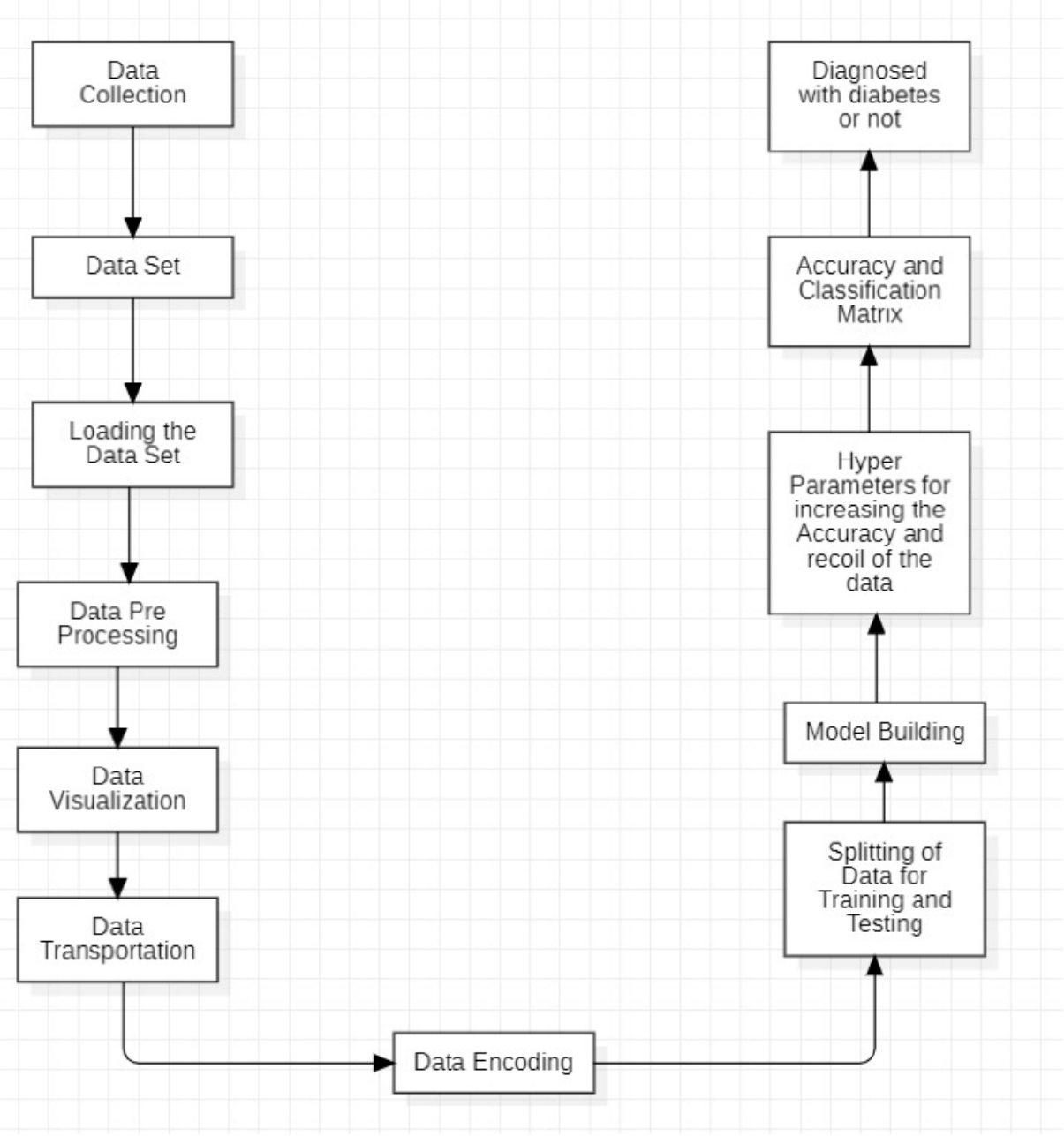
DFD Level-0:



DFD Level-1:



DFD Level 2:


Data Collection
Data Set
Loading the Data Set
Data Pre Processing
Data Visualization
Data Transportation
Data Encoding
Splitting of Data for Training and Testing
Model Building
Hyper Parameters for increasing the Accuracy and recoil of the data
Accuracy and Classification Matrix
Diagnosed with diabetes or not

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration For Mobile	USN-1	Users can register for the application by providing their email, password, and confirming their password.	Users can log in to their account/dash board by entering their email and password.	High	Sprint-1
		USN-2	Upon successful registration, users will receive a confirmation email	Users can receive a confirmation email and click on the confirmation link to verify their account.	High	Sprint-1
		USN-3	Users can also register for the application using their Facebook account.	Users can register for the application using their Google account	High	Sprint-1
	Login	USN-4	Users can log into the application by entering their email and password.		High	Sprint-2

Solution Architecture:

The solution architecture for diabetes prediction using ML models is a multi-step process that aims to develop a robust and scalable system that can analyze various health parameters to predict the likelihood of an individual developing diabetes. The following is a detailed overview of the architecture:

1.Data Collection: To collect relevant health data for diabetes prediction, various sources such as electronic health records, wearables, or patient inputs are used. The data collected includes BMI, blood pressure, glucose levels, family history, and other relevant parameters.

2.Data Preprocessing To ensure data quality and consistency, the collected health data undergoes preprocessing steps such as cleaning, normalization, and feature extraction.

3.Feature Selection: To select the most relevant features for diabetes prediction, statistical techniques or domain knowledge are applied. This step helps reduce noise and focus on the most impactful variables..

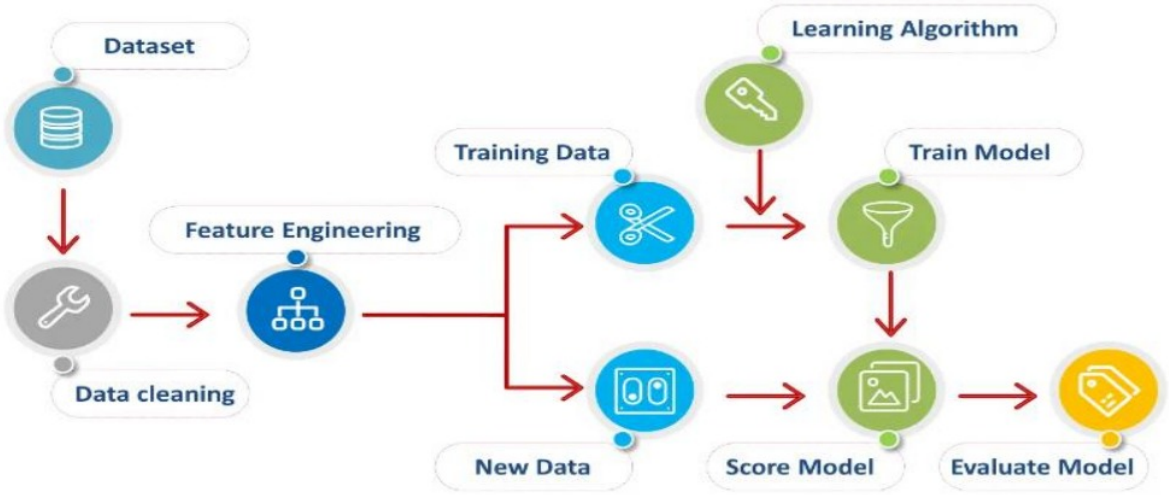
4.Model Training: To train ML models like logistic regression, decision trees, random forests, or support vector machines for diabetes prediction, historical diabetes data is used. The training process involves feeding the pre-processed data to the models and adjusting their parameters to learn patterns and correlations.

5.Model Evaluation: To assess the performance of the trained models and select the most accurate model(s) for predictions, evaluation metrics such as accuracy, precision, recall, and F1 score are commonly used.

6.Predictive Analysis: After selecting the most accurate ML models, they are deployed to predict the likelihood of an individual developing diabetes based on new input data. The models utilize the selected features and their learned patterns to generate predictions with associated probability scores.

7.Integration and Deployment: The prediction system can be integrated into existing healthcare systems or deployed as a standalone application, which can be accessed by healthcare professionals, individuals, or other stakeholders.

8.Continuous Improvement: To ensure the prediction system's performance is continuously monitored and evaluated, the ML models can be retrained to adapt to evolving patterns and improve prediction accuracy as new data becomes available.. The primary objective of the overall process is to deliver an accurate and reliable diabetes prediction system that enables early detection and prevention of the disease. The architecture ensures flexibility, scalability, and adaptability to incorporate advancements in ML techniques and additional health parameters for ongoing enhancement of the system



PROJECT PLANNING &SCHEDULING

Technical Architecture

Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	The application provides a user interface that allows users to sort and filter their test results by column.	Python, Flask
2.	Application Logic-1	We performed unit testing on Logic-1 to ensure that it meets all of its functional requirements.	Python
3.	Application Logic-2	We performed unit testing on Logic-2 to ensure that it meets all of its functional requirements.	Decision trees, Logistic regression
4.	Application Logic-3	We performed unit testing on Logic-3 to ensure that it meets all of its functional requirements.	Random forest classification and SVM
5.	Database	medical records of diabetic patients	Kaggle
6.	Cloud Database	Relational database management system (RDBMS) containing information about diabetic patients, hosted on a cloud platform	Github
7.	File Storage	The capacity and performance requirements for storing files on a computer system	Github
8.	External API-1	The application makes requests to an external API using a RESTful protocol.	Flask
9.	External API-2	The application makes requests to an external API using a RESTful protocol.	Python
10.	Machine Learning Model	A statistical model that has been trained on a large dataset of data to learn patterns and make predictions.	Random forest classification

Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	enumerate the set of open-source frameworks that are used to implement the application's functionality	NumPy,Pandas,Flask
2.	Scalable Architecture	Analyze the scalability characteristics of the proposed architecture (3-tier or microservices) and provide evidence to support your claims	Technology used 2-Tire
3.	Availability	Analyze the availability requirements of the application and design an architecture that can meet those requirements	Flask is used to implement a REST API that allows users to submit their data to the machine learning model.

Sprint Planning & Delivery Schedule:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Preparation	USN-1	To train a machine learning model to predict diabetes, you need to collect and preprocess diabetes-related data.	4	High	Data Scientist (Likhith)
Sprint-2	Model Development	USN-2	To understand the distribution of the data in the dataset and identify any potential issues.	3	High	Data Analyst (HarshaVardhan)
Sprint-3	Model Evaluation	USN-3	To develop and train machine learning models to predict diabetes.	3	Low	MachineLearning Engineer (Meher Baba)
Sprint-4	User Interface and Deployment	USN-4	To develop and train machine learning models to predict diabetes.	4	Medium	Project Manager (Phanindra)
Sprint-5	Testing	USN-5	To deploy the model and user interface to a production environment.	2	High	Developer + Technical Writer (Likhith ,Harsha)
			To create documentation and manuals for projects and products.			

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	4	6 Days	24 Oct 2023	29 Oct 2023	20	29 Oct 2022
Sprint-2	3	3 Days	31 Oct 2023	02 Nov 2023	20	02 Nov 2023
Sprint-3	3	4 Days	03 Nov 2023	6 Nov 2023	20	6 Nov 2023
Sprint-4	4	5 Days	07 Nov 2023	11 Nov 2023	20	11 Nov 2023
Sprint-5	2	6 Days	12 Nov 2023	19 Nov 2023	20	19 Nov 2023

CODING & SOLUTIONING

In a machine learning project for diabetes prediction, We need to define the features (input variables) that our model will use to make predictions about whether an individual is likely to have diabetes. These features should be carefully chosen based on their relevance and availability in the dataset. Based on some common features used in diabetes prediction project.

Age: Age of the individual can be a significant factor in diabetes risk.

Sex: Gender may play a role in diabetes risk, as some studies have shown differences between males and females.

Body Mass Index (BMI): BMI is a measure of body fat based on height and weight and is a strong predictor of diabetes.

Blood Pressure: Systolic and diastolic blood pressure values can be important features.

Cholesterol Levels: Both high total cholesterol and high low-density lipoprotein (LDL) cholesterol levels may be associated with diabetes risk.

Physical Activity: Information about the individual's physical activity or exercise

habits can be relevant.

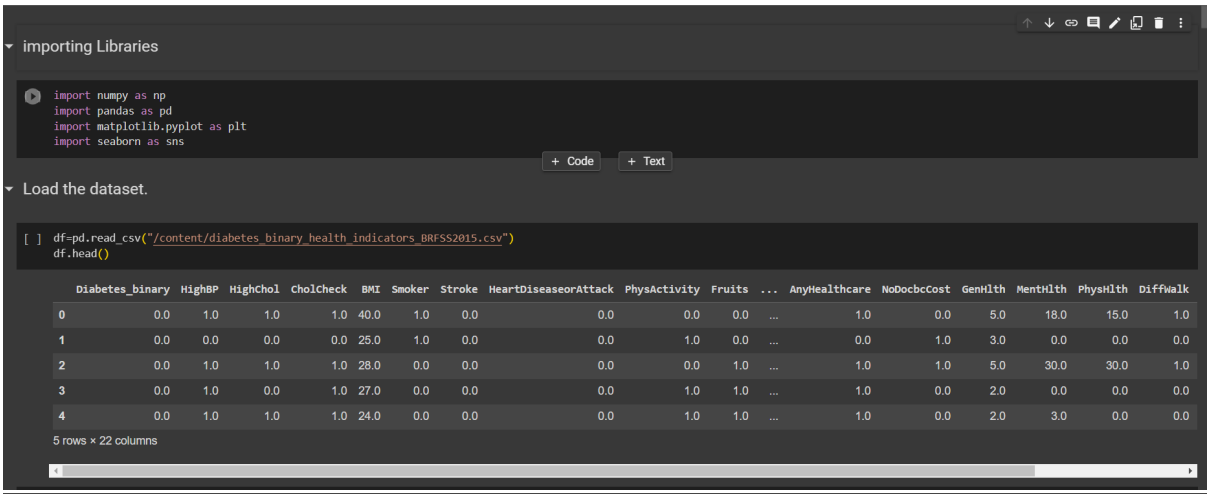
Smoking Status: Smoking can impact diabetes risk, so it's worth including this as a feature.

Alcohol Consumption: Alcohol consumption can also affect diabetes risk and may be included as a feature.

Any Medical Conditions: Presence of other medical conditions, such as polycystic ovary syndrome (PCOS), can be relevant.

Stress Levels: Chronic stress can affect diabetes risk, and this information can be included if available.

PERFORMANCE MATRIX



importing Libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

+ Code + Text

Load the dataset.

```
df=pd.read_csv("/content/diabetes_binary_health_indicators_BRFSS2015.csv")
df.head()
```

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk
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	18.0	15.0	1.0
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	0.0	0.0	0.0
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	30.0	30.0	1.0
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	0.0	0.0	0.0
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	3.0	0.0	0.0

5 rows × 22 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23391 entries, 0 to 23390
Data columns (total 22 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Diabetes_binary       23391 non-null  float64
 1   HighBP                23391 non-null  float64
 2   HighChol              23391 non-null  float64
 3   CholCheck             23391 non-null  float64
 4   BMI                   23391 non-null  float64
 5   Smoker                23391 non-null  float64
 6   Stroke                23391 non-null  float64
 7   HeartDiseaseorAttack  23390 non-null  float64
 8   PhysActivity          23390 non-null  float64
 9   Fruits                23390 non-null  float64
 10  Veggies               23390 non-null  float64
 11  HvyAlcoholConsump     23390 non-null  float64
 12  AnyHealthcare         23390 non-null  float64
 13  NoDocbcCost           23390 non-null  float64
 14  GenHlth               23390 non-null  float64
 15  MentHlth              23390 non-null  float64
 16  PhysHlth              23390 non-null  float64
 17  DiffWalk              23390 non-null  float64
 18  Sex                   23390 non-null  float64
 19  Age                   23390 non-null  float64
 20  Education             23390 non-null  float64
 21  Income                23390 non-null  float64
dtypes: float64(22)
memory usage: 3.9 MB
```

```
[ ] df.shape
```

```
(23391, 22)
```

Checking null values

```
df.isnull().sum()
```

```
Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    1
PhysActivity            1
Fruits                  1
Veggies                 1
HvyAlcoholConsump       1
AnyHealthcare           1
NoDocbcCost             1
GenHlth                 1
MentHlth                1
PhysHlth                1
DiffWalk                1
Sex                     1
Age                     1
Education               1
Income                  1
dtype: int64
```

```
df.describe()
```

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCo
count	23391.000000	23391.000000	23391.000000	23391.000000	23391.000000	23391.000000	23391.000000	23390.000000	23390.000000	23390.000000	...	23390.000000	23390.0000
mean	0.143816	0.425164	0.420589	0.961994	28.129152	0.431747	0.042751	0.091791	0.768063	0.627148	...	0.943309	0.0967
std	0.350911	0.494378	0.493664	0.191215	6.222746	0.495330	0.202301	0.288737	0.422078	0.483573	...	0.231256	0.2955
min	0.000000	0.000000	0.000000	0.000000	14.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.0000
25%	0.000000	0.000000	0.000000	1.000000	24.000000	0.000000	0.000000	0.000000	1.000000	0.000000	...	1.000000	0.0000
50%	0.000000	0.000000	0.000000	1.000000	27.000000	0.000000	0.000000	0.000000	1.000000	1.000000	...	1.000000	0.0000
75%	0.000000	1.000000	1.000000	1.000000	31.000000	1.000000	0.000000	0.000000	1.000000	1.000000	...	1.000000	0.0000
max	1.000000	1.000000	1.000000	1.000000	92.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.0000

8 rows × 22 columns

```
df['HighChol'].fillna(df['HighChol'].median(),inplace =True)
df['CholCheck'].fillna(df['CholCheck'].median(),inplace =True)
df['BMI'].fillna(df['BMI'].median(),inplace =True)
df['Smoker'].fillna(df['Smoker'].median(),inplace =True)
df['Stroke'].fillna(df['Stroke'].median(),inplace =True)
df['HeartDiseaseorAttack'].fillna(df['HeartDiseaseorAttack'].median(),inplace =True)
df['PhysActivity'].fillna(df['PhysActivity'].median(),inplace =True)
df['Fruits'].fillna(df['Fruits'].median(),inplace =True)
df['Veggies'].fillna(df['Veggies'].median(),inplace =True)
df['HvyAlcoholConsump'].fillna(df['HvyAlcoholConsump'].median(),inplace =True)
df['AnyHealthcare'].fillna(df['AnyHealthcare'].median(),inplace =True)
df['NoDocbcCost'].fillna(df['NoDocbcCost'].median(),inplace =True)
df['GenHlth'].fillna(df['GenHlth'].median(),inplace =True)
df['MentHlth'].fillna(df['MentHlth'].median(),inplace =True)
df['PhysHlth'].fillna(df['PhysHlth'].median(),inplace =True)
df['DiffWalk'].fillna(df['DiffWalk'].median(),inplace =True)
df['Sex'].fillna(df['Sex'].median(),inplace =True)
df['Age'].fillna(df['Age'].median(),inplace =True)
df['Education'].fillna(df['Education'].median(),inplace =True)
df['Income'].fillna(df['Income'].median(),inplace =True)
```

```
df.isnull().sum()
```

```
Diabetes_binary         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
```

Data Visualizations

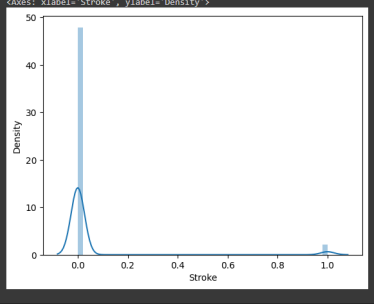
Univariate Analysis

```
df['Stroke'].value_counts()
```

```
0.0    22391
1.0     1000
Name: Stroke, dtype: int64
```

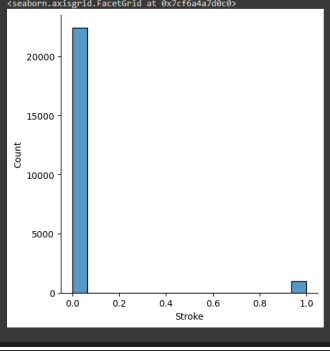
```
sns.distplot(df['Stroke'])
```

```
<ipython-input-10-ff0f33a28e05>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['Stroke'])
<Axes: xlabel='Stroke', ylabel='Density'>
```



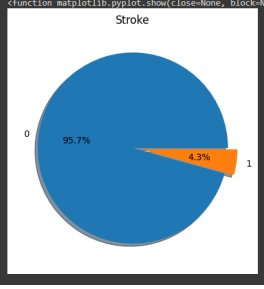

```
[ ] sns.displot(df['Stroke'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cf6a4a7d0c0>
```



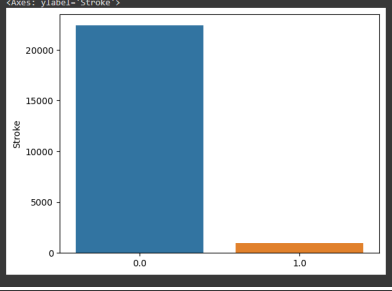

```
plt.pie(df['Stroke'].value_counts(),[0,0.1],labels=[0,1],autopct='%1.1f%%',shadow=True)
plt.title('Stroke')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



```
sns.barplot(x=df['Stroke'].value_counts().index,y=df['Stroke'].value_counts())
```

```
<Axes: ylabel='Stroke'>
```

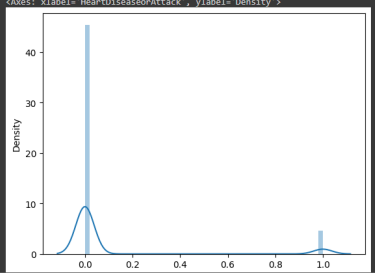



```
[ ] df['HeartDiseaseorAttack'].value_counts()
```

```
0.0    21244
1.0     2147
Name: HeartDiseaseorAttack, dtype: int64
```

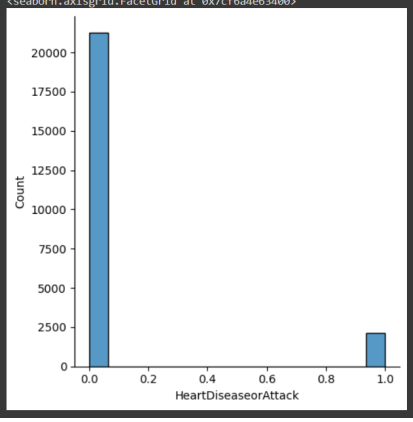
```
sns.distplot(df['HeartDiseaseorAttack'])
```

```
<ipython-input-15-77665ce3a4c6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['HeartDiseaseorAttack'])
<Axes: xlabel='HeartDiseaseorAttack', ylabel='Density'>
```



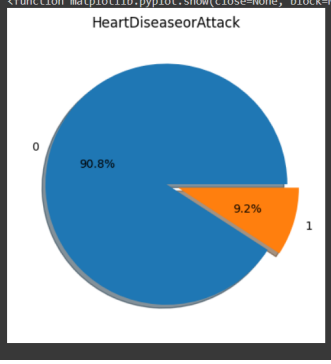

```
sns.displot(df['HeartDiseaseorAttack'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cf6a4e63400>
```



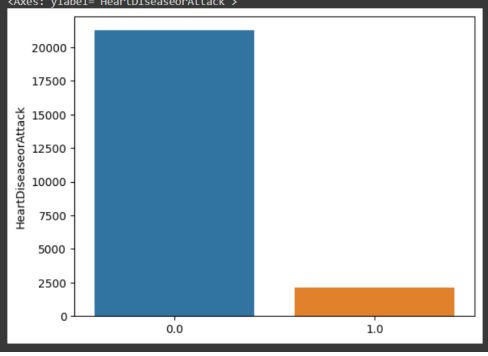
```
plt.pie(df['HeartDiseaseorAttack'].value_counts(),[0,0.1],labels=[0,1],autopct='%1.1f%%',shadow=True)
plt.title('HeartDiseaseorAttack')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
sns.barplot(x=df['HeartDiseaseorAttack'].value_counts().index,y=df['HeartDiseaseorAttack'].value_counts())
```

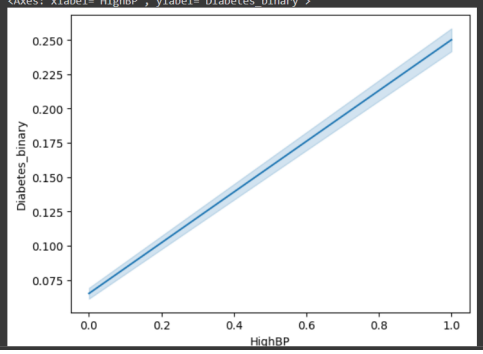
```
<Axes: ylabel='HeartDiseaseorAttack'>
```



Bivariate analysis

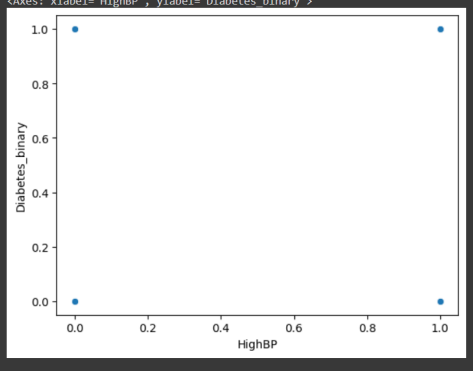
```
sns.lineplot(x=df['HighBP'],y=df['Diabetes_binary'])
```

```
<Axes: xlabel='HighBP', ylabel='Diabetes_binary'>
```



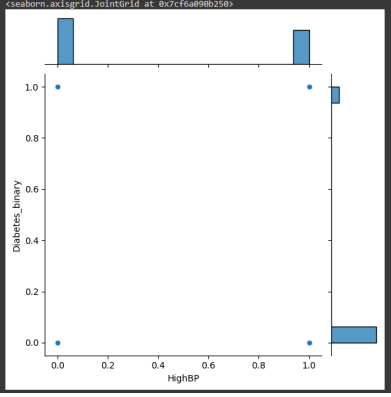
```
sns.scatterplot(x=df['HighBP'],y=df['Diabetes_binary'])
```

```
<Axes: xlabel='HighBP', ylabel='Diabetes_binary'>
```



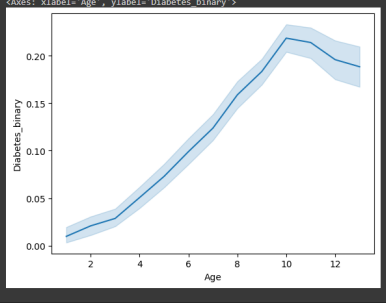

```
sns.jointplot(x='HighBP' ,y='Diabetes_binary',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7cf6a090b250>
```



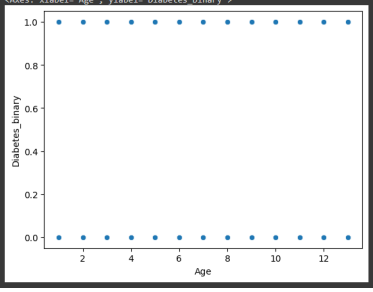

```
sns.lineplot(x=df['Age'],y=df['Diabetes_binary'])
```

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



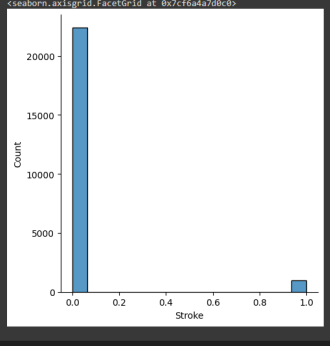

```
[ ] sns.scatterplot(x=df['Age'],y=df['Diabetes_binary'])
```

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



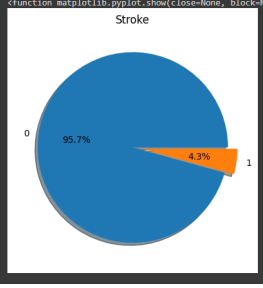
```
[ ] sns.displot(df['Stroke'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cf6a4a7d0c0>
```



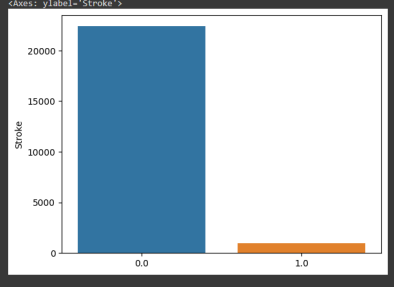

```
plt.pie(df['Stroke'].value_counts(),[0,0.1],labels=[0,1],autopct='%1.1f%%',shadow=True)
plt.title('Stroke')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
sns.barplot(x=df['Stroke'].value_counts().index,y=df['Stroke'].value_counts())
```

```
<Axes: ylabel='Stroke'>
```

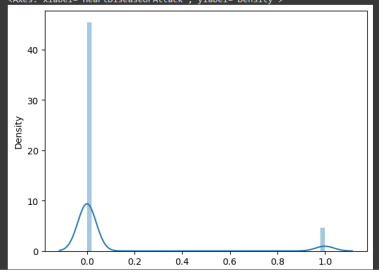



```
[ ] df['HeartDiseaseorAttack'].value_counts()
```

```
0.0    21244
1.0     2147
Name: HeartDiseaseorAttack, dtype: int64
```

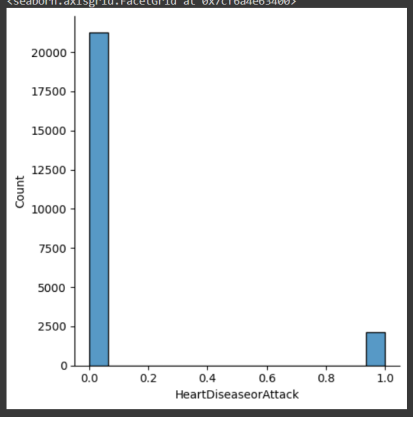
```
sns.distplot(df['HeartDiseaseorAttack'])
```

```
<ipython-input-15-77665ce3a4c6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['HeartDiseaseorAttack'])
<Axes: xlabel='HeartDiseaseorAttack', ylabel='Density'>
```



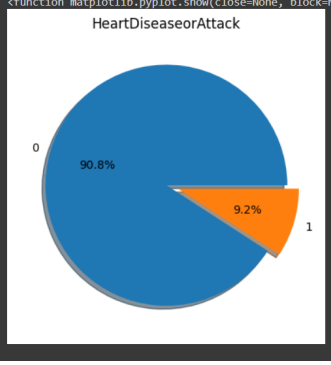
```
sns.displot(df['HeartDiseaseorAttack'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cf6a4e63400>
```



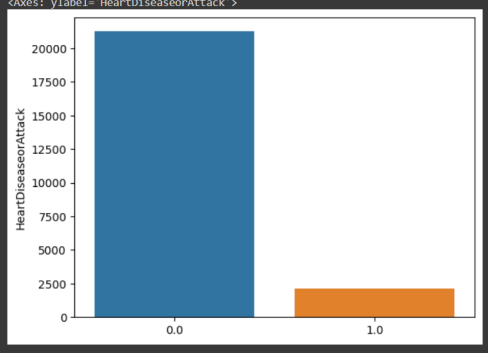

```
plt.pie(df['HeartDiseaseorAttack'].value_counts(),[0,0.1],labels=[0,1],autopct='%1.1f%%',shadow=True)
plt.title('HeartDiseaseorAttack')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
sns.barplot(x=df['HeartDiseaseorAttack'].value_counts().index,y=df['HeartDiseaseorAttack'].value_counts())
```

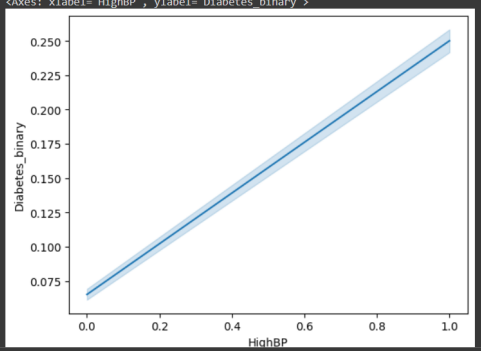
```
<Axes: ylabel='HeartDiseaseorAttack'>
```



Bivariate analysis

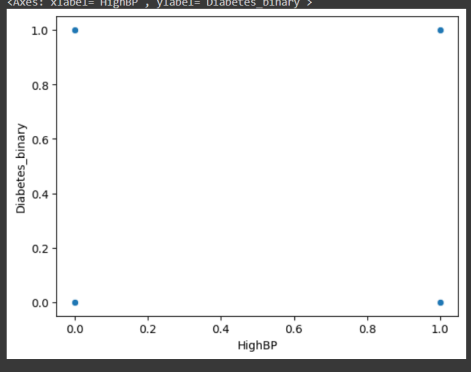
```
sns.lineplot(x=df['HighBP'],y=df['Diabetes_binary'])
```

```
<Axes: xlabel='HighBP', ylabel='Diabetes_binary'>
```



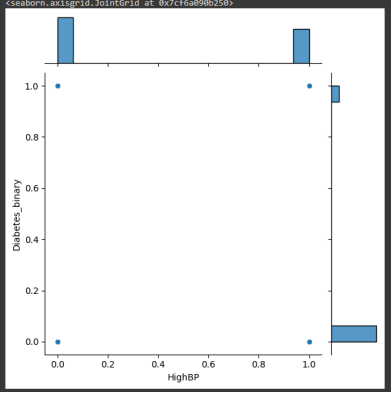

```
sns.scatterplot(x=df['HighBP'],y=df['Diabetes_binary'])
```

```
<Axes: xlabel='HighBP', ylabel='Diabetes_binary'>
```



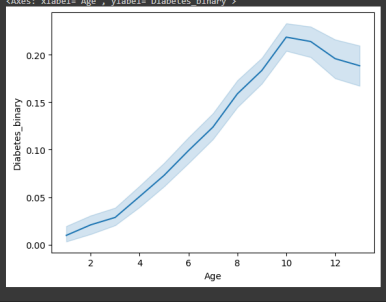

```
sns.jointplot(x='HighBP' ,y='Diabetes_binary',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7cf6a090b250>
```



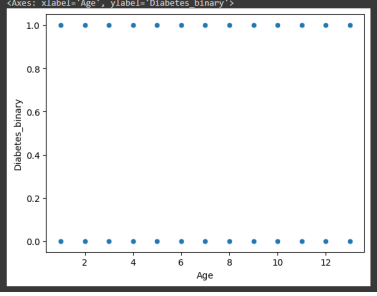

```
sns.lineplot(x=df['Age'],y=df['Diabetes_binary'])
```

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



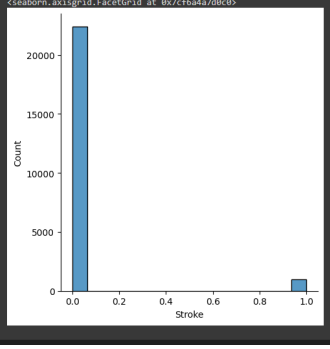
```
[ ] sns.scatterplot(x=df['Age'],y=df['Diabetes_binary'])
```

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



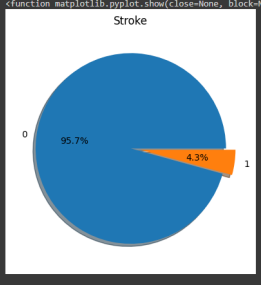

```
[ ] sns.displot(df['Stroke'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cf6a4a7d0c0>
```



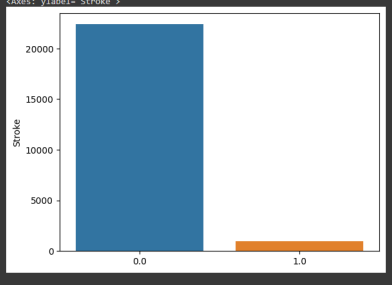

```
plt.pie(df['Stroke'].value_counts(),[0,0.1],labels=[0,1],autopct='%1.1f%%',shadow=True)
plt.title('Stroke')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```




```
sns.barplot(x=df['Stroke'].value_counts().index,y=df['Stroke'].value_counts())
```

```
<Axes: ylabel='Stroke'>
```

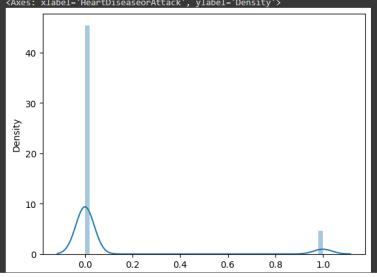


```
[ ] df['HeartDiseaseorAttack'].value_counts()
```

```
0.0    21244
1.0     2147
Name: HeartDiseaseorAttack, dtype: int64
```

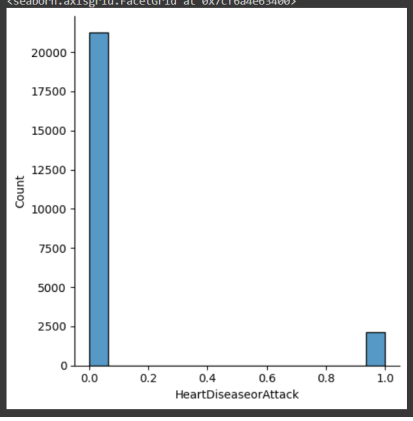
```
sns.distplot(df['HeartDiseaseorAttack'])
```

```
<ipython-input-15-7766Sce3a4c6>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df['HeartDiseaseorAttack'])
<Axes: xlabel='HeartDiseaseorAttack', ylabel='Density'>
```



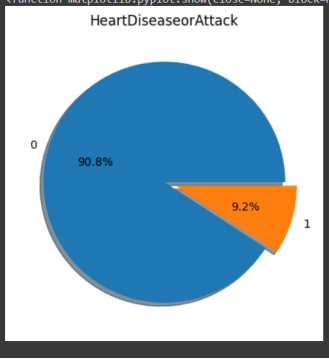

```
sns.displot(df['HeartDiseaseorAttack'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7cf6a4e63400>
```



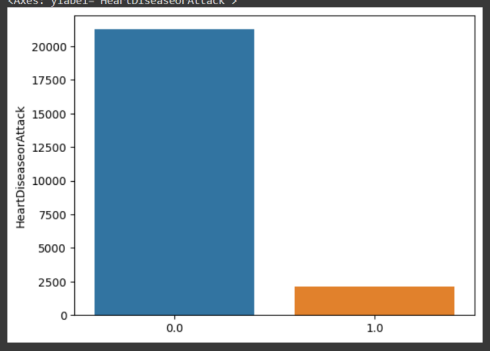

```
plt.pie(df['HeartDiseaseorAttack'].value_counts(),[0,0.1],labels=[0,1],autopct='%1.1f%%',shadow=True)
plt.title('HeartDiseaseorAttack')
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



```
sns.barplot(x=df['HeartDiseaseorAttack'].value_counts().index,y=df['HeartDiseaseorAttack'].value_counts())
```

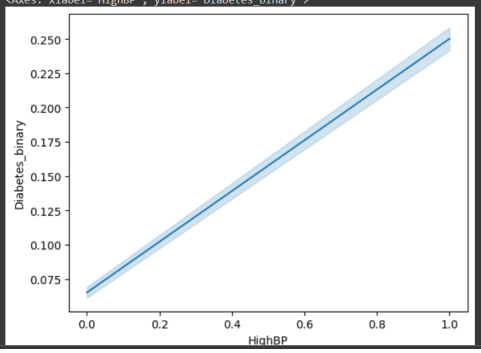
```
<Axes: ylabel='HeartDiseaseorAttack'>
```



Bivariate analysis

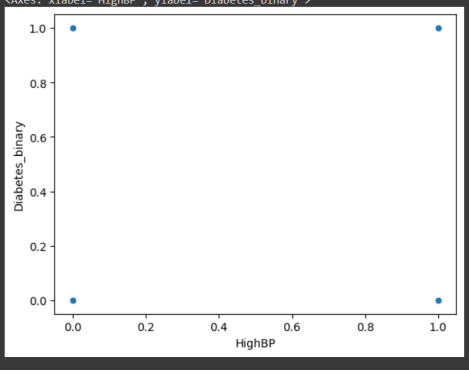
```
sns.lineplot(x=df['HighBP'],y=df['Diabetes_binary'])
```

```
<Axes: xlabel='HighBP', ylabel='Diabetes_binary'>
```



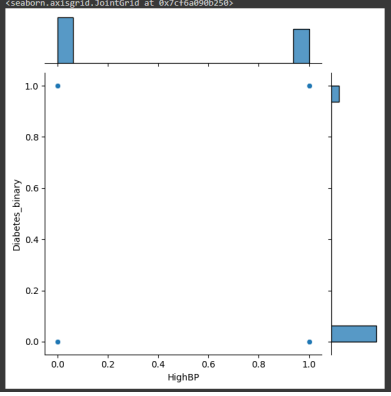

```
sns.scatterplot(x=df['HighBP'],y=df['Diabetes_binary'])
```

```
<Axes: xlabel='HighBP', ylabel='Diabetes_binary'>
```



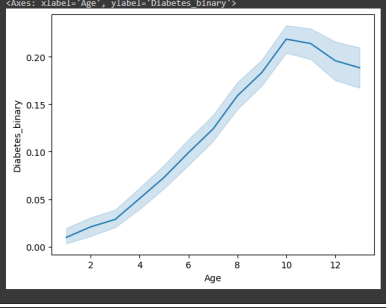
```
sns.jointplot(x='HighBP' ,y='Diabetes_binary',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7cf6a090b250>
```



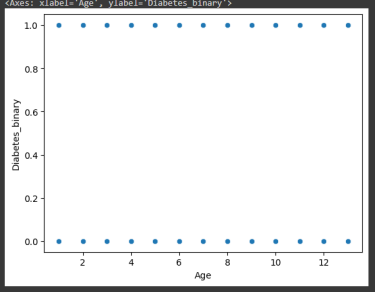

```
sns.lineplot(x=df['Age'],y=df['Diabetes_binary'])
```

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



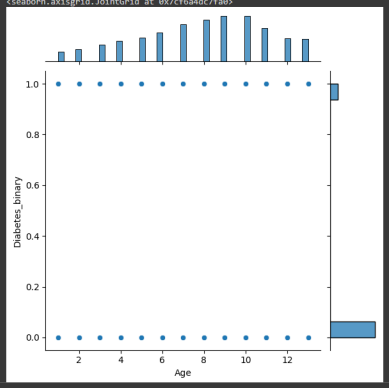

```
[ ] sns.scatterplot(x=df['Age'],y=df['Diabetes_binary'])
```

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



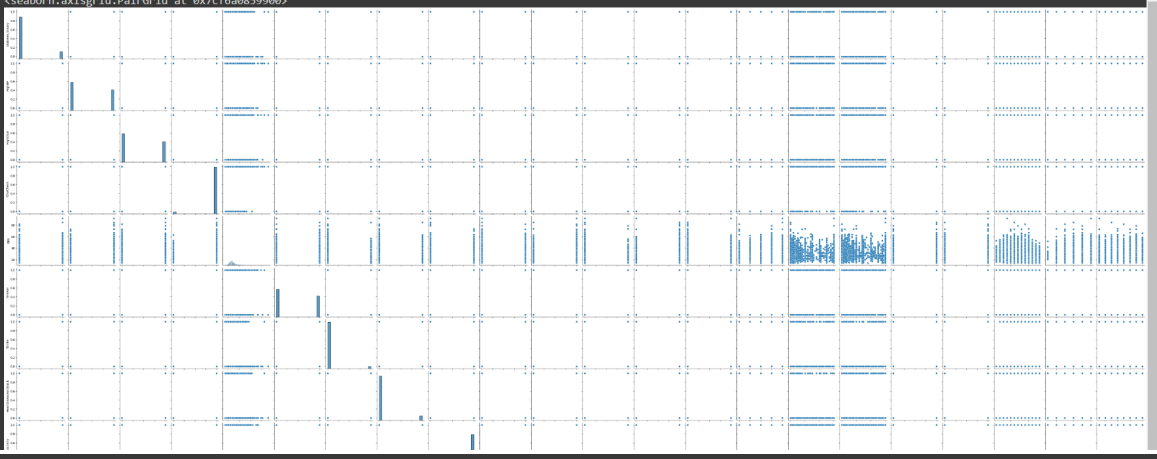

```
sns.jointplot(x='Age' ,y='Diabetes_binary',data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x7cf6a4dc7fa0>
```

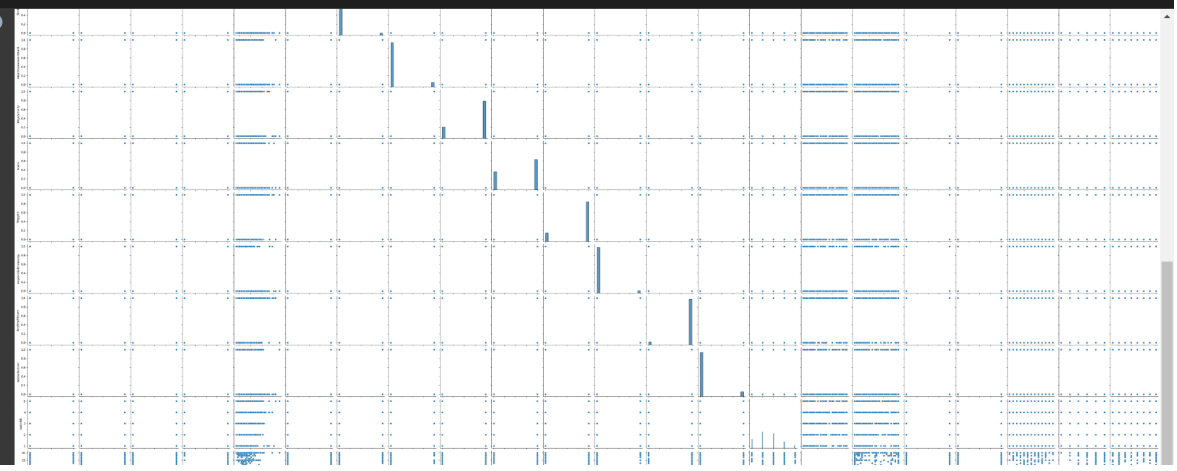


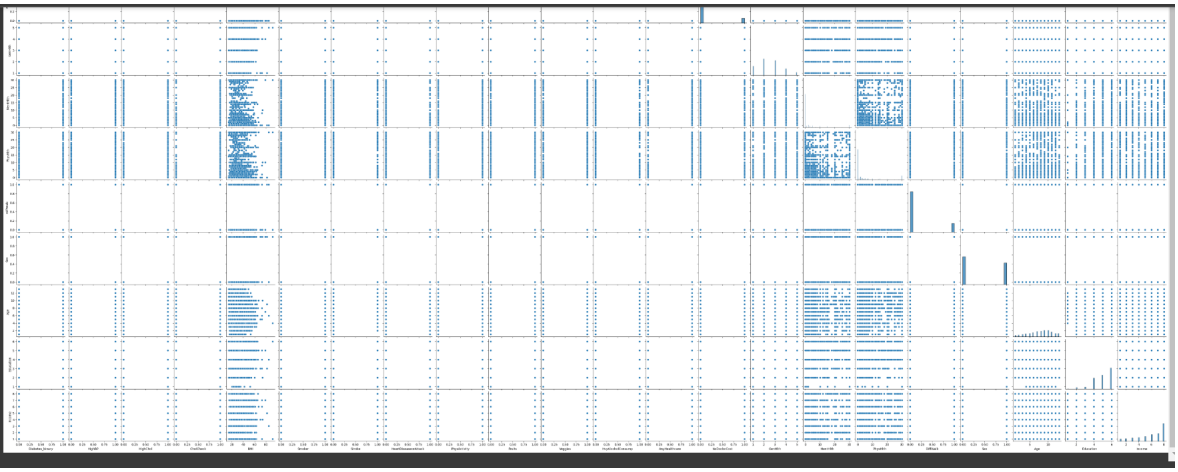
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7cf6a0859900>
```

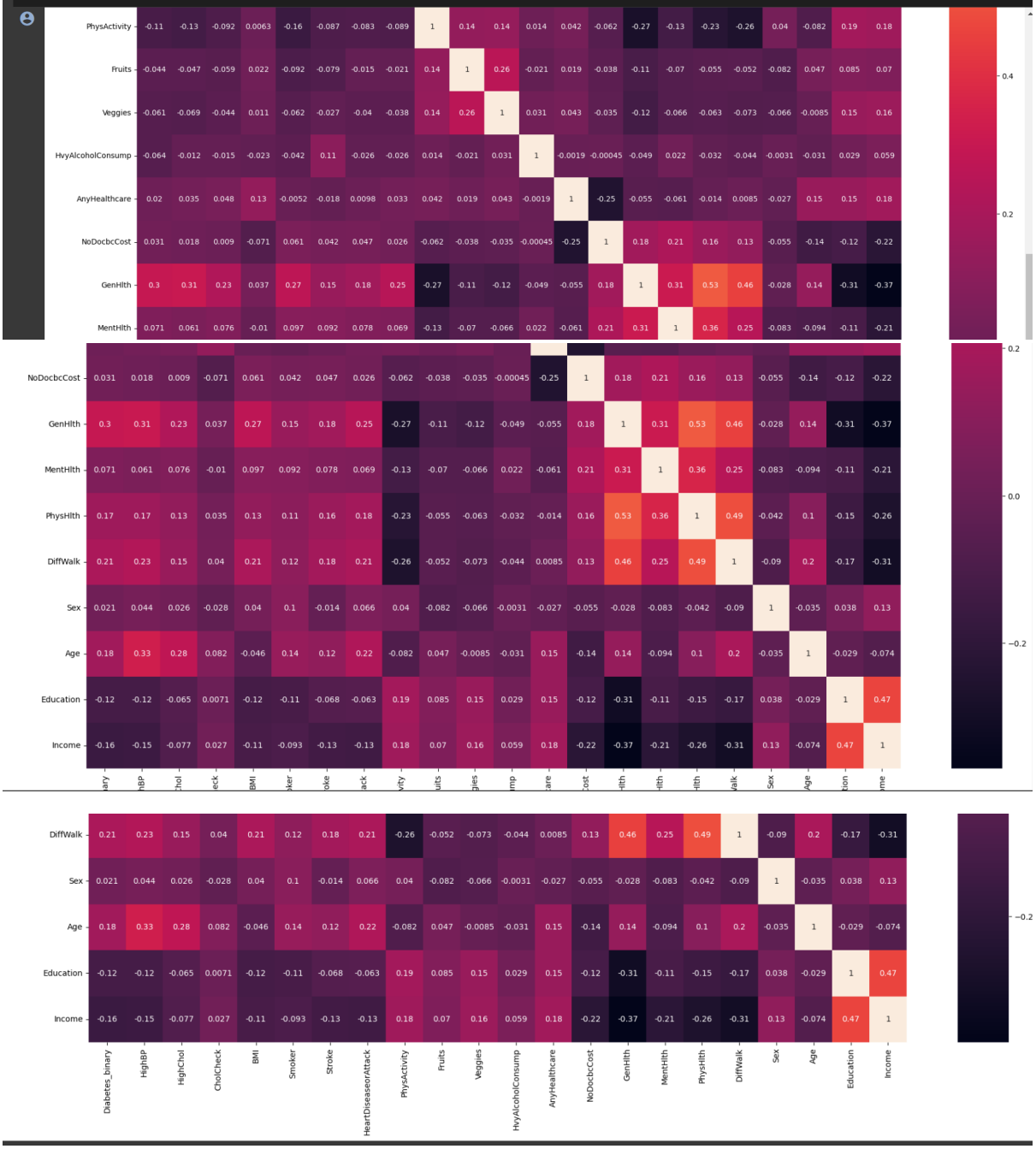


```
sns.pairplot(df)
```





```
plt.subplots(figsize= (23,23))
sns.heatmap(df.corr(),annot=True)
```



outlier detection

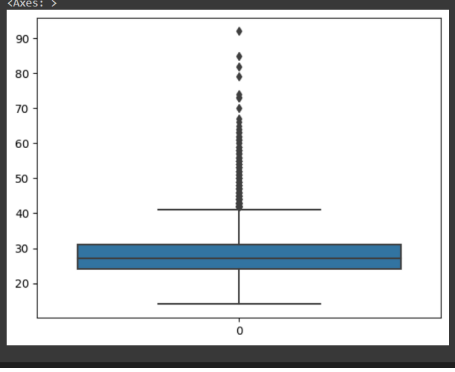
```
df.head()
```

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk
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	18.0	15.0	1.0
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	0.0	0.0	0.0
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	30.0	30.0	1.0
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	0.0	0.0	0.0
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	3.0	0.0	0.0

5 rows × 22 columns

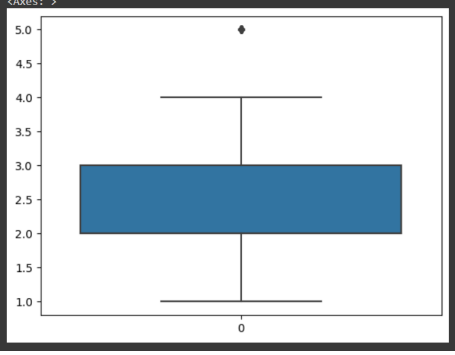
```
sns.boxplot(df.BMI)
```

```
<Axes: >
```



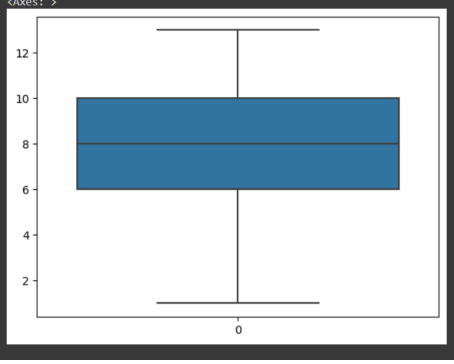

```
sns.boxplot(df.GenHlth)
```

```
<Axes: >
```



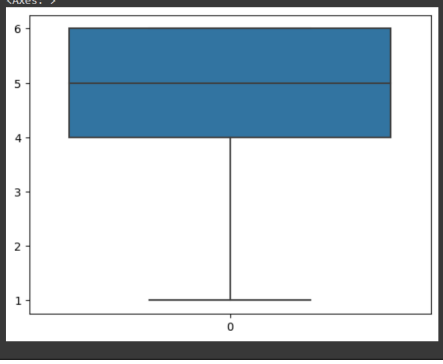

```
[ ] sns.boxplot(df.Age)
```

```
<Axes: >
```



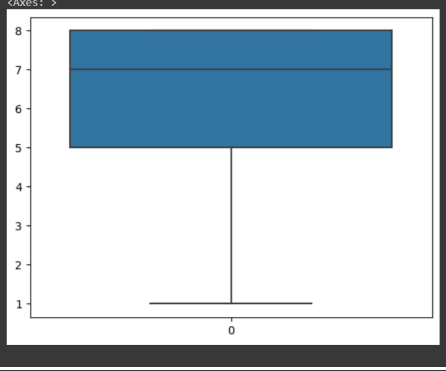

```
sns.boxplot(df.Education)
```

```
<Axes: >
```



```
sns.boxplot(df.Income)
```

```
<Axes: >
```



Outlier Replacement

```
df.median()
```

```
Diabetes_binary         0.0
HighBP                  0.0
HighChol                0.0
CholCheck               1.0
BMI                    27.0
Smoker                  0.0
Stroke                  0.0
HeartDiseaseorAttack    0.0
PhysActivity            1.0
Fruits                  1.0
Veggies                 1.0
HvyAlcoholConsump       0.0
AnyHealthcare           1.0
NoDocbcCost             0.0
GenHlth                 2.0
MentHlth                0.0
PhysHlth                0.0
DiffWalk                0.0
Sex                     0.0
Age                     8.0
Education               5.0
Income                  7.0
dtype: float64
```

```
[ ] q1 = df.BMI.quantile(0.25) #Q1
    q3 = df.BMI.quantile(0.75) #Q3
```

```
[ ] print(q1)
    print(q3)
```

```
24.0
31.0
```

+ Code + Text

```
[ ] IQR=q3-q1
    IQR
```

```
7.0
```

```
[ ] upper_limit = q3+1.5*IQR
    upper_limit
```

```
41.5
```

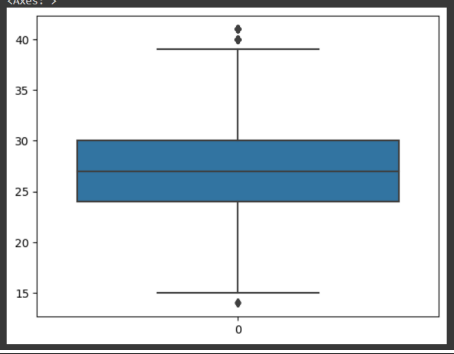
```
[ ] lower_limit = q1-1.5*IQR
    lower_limit
```

```
13.5
```

```
[ ] df['BMI'] = np.where(df['BMI']>upper_limit,27.0,df['BMI'])
    df['BMI'] = np.where(df['BMI']<lower_limit,27.0,df['BMI'])
```

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

```
<Axes: >
```




```
[ ] q1 = df.GenHlth.quantile(0.25) #Q1
    q3 = df.GenHlth.quantile(0.75) #Q3
```

```
[ ] print(q1)
    print(q3)
```

```
2.0
3.0
```

```
[ ] IQR=q3-q1
    IQR
```

```
1.0
```

```
[ ] upper_limit = q3+1.5*IQR
    upper_limit
```

```
4.5
```

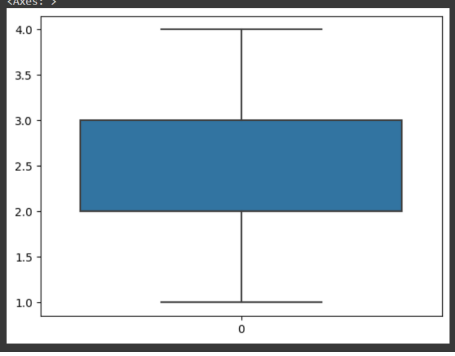
```
[ ] lower_limit = q1-1.5*IQR
    lower_limit
```

```
0.5
```

```
[ ] df['GenHlth'] = np.where(df['GenHlth']>upper_limit,2.0,df['GenHlth'])
    # df['GenHlth'] = np.where(df['GenHlth']<lower_limit,2.0,df['GenHlth'])
```

```
sns.boxplot(df.GenHlth)
```

```
<Axes: >
```



x and y Split

```
x=df.drop(columns=['Diabetes_binary'],axis=1)
x.head()
```

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Ag
0	1.0	1.0	1.0	40.0	1.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	2.0	18.0	15.0	1.0	0.0	9.
1	0.0	0.0	0.0	25.0	1.0	0.0	0.0	1.0	0.0	0.0	...	0.0	1.0	3.0	0.0	0.0	0.0	0.0	7.
2	1.0	1.0	1.0	28.0	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	1.0	2.0	30.0	30.0	1.0	0.0	9.
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	0.0	0.0	11.
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	3.0	0.0	0.0	0.0	11.

5 rows × 21 columns

+ Code + Text

```
y=df['Diabetes_binary']
y.head()
```

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

```
from sklearn.preprocessing import MinMaxScaler
scale =MinMaxScaler()
```

```
x_scaled=pd.DataFrame(scale.fit_transform(x),columns=x.columns)
x_scaled.head()
```

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex
0	1.0	1.0	1.0	0.962963	1.0	0.0	0.0	0.0	0.0	1.0	...	1.0	0.0	0.333333	0.6	0.5	1.0	0.0
1	0.0	0.0	0.0	0.407407	1.0	0.0	0.0	1.0	0.0	0.0	...	0.0	1.0	0.666667	0.0	0.0	0.0	0.0
2	1.0	1.0	1.0	0.518519	0.0	0.0	0.0	0.0	1.0	0.0	...	1.0	1.0	0.333333	1.0	1.0	1.0	0.0
3	1.0	0.0	1.0	0.481481	0.0	0.0	0.0	1.0	1.0	1.0	...	1.0	0.0	0.333333	0.0	0.0	0.0	0.0
4	1.0	1.0	1.0	0.370370	0.0	0.0	0.0	1.0	1.0	1.0	...	1.0	0.0	0.333333	0.1	0.0	0.0	0.0

5 rows × 21 columns

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x_scaled,y,test_size=0.3,random_state=0)
```

```
x_train.head()
```

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk
10701	1.0	0.0	1.0	1.000000	1.0	0.0	1.0	1.0	0.0	1.0	...	1.0	0.0	1.000000	0.333333	0.833333	1.0
3249	1.0	0.0	1.0	0.666667	0.0	0.0	0.0	1.0	1.0	1.0	...	1.0	0.0	0.333333	0.100000	0.000000	0.0
7743	1.0	1.0	1.0	0.259259	1.0	1.0	1.0	1.0	1.0	1.0	...	1.0	0.0	1.000000	0.033333	0.466667	0.0
1290	0.0	0.0	0.0	0.259259	0.0	0.0	0.0	1.0	1.0	1.0	...	1.0	0.0	0.000000	0.000000	0.000000	0.0
5930	0.0	1.0	1.0	0.518519	0.0	0.0	1.0	1.0	1.0	1.0	...	1.0	0.0	1.000000	0.500000	0.500000	1.0

5 rows × 21 columns

+ Code + Text

```
x_train.shape
```

```
(16373, 21)
```

```
x_test.head()
```

	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk
3633	0.0	0.0	1.0	0.407407	1.0	0.0	0.0	1.0	1.0	1.0	...	1.0	0.0	0.333333	0.000000	0.233333	0.0
19920	1.0	0.0	1.0	0.444444	0.0	0.0	0.0	1.0	1.0	1.0	...	1.0	0.0	0.333333	0.166667	0.000000	0.0
7767	1.0	0.0	1.0	0.296296	0.0	0.0	0.0	0.0	1.0	1.0	...	1.0	0.0	1.000000	0.033333	0.000000	0.0
14063	0.0	0.0	0.0	0.185185	1.0	0.0	0.0	1.0	0.0	1.0	...	1.0	0.0	0.666667	0.000000	0.000000	0.0
449	1.0	1.0	1.0	0.518519	1.0	0.0	1.0	1.0	1.0	1.0	...	1.0	0.0	0.333333	0.000000	0.000000	0.0

5 rows × 21 columns

+ Code + Text

```
x_test.shape
```

```
(7018, 21)
```

```
y_train.shape
```

```
(16373,)
```

```
y_test.shape
```

```
(7018,)
```

Balancing technique

```
[ ] y_train.value_counts()
```

```
0.0    13982
1.0     2391
Name: Diabetes_binary, dtype: int64
```

```
[ ] from imblearn.over_sampling import SMOTE
```

```
[ ] smote = SMOTE()
```

```
[ ] x_train_smote,y_train_smote = smote.fit_resample(x_train,y_train)
```

```
y_train_smote.value_counts()
```

```
1.0    13982
0.0    13982
Name: Diabetes_binary, dtype: int64
```

Decision Tree Classifier

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

```
[ ] model1 = DecisionTreeClassifier()
```

Decision Tree Classifier

```
[ ] from sklearn.tree import DecisionTreeClassifier
```

```
[ ] model1 = DecisionTreeClassifier()
```

```
[ ] model1.fit(x_train,y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

+ Code + Text

```
[ ] y_predict = model1.predict(x_test)
```

+ Code + Text

```
[ ] y_predict
```

```
array([0., 0., 1., ..., 0., 0., 0.])
```

+ Code + Text

```
[ ] y_predict_train = model1.predict(x_train)
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```


```
[ ] from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
```


```
[ ] print('Testing Accuracy = ', accuracy_score(y_test,y_predict))
    print('Training Accuracy = ', accuracy_score(y_train,y_predict_train))
```

```
Testing Accuracy =  0.7774294670846394
Training Accuracy =  0.9989617052464423
```

```
[ ] pd.crosstab(y_test,y_predict)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	5164	881
1.0	681	292

```
[ ] print(classification_report(y_test,y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.85      0.87      6045
         1.0       0.25      0.30      0.27       973

    accuracy                           0.78      7018
   macro avg       0.57      0.58      0.57      7018
weighted avg       0.80      0.78      0.79      7018
```

```
[ ] from sklearn.model_selection import GridSearchCV
```

```
[ ] parameters = {
        'criterion': ['gini', 'entropy'],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }
```

```
[ ] clf = GridSearchCV(model1,param_grid = parameters,verbose =2)
```

```
clf.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 72 candidates, totalling 360 fits
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
```

```
[ ] clf.fit(x_train,y_train)
```

```
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
```

```
clf.fit(x_train,y_train)
```

```
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=4, min_samples_split=5; total time=   0.0s
```

```
clf.fit(x_train,y_train)
```

```
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
```

```
clf.fit(x_train,y_train)
```

```
[CV] END criterion=gini, max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.0s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=2; total time=   0.0s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.0s
[CV] END criterion=gini, max_depth=30, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
```

```
clf.fit(x_train,y_train)
```

```
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=4, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=4, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=1, min_samples_split=10; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=2; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=5; total time=   0.1s
[CV] END criterion=entropy, max_depth=30, min_samples_leaf=2, min_samples_split=10; total time=   0.1s
```

```
[ ] print(classification_report(y_test,y_))
```

```
              precision    recall  f1-score   support

         0.0       0.88      0.95      0.91      6045
         1.0       0.37      0.18      0.24       973

    accuracy                           0.84      7018
   macro avg       0.62      0.57      0.58      7018
weighted avg       0.81      0.84      0.82      7018
```

+ Code + Text

RandomForest

```
[ ] from sklearn.ensemble import RandomForestClassifier

[ ] model3 =RandomForestClassifier()

[ ] model3.fit(x_train,y_train)
```

```
RandomForestClassifier()
```

```
[ ] r_y_predict = model3.predict(x_test)
    r_y_predict_train = model3.predict(x_train)
```

```
[ ] print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict))
    print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train))
```

```
Testing Accuracy =  0.8605015673981191
Training Accuracy =  0.9989617052464423
```

```
[ ] pd.crosstab(y_test,r_y_predict)
```

col_0	0.0	1.0
Diabetes_binary		
0.0	5911	134
1.0	845	128

```
print(classification_report(y_test,r_y_predict))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6045
         1.0       0.49      0.13      0.21       973

    accuracy                           0.86      7018
   macro avg       0.68      0.55      0.57      7018
weighted avg       0.82      0.86      0.82      7018
```

```
[ ] from sklearn.model_selection import GridSearchCV

parameters1 = {
    'n_estimators': [100, 200],  # Number of trees in the forest
    'criterion': ['gini', 'entropy'],  # Criterion for splitting
    'max_depth': [None, 10, 20],  # Maximum depth of trees
    'min_samples_split': [2, 5],  # Minimum samples required to split a node
    'min_samples_leaf': [1, 2],  # Minimum samples required to be in a leaf node
}
```

+ Code + Text

```
clf1 = GridSearchCV(model3,param_grid = parameters1,verbose =2)
```

```
clf1.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.9s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.6s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.6s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   3.5s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   2.4s
```

```
clf1.fit(x_train,y_train)
```

```
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   1.8s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   1.8s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.3s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.0s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   0.8s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   0.9s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   1.7s
[CV] END criterion=gini, max_depth=10, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.7s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   2.5s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   4.4s
```

```
clf1.fit(x_train,y_train)
```

```
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.5s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   1.7s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   2.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   3.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   3.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.1s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.4s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.9s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.2s
[CV] END criterion=gini, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   2.0s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   2.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   1.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.6s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   3.1s
```

```
clf1.fit(x_train,y_train)
```

```
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time=   2.8s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.5s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   3.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.4s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.4s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.8s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   3.1s
[CV] END criterion=entropy, max_depth=None, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.5s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.8s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   2.7s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time=   1.7s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   0.9s
[CV] END criterion=entropy, max_depth=10, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time=   0.9s
```

```
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   3.0s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time=   2.4s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.1s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=100; total time=   1.2s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   3.4s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.6s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.6s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.4s
[CV] END criterion=entropy, max_depth=20, min_samples_leaf=2, min_samples_split=5, n_estimators=200; total time=   2.3s
```

GridSearchCV
estimator: RandomForestClassifier
RandomForestClassifier

```
clf1.best_score_
```

```
0.8573262018121028
```

```
clf1.best_params_
```

```
{'criterion': 'gini',
 'max_depth': None,
 'min_samples_leaf': 2,
 'min_samples_split': 2,
 'n_estimators': 100}
```

```
model4 =RandomForestClassifier(criterion= 'gini',
 max_depth= 10,
 min_samples_leaf= 1,
 min_samples_split= 2,
 n_estimators= 200
)
```

```
model4.fit(x_train,y_train)
```

RandomForestClassifier
RandomForestClassifier(max_depth=10, n_estimators=200)

```
r_y_predict1 = model4.predict(x_test)
r_y_predict_train1= model4.predict(x_train)
```

```
print('Testing Accuracy = ', accuracy_score(y_test,r_y_predict1))
print('Training Accuracy = ', accuracy_score(y_train,r_y_predict_train1))
```

```
Testing Accuracy =  0.8639213451125677
Training Accuracy =  0.8849935870029927
```

```
pd.crosstab(y_test,r_y_predict1)
```

Diabetes_binary	col_0 0.0	1.0
0.0	6003	42
1.0	913	60

```
print(classification_report(y_test,r_y_predict1))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.99      0.93      6045
         1.0       0.59      0.06      0.11       973

    accuracy                           0.86      7018
   macro avg       0.73      0.53      0.52      7018
weighted avg       0.83      0.86      0.81      7018
```

LogisticRegression

```
from sklearn.linear_model import LogisticRegression
model5 = LogisticRegression()
```

```
model5.fit(x_train,y_train)
```

```
[ ] model5.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
[ ] pred = model5.predict(x_test)
    pred
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
[ ] from sklearn.metrics import accuracy_score, confusion_matrix,classification_report,roc_auc_score,roc_curve
```

```
[ ] accuracy_score(y_test,pred)
```

```
0.8587916785408949
```

+ Code + Text

```
[ ] pd.crosstab(y_test,pred)
```

Diabetes_binary	0.0	1.0
col_0		
0.0	5912	133
1.0	858	115

```
[ ] print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6045
         1.0       0.46      0.12      0.19       973

    accuracy                           0.86      7018
   macro avg       0.67      0.55      0.56      7018
weighted avg       0.82      0.86      0.82      7018
```

SVC

+ Code + Text

```
[ ] from sklearn.svm import SVC
```

```
[ ] model6 = SVC()
```

+ Code + Text

```
[ ] model6.fit(x_train,y_train)
```

```
SVC()
```

```
[ ] y_pred1 = model6.predict(x_train)
```

```
[ ] y_pred = model6.predict(x_test)
```

```
[CV] END .....................C=0.5, gamma=auto, kernel=rbf; total time=   6.1s
[CV] END .................C=1.0, gamma=scale, kernel=linear; total time=   2.3s
[CV] END .................C=1.0, gamma=scale, kernel=linear; total time=   3.1s
[CV] END .................C=1.0, gamma=scale, kernel=linear; total time=   2.6s
[CV] END .................C=1.0, gamma=scale, kernel=linear; total time=   2.2s
[CV] END .................C=1.0, gamma=scale, kernel=linear; total time=   2.2s
[CV] END ....................C=1.0, gamma=scale, kernel=rbf; total time=   7.6s
[CV] END ....................C=1.0, gamma=scale, kernel=rbf; total time=   6.8s
[CV] END ....................C=1.0, gamma=scale, kernel=rbf; total time=   7.5s
[CV] END ....................C=1.0, gamma=scale, kernel=rbf; total time=   6.5s
[CV] END ....................C=1.0, gamma=scale, kernel=rbf; total time=   7.6s
[CV] END ..................C=1.0, gamma=auto, kernel=linear; total time=   2.1s
[CV] END ..................C=1.0, gamma=auto, kernel=linear; total time=   2.1s
[CV] END ..................C=1.0, gamma=auto, kernel=linear; total time=   3.4s
[CV] END ..................C=1.0, gamma=auto, kernel=linear; total time=   3.3s
[CV] END ..................C=1.0, gamma=auto, kernel=linear; total time=   2.4s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   7.7s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   8.2s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   8.1s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   7.4s
[CV] END .....................C=1.0, gamma=auto, kernel=rbf; total time=   8.2s
```

```
GridSearchCV
estimator: SVC
    SVC
```

```
[ ] clf2.best_score_
```

```
0.8549441307176131
```

```
[ ] clf2.best_params_
```

```
{'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'}
```

```
[ ] model7 = SVC(C= 1.0, gamma= 'scale', kernel= 'rbf')
```

```
[ ] model7.fit(x_train,y_train)
```

```
▾ SVC
SVC()
```

```
[ ] y_pred2 = model7.predict(x_train)
```

```
[ ] y_pred3 = model7.predict(x_test)
```

```
[ ] from sklearn.metrics import accuracy_score,classification_report
```

```
[ ] print("Test accuracy", accuracy_score(y_test,y_pred3))
    print("Train accuracy", accuracy_score(y_train,y_pred2))
```

```
Test accuracy 0.8610715303505272
Train accuracy 0.857570390276675
```

```
[ ] pd.crosstab(y_test,y_pred3)
```

Diabetes_binary \ col_0	0.0	1.0
0.0	6029	16
1.0	959	14

```
print(classification_report(y_test,y_pred3))
```

```
              precision    recall  f1-score   support

         0.0       0.86      1.00      0.93      6045
         1.0       0.47      0.01      0.03       973

    accuracy                           0.86      7018
   macro avg       0.66      0.51      0.48      7018
weighted avg       0.81      0.86      0.80      7018
```

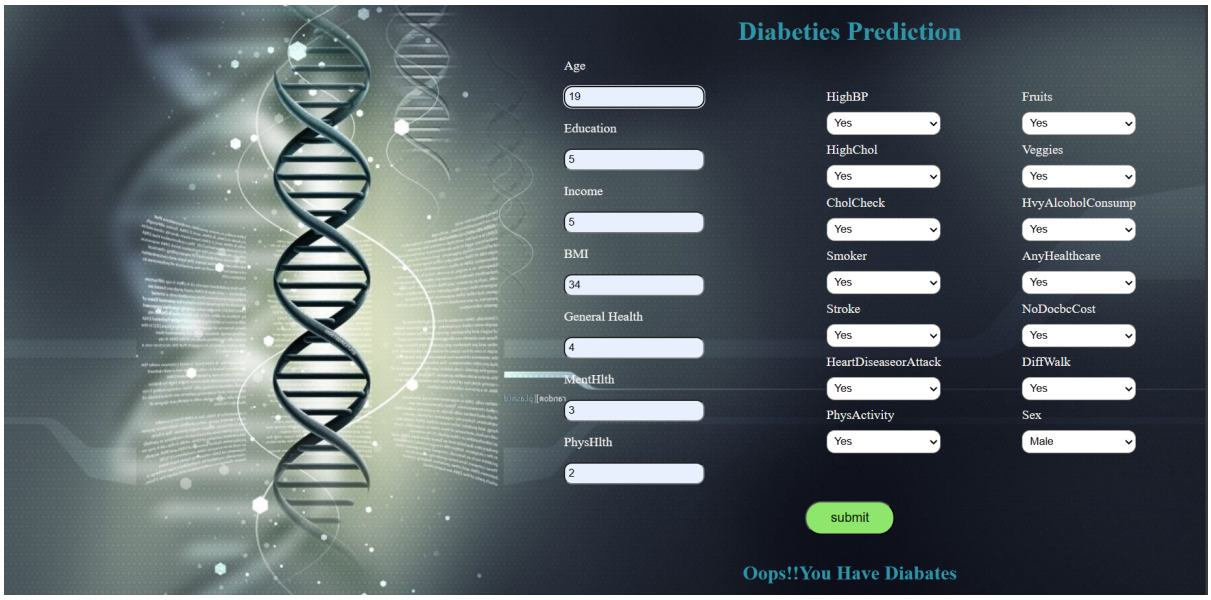
▾ we are using randomforest model

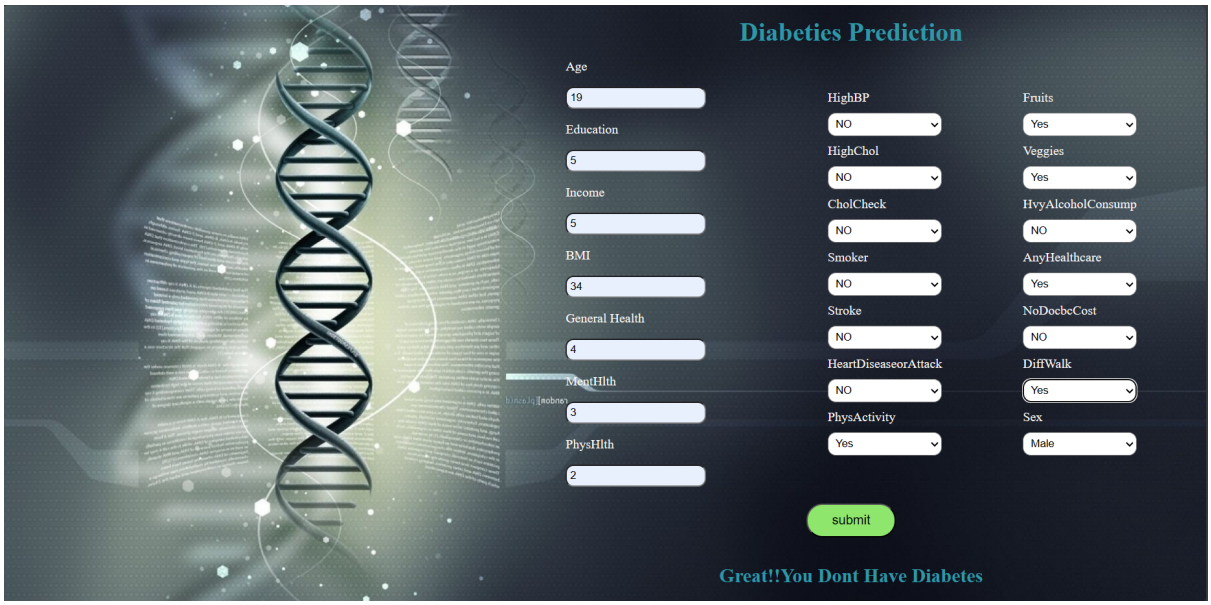
```
[ ] r_y_predict5 = model4.predict(x_test)
    r_y_predict5
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
[ ] import pickle


    pickle.dump(model4,open('startup.pkl','wb'))
```

RESULTS


Diabeties Prediction
Age
19
Education
5
Income
5
BMI
34
General Health
4
MentHlth
3
PhysHlth
2
HighBP
Yes
HighChol
Yes
CholCheck
Yes
Smoker
Yes
Stroke
Yes
HeartDiseaseorAttack
Yes
PhysActivity
Yes
Fruits
Yes
Veggies
Yes
HvyAlcoholConsump
Yes
AnyHealthcare
Yes
NoDocbcCost
Yes
DiffWalk
Yes
Sex
Male
submit
Oops!!You Have Diabates


Diabeties Prediction
Age
19
Education
5
Income
5
BMI
34
General Health
4
MentHlth
3
PhysHlth
2
HighBP
NO
HighChol
NO
CholCheck
NO
Smoker
NO
Stroke
NO
HeartDiseaseorAttack
NO
PhysActivity
Yes
Fruits
Yes
Veggies
Yes
HvyAlcoholConsump
NO
AnyHealthcare
Yes
NoDocbcCost
NO
DiffWalk
Yes
Sex
Male
submit
Great!!You Dont Have Diabetes

ADVANTAGES & DISADVANTAGES

Advantages:

Early detection: Machine learning algorithms can help in detecting diabetes at an early stage, which can lead to timely treatment and better outcomes.

Accuracy: Machine learning algorithms can analyze large amounts of data and provide accurate predictions.

Personalized treatment: Machine learning algorithms can help in providing personalized treatment plans based on the patient's medical history and other factors.

Cost-effective: Machine learning algorithms can help in reducing healthcare costs by predicting the onset of diabetes and preventing complications.

Disadvantages:

Data quality: Machine learning algorithms require high-quality data to provide accurate predictions. Poor data quality can lead to inaccurate predictions.

Data privacy: Machine learning algorithms require access to sensitive patient data, which can raise privacy concerns.

Bias: Machine learning algorithms can be biased if the training data is not diverse enough. This can lead to inaccurate predictions for certain groups of patients.

CONCLUSION

In conclusion, "Diabetes Prediction Using Machine Learning" represents a significant stride towards personalized healthcare and preventive well-being. By leveraging the capabilities of artificial intelligence, this innovative application empowers individuals to take proactive control of their health journey. Through a careful analysis of diverse health parameters and lifestyle choices, the machine learning model provides timely predictions, enabling users to make informed decisions and adopt preventive measures.

The seamless integration of this predictive model into a user-friendly web application not only enhances accessibility but also fosters a culture of health consciousness. The intersection of technology and healthcare in this context underscores the transformative potential of machine learning in predicting and preventing diseases.

The successful development of this diabetes prediction model paves the way for future research and applications in personalized medicine and targeted interventions. The insights gained from this project contribute to a better understanding of the underlying factors and mechanisms associated with diabetes. The user-friendly and accessible nature of the developed tool allows for its potential deployment in real-world healthcare settings, benefiting both healthcare providers and patients.

Overall, the project represents a significant contribution to the field of healthcare analytics and demonstrates the potential of machine learning in improving the early detection and management of diabetes, ultimately leading to enhanced patient care and improved quality of life.

FUTURE SCOPE

Improving accuracy: Machine learning algorithms can be further optimized to improve the accuracy of diabetes prediction. This can be achieved by using more advanced algorithms, incorporating more data sources, and improving data quality.

Personalized treatment: Machine learning algorithms can be used to provide personalized treatment plans based on the patient's medical history and other factors. This can lead to better outcomes and improved patient satisfaction.

Real-time monitoring: Machine learning algorithms can be used to monitor patients in real-time and provide timely alerts when there are signs of diabetes. This can help in preventing complications and improving patient outcomes.

Integration with wearables: Machine learning algorithms can be integrated with wearable devices such as smartwatches and fitness trackers to monitor patients' health in real-time. This can provide valuable insights into patients' health and help in predicting the onset of diabetes.

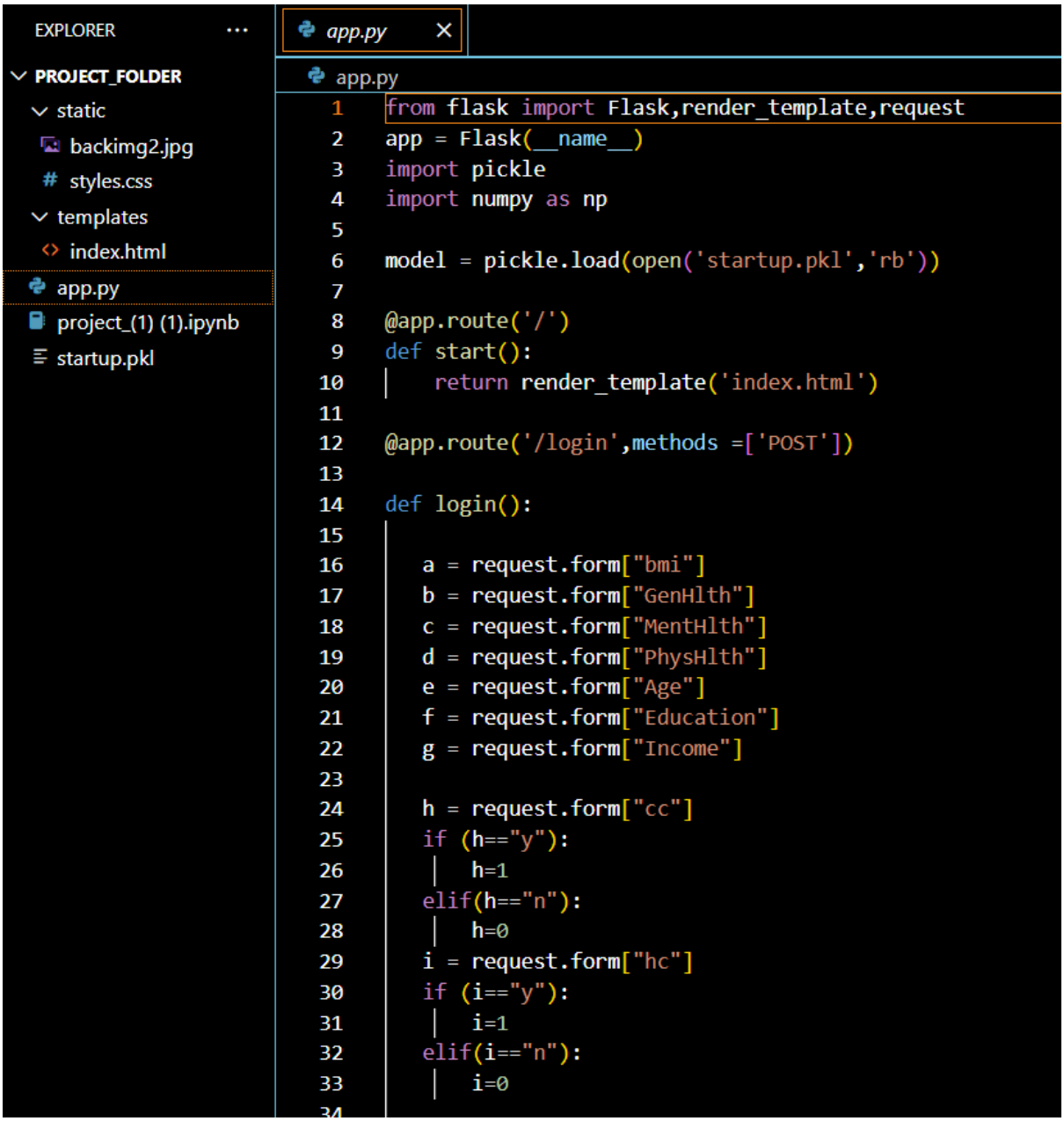
Predicting complications: Machine learning algorithms can be used to predict the likelihood of complications associated with diabetes, such as heart disease, kidney disease, and nerve damage. This can help in preventing complications and improving patient outcomes.

Addressing privacy concerns: Machine learning algorithms require access to sensitive patient data, which can raise privacy concerns. Future research can focus on developing secure and privacy-preserving machine learning algorithms for diabetes prediction.

APPENDIX

Source Code:

App.py

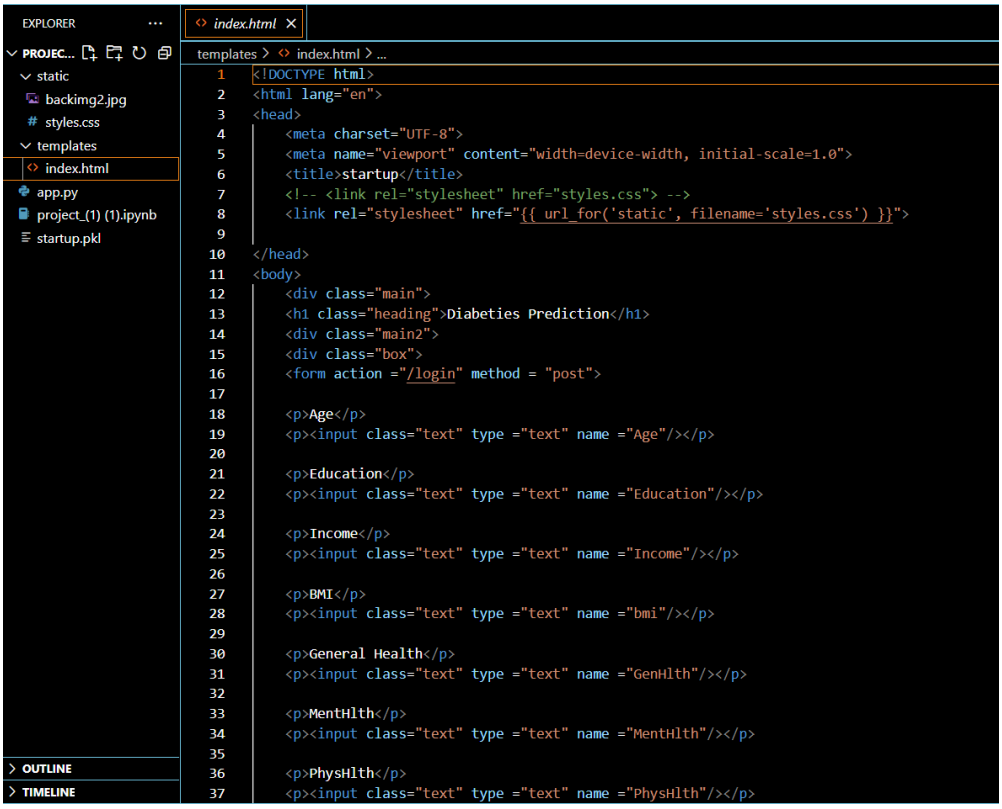


```
34
35      j = request.form["hbp"]
36      if (j=="y"):
37      |   j=1
38      elif(j=="n"):
39      |   j=0
40
41      k = request.form["s"]
42      if (k=="y"):
43      |   k=1
44      elif(k=="n"):
45      |   k=0
46
47
48      l = request.form["st"]
49      if (l=="y"):
50      |   l=1
51      elif(l=="n"):
52      |   l=0
53
54      m = request.form["hda"]
55      if (m=="y"):
56      |   m=1
57      elif(m=="n"):
58      |   m=0
59
60      n = request.form["pa"]
61      if (n=="y"):
62      |   n=1
63      elif(n=="n"):
64      |   n=0
```

```
66      o = request.form["f"]
67      if (o=="y"):
68      |   o=1
69      elif(o=="n"):
70      |   o=0
71
72      p = request.form["v"]
73      if (p=="y"):
74      |   p=1
75      elif(p=="n"):
76      |   p=0
77
78
79      q = request.form["hac"]
80      if (q=="y"):
81      |   q=1
82      elif(q=="n"):
83      |   q=0
84
85
86      r = request.form["ahc"]
87      if (r=="y"):
88      |   r=1
89      elif(r=="n"):
90      |   r=0
91
92      s = request.form["ndc"]
93      if (s=="y"):
94      |   s=1
95      elif(s=="n"):
96      |   s=0
97
```

```
98      u = request.form["dw"]
99      if (u=="y"):
100     |   u=1
101     elif(u=="n"):
102     |   u=0
103
104     v = request.form["sex"]
105     if (v=="y"):
106     |   v=1
107     elif(v=="n"):
108     |   v=0
109
110     t =  [[float(a),float(b),float(c),float(d),float(e),float(f),float(g),float(h),float(i),float(j),float(k),float(l),float(m),float(n),float(o),
111     output =model.predict(t)
112
113     if(output==1.0):
114     |   output="Oops!!You Have Diabates"
115     elif(output==0.0):
116     |   output="Great!!You Dont Have Diabetes"
117
118     return render_template("index.html", y = str((output)))
119 #    return render_template("front.html")
120
121 if __name__ == '__main__' :
122 |   app.run(debug=True)
123
124
```

Index.html:


EXPLORER
index.html
templates > index.html > ...
PROJEC...
static
backimg2.jpg
styles.css
templates
index.html
app.py
project_(1) (1).ipynb
startup.pkl
OUTLINE
TIMELINE
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>startup</title>
7 <!-- <link rel="stylesheet" href="styles.css"> -->
8 <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
9
10 </head>
11 <body>
12
13 <h1 class="heading">Diabeties Prediction</h1>
14
15
16 <form action ="/login" method = "post">
17
18 Age
19 <input class="text" type ="text" name ="Age"/>
20
21 Education
22 <input class="text" type ="text" name ="Education"/>
23
24 Income
25 <input class="text" type ="text" name ="Income"/>
26
27 BMI
28 <input class="text" type ="text" name ="bmi"/>
29
30 General Health
31 <input class="text" type ="text" name ="GenHlth"/>
32
33 MentHlth
34 <input class="text" type ="text" name ="MentHlth"/>
35
36 PhysHlth
37 <input class="text" type ="text" name ="PhysHlth"/>

```
</div>

    <div class="drop">
        <div class="drop1">
    <label for ="HighBP">HighBP </label>
    <select class="down" name = "hbp">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="HighChol">HighChol</label>
    <select class="down" name = "hc">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="CholCheck">CholCheck</label>
    <select class="down" name = "cc">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>
```

```
    <label for ="Smoker">Smoker</label>
    <select class="down" name = "s">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="Stroke">Stroke</label>
    <select class="down" name = "st">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="HeartDiseaseorAttack">HeartDiseaseorAttack</label>
    <select class="down" name = "hda">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="PhysActivity">PhysActivity</label>
    <select class="down" name = "pa">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>
</div>
<div class="drop1">
    <label for ="Fruits">Fruits</label>
    <select class="down" name = "f">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="Veggies">Veggies</label>
    <select class="down" name = "v">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>
```

```
    <label for ="HvyAlcoholConsump">HvyAlcoholConsump</label>
    <select class="down" name = "hac">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="AnyHealthcare">AnyHealthcare</label>
    <select class="down" name = "ahc">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>
```

```
    <label for ="NoDocbcCost">NoDocbcCost</label>
    <select class="down" name = "ndc">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="DiffWalk">DiffWalk</label>
    <select class="down" name = "dw">
    <option value ="y">Yes</option>
    <option value ="n">NO</option>
    </select>

    <label for ="Sex">Sex</label>
    <select class="down" name = "sex">
    <option value ="y">Male</option>
    <option value ="n">Female</option>
    </select>
</div>
</div>
</div>

    <p><input class="submit" type ="submit" value ="submit"/></p>

    <h2 class="output"><b>{{y}}</b></h2>



</form>
</div>
</body>
</html>
```

Styles.css:

```css
body{
    background-image: url("./backimg2.jpg");
    background-size: 100vw 100vh;
    background-repeat: no-repeat;

}
.heading
{
    color: #2E97A7;
}
.drop
{
    display: flex;
    flex-direction: column;

}
```

```css
.main
{
    display: flex;
    flex-direction: column;
    align-items: center;
    height: 95vh;
    width: 40vw;
    margin-left: 50vw;
    margin-top: -15px;
}
.main2
{
    display: flex;
    flex-direction: row;
    color: #F8F6F4;
    margin-top: -23px;
}
.box
{
    padding-right: 100px;
}
.drop{
    display: flex;
    flex-direction: row;

}
.drop1
{
    display: flex;
    flex-direction: column;
    padding: 50px;
    justify-content: space-around;
}
```

```css
.text
{
    padding: 4px;
    border-radius: 10px;

}
.down
{
    padding: 5px;
    border-radius: 10px;
}
.submit
{
    padding: 10px 30px;
    font-size: medium;
    background-color: rgb(143, 230, 109);
    border-radius: 20px;
    margin-top: -15px;
    margin-bottom: -20px;
}
.main2
{

    padding-left: 50px;
    border-radius: 20px;
}
.output
{
    padding: 10px 30px;
    border-radius: 5px;
    color: #2E97A7;
}
```

GitHub Link:

https://github.com/smartinternz02/SI-GuidedProject-600733-1697543599

Project Demo Link:

https://drive.google.com/file/d/1QxA0jN9Hf3uaNTWc0knHgStB-e7gw0Ai/view?usp=sharing