In [2]:
```
import pandas as pd
import pickle
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
data=pd.read_csv("C:/Users/gonab/OneDrive/Desktop/diabities.csv")
data
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

In [4]:
```
data.describe()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diab
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [6]:
```
data.shape
```

Out[6]:
```
(768, 9)
```

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

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

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.

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

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

In [10]:
```
data['Age'].unique()
```

Out[10]:
```
array([50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 34, 57, 59, 51, 27, 41, 43,
       22, 38, 60, 28, 45, 35, 46, 56, 37, 48, 40, 25, 24, 58, 42, 44, 39,
       36, 23, 61, 69, 62, 55, 65, 47, 52, 66, 49, 63, 67, 72, 81, 64, 70,
       68], dtype=int64)
```

In [11]:
```
data['Glucose'].unique()
```

Out[11]:
```
array([148,  85, 183,  89, 137, 116,  78, 115, 197, 125, 110, 168, 139,
       189, 166, 100, 118, 107, 103, 126,  99, 196, 119, 143, 147,  97,
       145, 117, 109, 158,  88,  92, 122, 138, 102,  90, 111, 180, 133,
       106, 171, 159, 146,  71, 105, 101, 176, 150,  73, 187,  84,  44,
       141, 114,  95, 129,  79,   0,  62, 131, 112, 113,  74,  83, 136,
        80, 123,  81, 134, 142, 144,  93, 163, 151,  96, 155,  76, 160,
       124, 162, 132, 120, 173, 170, 128, 108, 154,  57, 156, 153, 188,
       152, 104,  87,  75, 179, 130, 194, 181, 135, 184, 140, 177, 164,
        91, 165,  86, 193, 191, 161, 167,  77, 182, 157, 178,  61,  98,
       127,  82,  72, 172,  94, 175, 195,  68, 186, 198, 121,  67, 174,
       199,  56, 169, 149,  65, 190], dtype=int64)
```

In [12]:
```
data.groupby(['Outcome']).count()
```

Out[12]:

Outcome	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigr
0	500	500	500	500	500	500	
1	268	268	268	268	268	268	

In [13]:
```
data=pd.get_dummies(data,dtype=int)
data.head()
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	6	148	72	35	0	33.6	0.
1	1	85	66	29	0	26.6	0.
2	8	183	64	0	0	23.3	0.
3	1	89	66	23	94	28.1	0.
4	0	137	40	35	168	43.1	2.

In [14]:
```
cor_mat=data.corr()
cor_mat
```

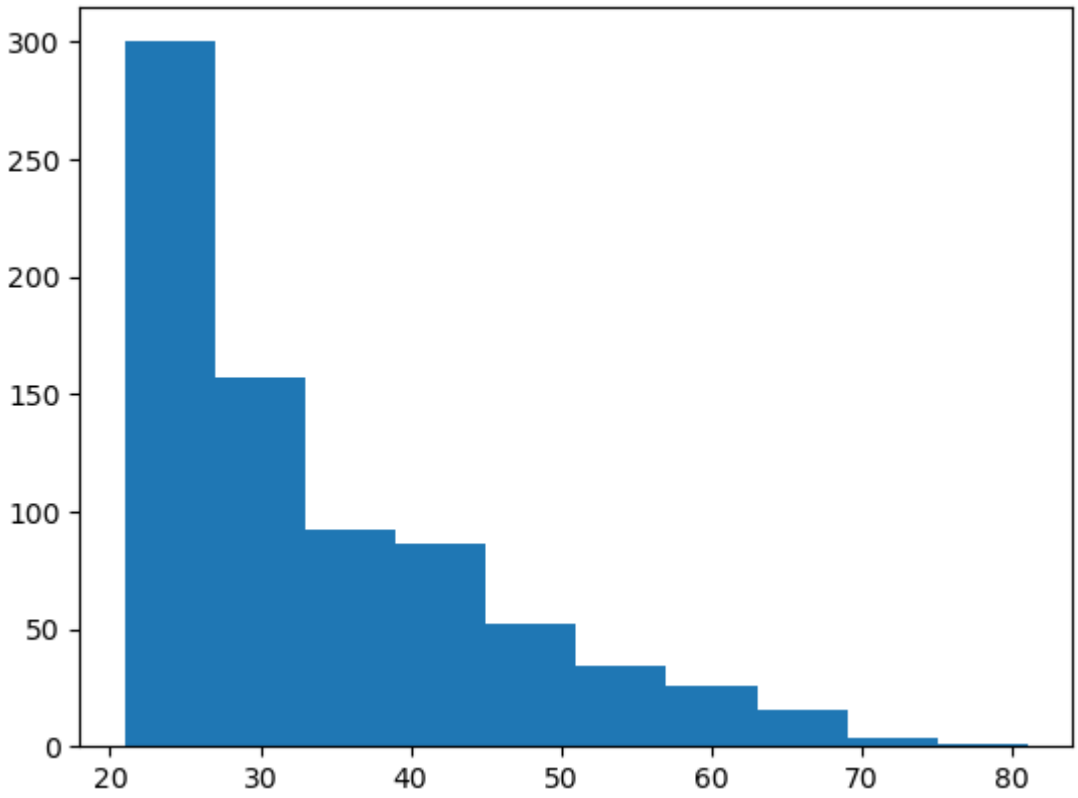
Out[14]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0

In [15]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [16]:
```
plt.hist(data['Age'])
```

Out[16]:
```
(array([300., 157.,  92.,  86.,  52.,  34.,  26.,  16.,   4.,   1.]),
 array([21., 27., 33., 39., 45., 51., 57., 63., 69., 75., 81.]),
 <BarContainer object of 10 artists>)
```



In [17]:
```
x=data.drop("Outcome",axis=1)
y=data['Outcome']
x
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 8 columns

In [18]:
```
y
```

Out[18]:
```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [19]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_sta
```

In [20]:
```
x_train.shape
```

Out[20]:
```
(537, 8)
```

In [21]:
```
x_test.shape
```

Out[21]:
```
(231, 8)
```

In [22]:
```
y_train.shape
```

Out[22]:
```
(537,)
```

In [23]:
```
y_test.shape
```

Out[23]:
```
(231,)
```

In [24]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression()
classifier.fit(x_train,y_train)#for fitting and training the model
```

Out[24]:
```
LogisticRegression
LogisticRegression()
```

In [25]:
```
x_test
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
285	7	136	74	26	135	26.0	
101	1	151	60	0	0	26.1	
581	6	109	60	27	0	25.0	
352	3	61	82	28	0	34.4	
726	1	116	78	29	180	36.1	
...	...	...	...	...	...	...	
241	4	91	70	32	88	33.1	
599	1	109	38	18	120	23.1	
650	1	91	54	25	100	25.2	
11	10	168	74	0	0	38.0	
214	9	112	82	32	175	34.2	

231 rows × 8 columns

In [26]:
```
ypred=classifier.predict(x_test)
ypred
```

Out[26]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0], dtype=int64)
```

In [27]:
```
Results=pd.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=ypred
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(15)
```

Out[27]:

	index	Price	Predicted	ID
0	285	0	0	0
1	101	0	0	1
2	581	0	0	2
3	352	0	0	3
4	726	0	0	4
5	472	0	0	5
6	233	0	0	6
7	385	0	0	7
8	556	0	0	8
9	59	0	0	9
10	756	0	0	10
11	341	0	0	11
12	445	1	1	12
13	614	1	1	13
14	371	0	0	14

In [28]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,ypred)
```

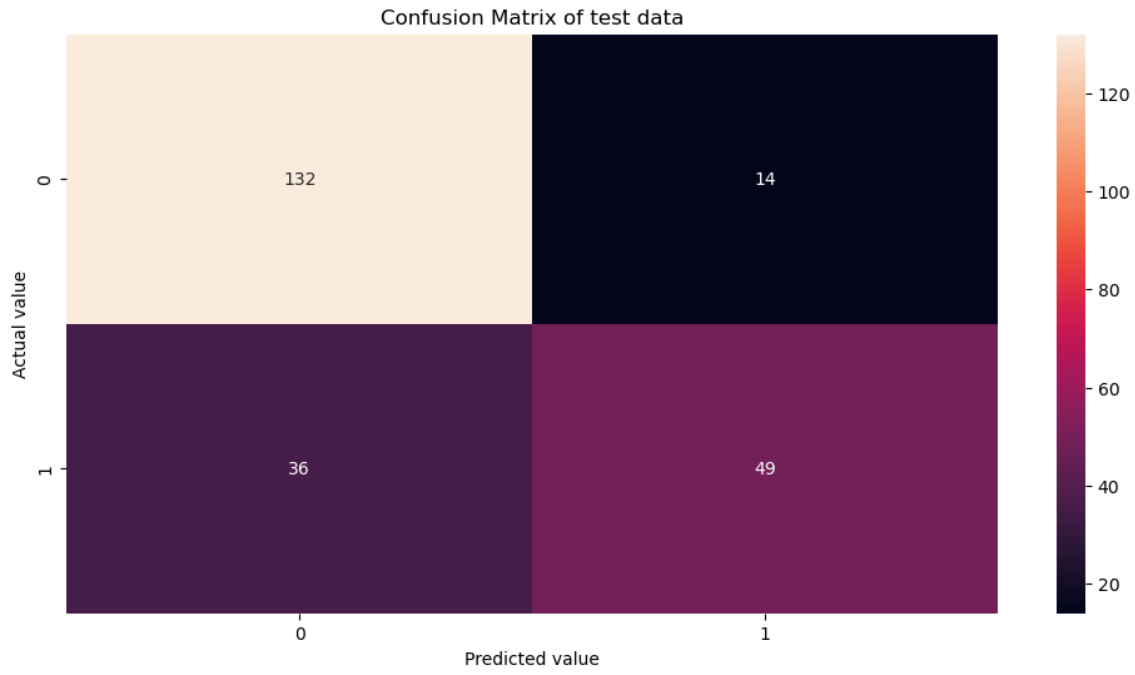
Out[28]:
```
0.7835497835497836
```

In [40]:
```
from sklearn.metrics import confusion_matrix
conf_mat=confusion_matrix(y_test,ypred)
confusion_matrix(y_test,ypred)
```

Out[40]:
```
array([[132,  14],
       [ 36,  49]], dtype=int64)
```

In [35]:
```
plt.figure(figsize=(12,6))
sns.heatmap(conf_mat,annot=True,fmt='d')
plt.title("Confusion Matrix of test data")
plt.xlabel("Predicted value")
plt.ylabel("Actual value")
```

Out[35]: Text(120.72222222222221, 0.5, 'Actual value')



In [36]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,ypred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       146
           1       0.78      0.58      0.66        85

    accuracy                           0.78       231
   macro avg       0.78      0.74      0.75       231
weighted avg       0.78      0.78      0.78       231
```

In [37]:
```
TN=conf_mat[0][0]
FP=conf_mat[0][1]
FN=conf_mat[1][0]
TP=conf_mat[1][1]
```

In [38]:
```
recall=TP/(TP+FN)
print("Recall=",recall)
```

```
Recall= 0.5764705882352941
```

In [39]:
```
precision=TP/(TP+FP)
print("Precision=",precision)
```

```
Precision= 0.7777777777777778
```

In [ ]:

In [ ]: