Prediction Usi

BY:- AM

In today's life, diabetes has become a widespread and serious heal
modern lifestyle changes. The rise of sedentary habits, unhealthy c
increasing levels of stress have contributed to the growing preval
diabetes, which now affects people of all ages, including younger in
chronic condition, characterized by the body's inability to properly
sugar levels, can lead to severe complications such as heart disea
failure, and vision loss if left untreated. Managing diabetes require
monitoring, dietary adjustments, and sometimes medication, which
burden on individuals and healthcare systems. Despite these challe
advancements in healthcare technology have made managing diab
accessible, and prevention through healthy lifestyle choices is now

Machine learning is transforming the field of diabetes care by offering new ways to
manage the disease. With the availability of large amounts of health data, machine l
can analyze key factors such as glucose levels, age, BMI, and family history to predi
risk of developing diabetes. In your project, applying algorithms like Logistic Regres
Forest, Gradient Boosting, and SVM enables you to model this risk accurately. Logis
helps in understanding the direct relationship between health indicators and diabet
while Random Forest and Gradient Boosting improve prediction accuracy by learnin
patterns in the data. SVM adds flexibility by handling non-linear relationships betwe
These models, when evaluated through metrics such as accuracy and ROC-AUC, car
identify at-risk individuals early, allowing for timely medical intervention and perso
plans, ultimately improving outcomes for patients in today's fast-paced, data-driver

Let's start the project :-

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

LOADING DATA & EDA

In [3]:
```
df = pd.read_csv(r"C:\Users\ameet\Downloads\archive (12)\diabetes.csv")
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [4]:
```
df.tail()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

In [5]:
```
df.shape
```

Out[5]:
```
(768, 9)
```

In [6]:
```
df.ndim
```

Out[6]:
```
2
```

Rename the column using the mapping :-

In [8]:
```
column_mapping = {'DiabetesPedigreeFunction': 'DPF'}
df.rename(columns=column_mapping, inplace=True)
```

In [9]:
```
df.columns
```

Out[9]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DPF', 'Age', 'Outcome'],
      dtype='object')
```

In [10]:
```
df.describe()
```

Out[10]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	Age	Outco
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.0000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.3489
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.4769
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.0000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.0000