In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df=pd.read_csv("diabetes.csv")
```

In [3]:

```
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
0	6	148	72	35	0	33.6	0.62
1	1	85	66	29	0	26.6	0.35
2	8	183	64	0	0	23.3	0.67
3	1	89	66	23	94	28.1	0.16
4	0	137	40	35	168	43.1	2.28

In [4]:

```
df.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 [5]:

```
df.describe()
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabet
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 [6]:

```
df.isna().sum()
```

Out[6]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:

```
df.corr().style.background_gradient()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.0
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.2
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.2
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.3
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.1
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.0
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.1
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.0
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.2

In [8]:

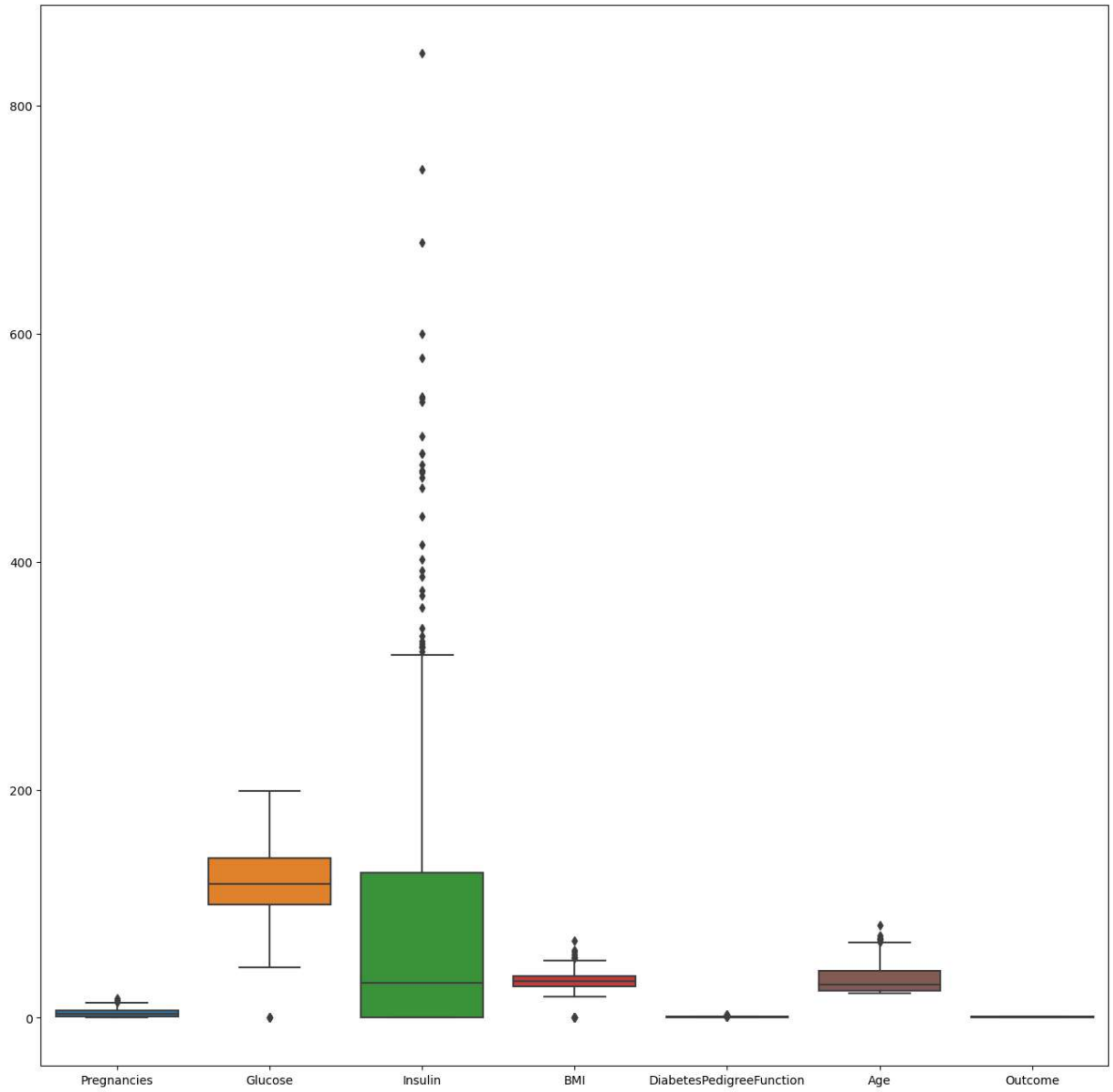
```
df.drop("SkinThickness",axis=1,inplace=True)
```

In [9]:

```
df.drop("BloodPressure",axis=1,inplace=True)
```

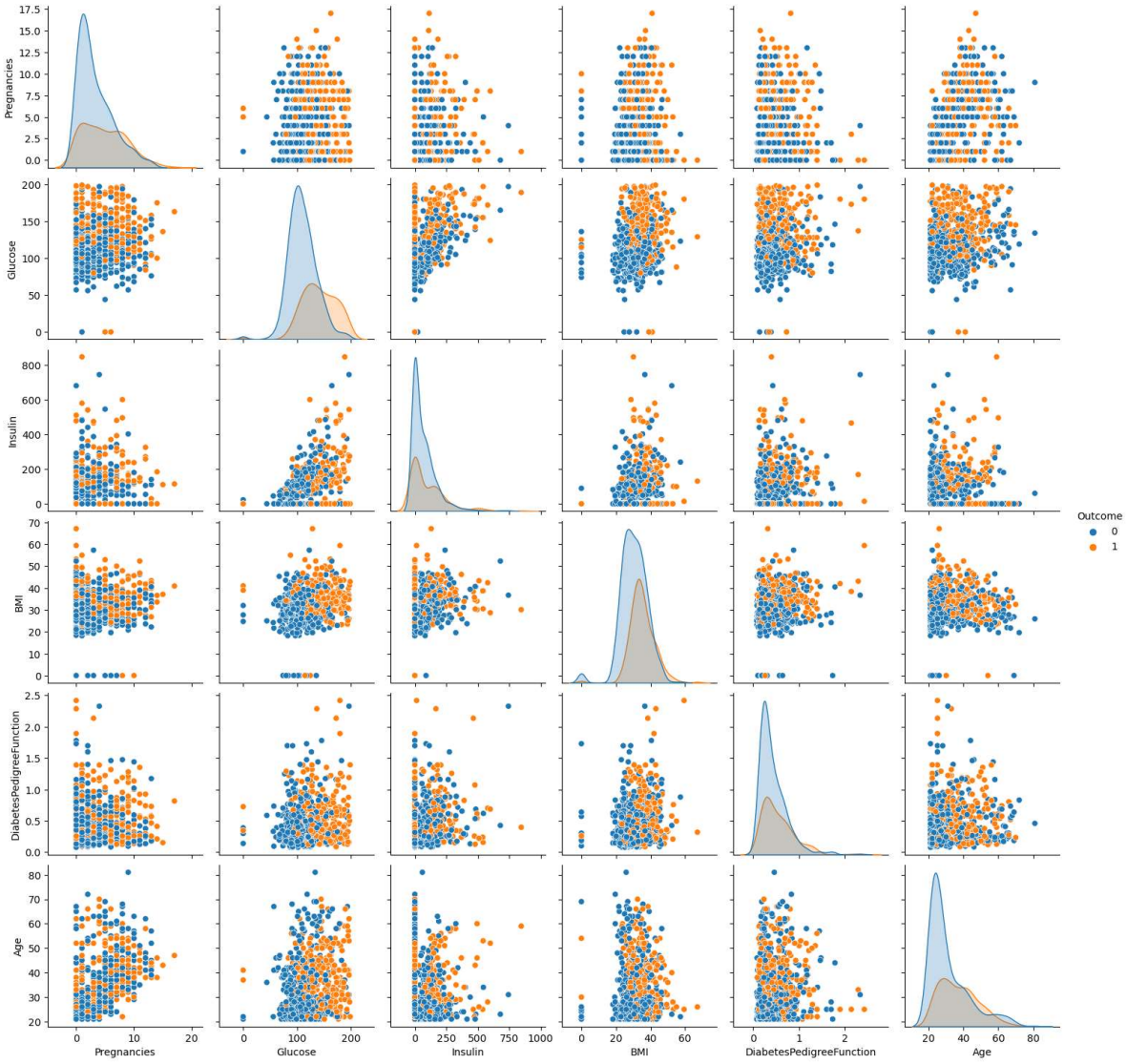
In [10]:

```
plt.figure(figsize=(16,16))
sns.boxplot(data=df);
```



In [11]:

```
sns.pairplot(data=df, hue="Outcome");
```



In [12]:

```
features=df.iloc[:,:-1]
```

In [13]:

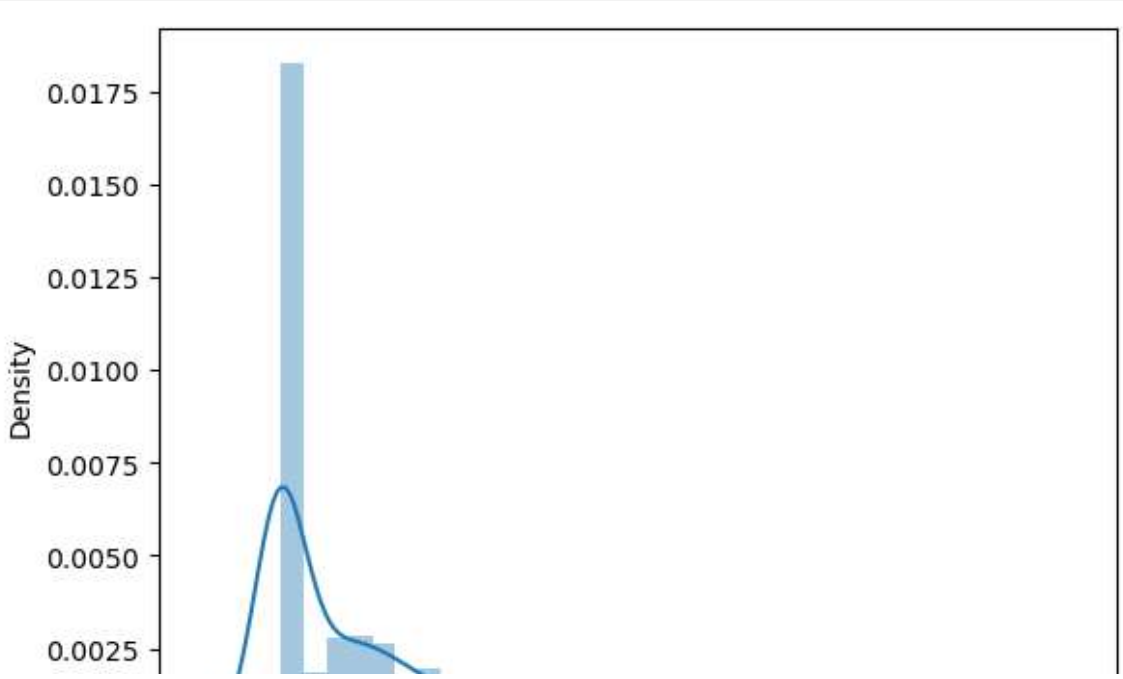
```
target=df.iloc[:,-1]
```

In [14]:

```
from scipy.stats import skew
```

In [15]:

```
for i in features:
    print(i)
    print(skew(features[i]))
    plt.figure()
    sns.distplot(features[i])
    plt.show()
```



In [16]:

```
features["Pregnancies"]=np.sqrt(features["Pregnancies"])
```

In [17]:

```
features["Glucose"]=np.sqrt(features["Glucose"])
```

In [18]:

```
features["Insulin"]=np.sqrt(features["Insulin"])
```

In [19]:

```
features["DiabetesPedigreeFunction"]=np.sqrt(features["DiabetesPedigreeFunction"])
```

In [20]:

```
features["Age"]=np.sqrt(features["Age"])
```

In [21]:

```
def whiskers(col):
    q1=np.quantile(col,0.25)
    q3=np.quantile(col,0.75)
    iqr=q3-q1
    uw=q3+1.5*iqr
    lw=q1-1.5*iqr
    return uw,lw
```

In [22]:

```
whiskers(df["Pregnancies"])
```

Out[22]:

```
(13.5, -6.5)
```

In [23]:

```
whiskers(df["Glucose"])
```

Out[23]:

```
(202.125, 37.125)
```

In [24]:

```
whiskers(df["Insulin"])
```

Out[24]:

```
(318.125, -190.875)
```

In [25]:

```
whiskers(df["BMI"])
```

Out[25]:

```
(50.550000000000004, 13.35)
```

In [26]:

```
whiskers(df["DiabetesPedigreeFunction"])
```

Out[26]:

```
(1.2, -0.32999999999999996)
```

In [27]:

```
whiskers(df["Age"])
```

Out[27]:

```
(66.5, -1.5)
```

In [28]:

```
a=df[df["Pregnancies"]>13.5].index
```

In [29]:

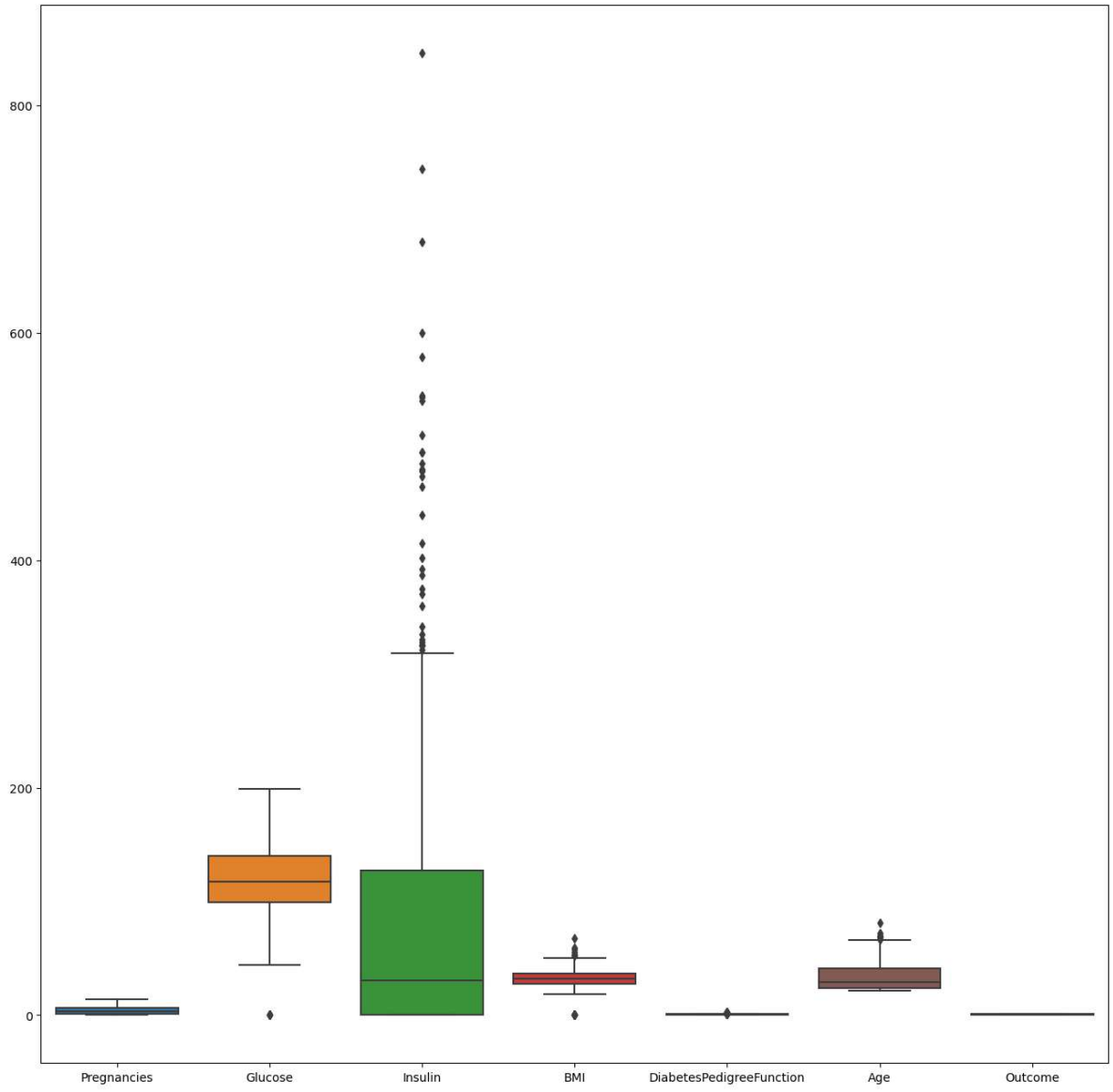
```
df.loc[a,"Pregnancies"]=13.5
```

In [30]:

```
plt.figure(figsize=(16,16))
sns.boxplot(data=df)
```

Out[30]:

```
<AxesSubplot:>
```



In [31]:

```
me=df["Glucose"].mean()
me
```

Out[31]:

```
120.89453125
```

In [32]:

```
b=df[df["Glucose"]<37.125].index
b
```

Out[32]:

```
Int64Index([75, 182, 342, 349, 502], dtype='int64')
```

In [33]:

```
df.loc[b,"Glucose"]=me
```

In [34]:

```
e=df[df["Insulin"]>318.125].index
```

In [35]:

```
df.loc[e,"Insulin"]=318.125
```

In [36]:

```
f=df[df["BMI"]>50.550000000000004].index
```

In [37]:

```
df.loc[f,"BMI"]=50.550000000000004
```

In [38]:

```
g=df[df["BMI"]< 13.35].index
```

In [39]:

```
df.loc[g,"BMI"]= 13.35
```

In [40]:

```
h=df[df["DiabetesPedigreeFunction"]>1.2].index
```

In [41]:

```
df.loc[h,"DiabetesPedigreeFunction"]=1.2
```

In [42]:

```
i=df[df["Age"]>66.5].index
```

In [43]:

```
df.loc[i,"Age"]=66.5
```

In [44]:

```
x=df.iloc[:,:-1]
```

In [45]:

```
y=df.iloc[:,-1]
```

In [46]:

```
from sklearn.preprocessing import StandardScaler
```

In [47]:

```
sd=StandardScaler()
x=sd.fit_transform(x)
x=pd.DataFrame(x)
x
```

Out[47]:

	0	1	2	3	4	5
0	0.647150	0.865276	-0.787602	0.209359	0.588927	1.445691
1	-0.848970	-1.205989	-0.787602	-0.784254	-0.378101	-0.189304
2	1.245598	2.015979	-0.787602	-1.252672	0.746595	-0.103252
3	-0.848970	-1.074480	0.217583	-0.571337	-1.022787	-1.049828
4	-1.148194	0.503626	1.008900	1.557835	2.596563	-0.017199
...	...	...	...	...	...	...
763	1.844045	-0.679954	1.137221	0.109998	-1.008772	2.564372
764	-0.549746	0.010468	-0.787602	0.663583	-0.416642	-0.533513
765	0.347926	-0.022409	0.410066	-0.841032	-0.749497	-0.275356
766	-0.848970	0.141977	-0.787602	-0.287447	-0.385109	1.187534
767	-0.848970	-0.942972	-0.787602	-0.244864	-0.504236	-0.877723

768 rows × 6 columns

In [48]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=1)
```

In [49]:

```
from sklearn.naive_bayes import GaussianNB,MultinomialNB,BernoulliNB
from sklearn.metrics import classification_report
from sklearn.svm import SVC
```

In [50]:

```
def pro(project):
    project.fit(xtrain,ytrain)
    ypred=project.predict(xtest)

    train=project.score(xtrain,ytrain)
    test=project.score(xtest,ytest)

    print(f"Training Acc:{train}\n Testing Acc:{test}\n\n")
    print(classification_report(ytest,ypred))
    return project
```

In [51]:

```
gnb=pro(GaussianNB())
```

```
Training Acc:0.750465549348231
 Testing Acc:0.8051948051948052


              precision    recall  f1-score   support

           0       0.82      0.88      0.85       146
           1       0.77      0.67      0.72        85

    accuracy                           0.81       231
   macro avg       0.80      0.78      0.78       231
weighted avg       0.80      0.81      0.80       231
```

In [52]:

```
bnb=pro(BernoulliNB())
```

```
Training Acc:0.7188081936685289
 Testing Acc:0.7619047619047619


              precision    recall  f1-score   support

           0       0.80      0.84      0.82       146
           1       0.69      0.64      0.66        85

    accuracy                           0.76       231
   macro avg       0.74      0.74      0.74       231
weighted avg       0.76      0.76      0.76       231
```

In [53]:

```
svm=SVC()
svm.fit(xtrain,ytrain)
ypred=svm.predict(xtest)
```

In [54]:

```
svm=pro(SVC())
```

```
Training Acc:0.8175046554934823
 Testing Acc:0.8008658008658008


              precision    recall  f1-score   support

           0       0.80      0.90      0.85       146
           1       0.79      0.62      0.70        85

    accuracy                           0.80       231
   macro avg       0.80      0.76      0.77       231
weighted avg       0.80      0.80      0.79       231
```

In [55]:

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       146
           1       0.79      0.62      0.70        85

    accuracy                           0.80       231
   macro avg       0.80      0.76      0.77       231
weighted avg       0.80      0.80      0.79       231
```

In [56]:

```
train=svm.score(xtrain,ytrain)
test=svm.score(xtest,ytest)
```

In [57]:

```
from sklearn.model_selection import GridSearchCV
```

In [58]:

```
parameter={
    "C":[0.1,1,10],
    "gamma":[0.1,0.01,0.001],
    "kernel":["rbf"]
}
```

In [59]:

```
grid=GridSearchCV(SVC(),parameter,verbose=3)
```

In [60]:

```
grid.fit(xtrain,ytrain)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
[CV 1/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.731 total time=
0.0s
[CV 2/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.713 total time=
0.0s
[CV 3/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.701 total time=
0.0s
[CV 4/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.738 total time=
0.0s
[CV 5/5] END ......C=0.1, gamma=0.1, kernel=rbf;, score=0.738 total time=
0.0s
[CV 1/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.657 total time=
0.0s
[CV 2/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.657 total time=
0.0s
[CV 3/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.654 total time=
0.0s
[CV 4/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.664 total time=
0.0s
[CV 5/5] END .....C=0.1, gamma=0.01, kernel=rbf;, score=0.664 total time=
0.0s
[CV 1/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.657 total time=
0.0s
[CV 2/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.657 total time=
0.0s
[CV 3/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.654 total time=
0.0s
[CV 4/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.664 total time=
0.0s
[CV 5/5] END ....C=0.1, gamma=0.001, kernel=rbf;, score=0.664 total time=
0.0s
[CV 1/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.750 total time=
0.0s
[CV 2/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.741 total time=
0.0s
[CV 3/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.794 total time=
0.0s
[CV 4/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.729 total time=
0.0s
[CV 5/5] END ........C=1, gamma=0.1, kernel=rbf;, score=0.785 total time=
0.0s
[CV 1/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.750 total time=
0.0s
[CV 2/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.778 total time=
0.0s
[CV 3/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.794 total time=
0.0s
[CV 4/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.729 total time=
0.0s
[CV 5/5] END .......C=1, gamma=0.01, kernel=rbf;, score=0.785 total time=
0.0s
[CV 1/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.657 total time=
0.0s
[CV 2/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.657 total time=
0.0s
[CV 3/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.654 total time=
0.0s
```

```
[CV 4/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.664 total time=
0.0s
[CV 5/5] END ......C=1, gamma=0.001, kernel=rbf;, score=0.664 total time=
0.0s
[CV 1/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.722 total time=
0.0s
[CV 2/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.685 total time=
0.0s
[CV 3/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.776 total time=
0.0s
[CV 4/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.710 total time=
0.0s
[CV 5/5] END .......C=10, gamma=0.1, kernel=rbf;, score=0.766 total time=
0.0s
[CV 1/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.750 total time=
0.0s
[CV 2/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.769 total time=
0.0s
[CV 3/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.794 total time=
0.0s
[CV 4/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.720 total time=
0.0s
[CV 5/5] END ......C=10, gamma=0.01, kernel=rbf;, score=0.785 total time=
0.0s
[CV 1/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.722 total time=
0.0s
[CV 2/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.787 total time=
0.0s
[CV 3/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.794 total time=
0.0s
[CV 4/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.738 total time=
0.0s
[CV 5/5] END .....C=10, gamma=0.001, kernel=rbf;, score=0.785 total time=
0.0s
```

Out[60]:

```
GridSearchCV(estimator=SVC(),
             param_grid={'C': [0.1, 1, 10], 'gamma': [0.1, 0.01, 0.001],
                         'kernel': ['rbf']},
             verbose=3)
```

In [61]:

```
grid.best_params_
```

Out[61]:

```
{'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}
```

In [62]:

```
grid.best_score_
```

Out[62]:

```
0.7672377985462098
```

In [63]:

```
grid.best_estimator_
```

Out[63]:

```
SVC(C=1, gamma=0.01)
```

In [64]:

```
svm=grid.best_estimator_
svm.fit(xtrain,ytrain)
ypred=svm.predict(xtest)
```

In [65]:

```
from sklearn.metrics import classification_report
print(classification_report(ytest,ypred))
```

```
              precision    recall  f1-score   support

           0       0.79      0.92      0.85       146
           1       0.80      0.58      0.67        85

    accuracy                           0.79       231
   macro avg       0.80      0.75      0.76       231
weighted avg       0.79      0.79      0.78       231
```

In [66]:

```
train=svm.score(xtrain,ytrain)
test=svm.score(xtest,ytest)
print(f"train acc:{train}\n test acc:{test}")
```

```
train acc:0.770949720670391
 test acc:0.7922077922077922
```

In [67]:

```
from sklearn.linear_model import LogisticRegression
```

In [68]:

```
lr=LogisticRegression()
```

In [69]:

```
lr.fit(xtrain,ytrain)
ypred=lr.predict(xtest)
```

In [70]:

```
train=lr.score(xtrain,ytrain)
test=lr.score(xtest,ytest)
print(f"train acc:{train}\n test acc:{test}")
```

```
train acc:0.7821229050279329
 test acc:0.7965367965367965
```

In [71]:

```
from sklearn.tree import DecisionTreeClassifier
```

In [72]:

```
dc=DecisionTreeClassifier(random_state=0)
dc.fit(xtrain,ytrain)
```

Out[72]:

```
DecisionTreeClassifier(random_state=0)
```

In [73]:

```
ypred=dc.predict(xtest)
```

In [74]:

```
train=dc.score(xtrain,ytrain)
test=dc.score(xtest,ytest)
print(f"train acc:{train}\n test acc:{test}")
```

```
train acc:1.0
 test acc:0.7402597402597403
```

In [ ]:

```

```

In [ ]:

```

```