In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
data = pd.read_csv(r"C:\Users\ASUS\Downloads\archive (8)\Healthcare-Diabetes.csv")
```

In [3]:
```
data.head()
```

Out[3]:

	Id	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	1	6	148	72	35	0	33.6	0.627	50	1
1	2	1	85	66	29	0	26.6	0.351	31	0
2	3	8	183	64	0	0	23.3	0.672	32	1
3	4	1	89	66	23	94	28.1	0.167	21	0
4	5	0	137	40	35	168	43.1	2.288	33	1

In [4]:
```
data = data.drop(["Id"], axis=1)
```

In [5]:
```
data.head()
```

Out[5]:

	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 [6]:
```
data.tail()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
2763	2	75	64	24	55	29.7	0.370	33	0
2764	8	179	72	42	130	32.7	0.719	36	1
2765	6	85	78	0	0	31.2	0.382	42	0
2766	0	129	110	46	130	67.1	0.319	26	1
2767	2	81	72	15	76	30.1	0.547	25	0

In [7]:
```
data.sample(15)
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
54	7	150	66	42	342	34.7	0.718	42	0
2104	5	143	78	0	0	45.0	0.190	47	0
324	2	112	75	32	0	35.7	0.148	21	0
1214	1	100	72	12	70	25.3	0.658	28	0
1119	4	137	84	0	0	31.2	0.252	30	0
2315	4	99	72	17	0	25.6	0.294	28	0
714	3	102	74	0	0	29.5	0.121	32	0
1862	2	94	76	18	66	31.6	0.649	23	0
37	9	102	76	37	0	32.9	0.665	46	1
2581	4	112	78	40	0	39.4	0.236	38	0
2513	1	120	80	48	200	38.9	1.162	41	0
1517	6	162	62	0	0	24.3	0.178	50	1
1712	4	114	64	0	0	28.9	0.126	24	0
2452	3	89	74	16	85	30.4	0.551	38	0
2522	4	145	82	18	0	32.5	0.235	70	1

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2768 entries, 0 to 2767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               2768 non-null   int64  
 1   Glucose                   2768 non-null   int64  
 2   BloodPressure             2768 non-null   int64  
 3   SkinThickness             2768 non-null   int64  
 4   Insulin                   2768 non-null   int64  
 5   BMI                       2768 non-null   float64
 6   DiabetesPedigreeFunction  2768 non-null   float64
 7   Age                       2768 non-null   int64  
 8   Outcome                   2768 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 194.8 KB
```

In [9]:
```
data.describe()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	2768.000000	2768.000000	2768.000000	2768.000000	2768.000000	2768.000000	2768.000000	2768.000000
mean	3.742775	121.102601	69.134393	20.824422	80.127890	32.137392	0.471193	33.132225
std	3.323801	32.036508	19.231438	16.059596	112.301933	8.076127	0.325669	11.777230
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.244000	24.000000
50%	3.000000	117.000000	72.000000	23.000000	37.000000	32.200000	0.375000	29.000000
75%	6.000000	141.000000	80.000000	32.000000	130.000000	36.625000	0.624000	40.000000
max	17.000000	199.000000	122.000000	110.000000	846.000000	80.600000	2.420000	81.000000

In [10]:
```
data.shape
```

Out[10]:
```
(2768, 9)
```

In [11]:
```
data.size
```

Out[11]:
```
24912
```

In [12]:
```
data.isnull().sum()
```

Out[12]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [13]:
```
data.Outcome.value_counts()
```

Out[13]:
```
Outcome
0    1816
1     952
Name: count, dtype: int64
```

0--> No 1--> Yes

In [14]:
```
No = data[data.Outcome == 0]
Yes = data[data.Outcome == 1]
```

In [15]:
```
No.shape
```

Out[15]:
```
(1816, 9)
```

In [16]:
```
Yes.shape
```

Out[16]:
```
(952, 9)
```

In [17]:
```
No_sample = No.sample(n=952)
No_sample.shape
```

Out[17]:
```
(952, 9)
```

In [18]:
```
newdata = pd.concat([No_sample , Yes], axis=0)
```

In [19]:
```
newdata.sample(15)
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
479	4	132	86	31	0	28.0	0.419	63	0
690	8	107	80	0	0	24.6	0.856	34	0
2548	2	121	70	32	95	39.1	0.886	23	0
1810	1	93	56	11	0	22.5	0.417	22	0
1512	13	153	88	37	140	40.6	1.174	39	0
2451	2	99	0	0	0	22.2	0.108	23	0
33	6	92	92	0	0	19.9	0.188	28	0
1074	10	161	68	23	132	25.5	0.326	47	1
1753	3	163	70	18	105	31.6	0.268	28	1
878	3	171	72	33	135	33.3	0.199	24	1
2217	0	107	62	30	74	36.6	0.757	25	1
338	9	152	78	34	171	34.2	0.893	33	1
1501	2	106	56	27	165	29.0	0.426	22	0
1372	4	183	0	0	0	28.4	0.212	36	1
2470	11	138	74	26	144	36.1	0.557	50	1

In [20]:
```
newdata.Outcome.value_counts()
```

Out[20]:
```
Outcome
0    952
1    952
Name: count, dtype: int64
```

In [21]:
```
X=newdata.drop("Outcome" , axis="columns")
X.head()
```

Out[21]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
887	4	99	76	15	51	23.2	0.223	21
2583	0	141	84	26	0	32.4	0.433	22
762	9	89	62	0	0	22.5	0.142	33
1343	1	119	44	47	63	35.5	0.280	25
1675	5	147	75	0	0	29.9	0.434	28

In [22]:
```
Y=newdata.Outcome.values
```

In [23]:
```
Y
```

Out[23]:
```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

In [24]:
```
print(Y)
```

```
[0 0 0 ... 1 1 1]
```

In [25]:
```
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X , Y , test_size= 0.2)
```

In [26]:
```
len(X_train)
```

Out[26]:
```
1523
```

In [27]:
```
len(X_test)
```

Out[27]:
```
381
```

In [28]:
```
len(y_train)
```

Out[28]:
```
1523
```

In [29]:
```
len(y_test)
```

Out[29]:
```
381
```

In [30]:
```
X_train.shape
```

Out[30]:
```
(1523, 8)
```

In [31]:
```
y_train.shape
```

Out[31]:
```
(1523,)
```

In [32]:
```
X_train.head()
```

Out[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
1928	2	129	0	0	0	38.5	0.304	41
1698	9	120	72	22	56	20.8	0.733	48
896	1	117	88	24	145	34.5	0.403	40
58	0	146	82	0	0	40.5	1.781	44
1175	0	101	62	0	0	21.9	0.336	25

In [33]:
```
y_train
```

Out[33]:
```
array([0, 0, 1, ..., 1, 0, 1], dtype=int64)
```

In [34]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [35]:
```
lrmodel = LogisticRegression()
lrmodel.fit(X_train , y_train)
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceW
arning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[35]:
```
▾ LogisticRegression
LogisticRegression()
```

In [36]:
```
lrmodel.predict(X_test)
```

Out[36]:
```
array([1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0,
       1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1,
       0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 1, 0, 0, 1], dtype=int64)
```

In [37]:
```
from sklearn.metrics import accuracy_score
X_train_prediction = lrmodel.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction , y_train)
```

In [38]:
```
training_data_accuracy
```

Out[38]:
```
0.7590282337491793
```

In [39]:
```
from sklearn.metrics import accuracy_score
X_test_prediction = lrmodel.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction , y_test)
```

In [40]:
```
testing_data_accuracy
```

Out[40]:
```
0.7532808398950132
```

In [41]:
```
dtmodel = DecisionTreeClassifier(random_state=42)
dtmodel.fit(X_train, y_train)
```

Out[41]:
```
DecisionTreeClassifier(random_state=42)
```

In [42]:
```
from sklearn.metrics import accuracy_score
X_train_prediction = dtmodel.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction , y_train)
```

In [43]:
```
training_data_accuracy
```

Out[43]:
```
1.0
```

In [44]:
```
from sklearn.metrics import accuracy_score
X_test_prediction = dtmodel.predict(X_test)
testing_data_accuracy = accuracy_score(X_test_prediction , y_test)
```

In [45]:
```
testing_data_accuracy
```

Out[45]:
```
0.952755905511811
```

In [46]:
```
'''
Pregnancies: To express the Number of pregnancies

Glucose: To express the Glucose level in blood

BloodPressure: To express the Blood pressure measurement

SkinThickness: To express the thickness of the skin

Insulin: To express the Insulin level in blood

BMI: To express the Body mass index

DiabetesPedigreeFunction: To express the Diabetes percentage

Age: To express the age

Outcome: To express the final result 1 is Yes and 0 is No
'''
```

Out[46]:
```
'\nPregnancies: To express the Number of pregnancies\n\nGlucose: To express the Glucose level in blood\n\nBlood
Pressure: To express the Blood pressure measurement\n\nSkinThickness: To express the thickness of the skin\n\nI
nsulin: To express the Insulin level in blood\n\nBMI: To express the Body mass index\n\nDiabetesPedigreeFunctio
n: To express the Diabetes percentage\n\nAge: To express the age\n\nOutcome: To express the final result 1 is Y
es and 0 is No\n'
```

In [47]:
```
dtmodel.predict([[9,122,56,0,0,33.3,1.114,33]])
```

```
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[47]:
```
array([1], dtype=int64)
```

In [48]:
```
input_data = (9,122,56,0,0,33.3,1.114,33)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
if prediction == [0]:
    print("The person does not have Diabetes.")
else:
    print("The person has Diabetes.")
```

```
The person has Diabetes.
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [49]:
```
import pickle
with open("Diabetes_model_pickle" , "wb") as file:
    pickle.dump(dtmodel , file)
```

In [1]:
```
import numpy as np
import pickle
with open("Diabetes_model_pickle" , "rb") as file:
    dtmodel = pickle.load(file)
```

In [2]:
```
input_data = (0,152,82,39,272,41.5,0.270,27)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = dtmodel.predict(input_data_reshaped)
if prediction == [0]:
    print("The person does not have Diabetes")
else:
    print("The person has Diabetes")
```

```
The person does not have Diabetes
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

In [56]:
```
Pregnancies = int(input(""))
Glucose = int(input(""))
BloodPressure = int(input(""))
SkinThickness = int(input(""))
Insulin = int(input(""))
BMI = float(input(""))
DiabetesPedigreeFunction = float(input(""))
Age = int(input(""))
```

In [57]:
```
input_data = (Pregnancies,Glucose,BloodPressure,SkinThickness,Insulin,BMI,DiabetesPedigreeFunction,Age)
input_data_as_numpy_array = np.asarray(input_data)
input_data_reshaped = input_data_as_numpy_array.reshape(1 , -1)
prediction = model.predict(input_data_reshaped)
if prediction == [0]:
    print("The person does not have Diabetes")
else:
    print("The person has Diabetes")
```

```
The person has Diabetes
C:\Users\ASUS\AppData\Roaming\Python\Python311\site-packages\sklearn\base.py:465: UserWarning: X does not have v
alid feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```

In [ ]: