naive bayes basic

February 20, 2020

[1]:
```
import numpy as np
import pandas as pd
import random
```

[2]:
```
from sklearn import datasets,metrics
```

[3]:
```
from sklearn.naive_bayes import GaussianNB
```

[4]:
```
from sklearn.model_selection import train_test_split
```

Basic Naive-Bayes performed:

[5]:
```
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)
```

[6]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.3)
```

[7]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(105, 4)
(45, 4)
(105,)
(45,)
```

[8]:
```
model=GaussianNB()
model.fit(x_train,y_train)
```

[8]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[9]:
```
y_predict=model.predict(x_test)
y_predict
```

[9]:
```
array([2, 2, 0, 1, 2, 0, 0, 2, 0, 1, 1, 2, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0,
       1, 0, 1, 0, 2, 2, 2, 2, 2, 2, 0, 0, 2, 1, 0, 1, 1, 1, 0, 2, 2, 1,
       1])
```

[10]:
```
print("Number of mislabeled points out of a total %d points : %d",(x_test.
 ↪shape[0], (y_test != y_predict).sum()))
```

```
Number of mislabeled points out of a total %d points : %d (45, 3)
```

[11]:
```
count=0
count_error=0
print(len(y_test))
print(len(y_predict))
print(len(y_test))
count=(y_test==y_predict).sum()
print(count)
print("the predict in percentage is :",(count/len(y_test)))
```

```
45
45
45
42
the predict in percentage is : 0.9333333333333333
```

[12]:
```
model.predict_proba(x_test)
```

[12]:
```
array([[6.15666196e-219, 1.12703696e-009, 9.99999999e-001],
       [1.35315837e-198, 1.13324899e-006, 9.99998867e-001],
       [1.00000000e+000, 1.69131041e-018, 2.98831847e-026],
       [2.83221253e-078, 9.99538870e-001, 4.61130046e-004],
       [2.18473264e-249, 4.43958559e-012, 1.00000000e+000],
       [1.00000000e+000, 9.49881287e-019, 1.08138364e-026],
       [1.00000000e+000, 1.42573614e-018, 6.26378013e-027],
       [2.65479629e-165, 3.06953042e-006, 9.99996930e-001],
       [1.00000000e+000, 4.17163911e-018, 2.60467870e-026],
       [2.94875851e-068, 9.99714154e-001, 2.85846424e-004],
       [2.80727767e-042, 9.99999088e-001, 9.12175960e-007],
       [1.34042396e-222, 2.28892445e-011, 1.00000000e+000],
       [2.16047294e-095, 9.76197876e-001, 2.38021239e-002],
       [1.00000000e+000, 1.27555095e-014, 7.67976199e-023],
       [3.53472902e-049, 9.99979012e-001, 2.09884427e-005],
       [2.36650363e-055, 9.99974344e-001, 2.56562415e-005],
       [2.25711684e-064, 9.99810720e-001, 1.89280318e-004],
       [1.00000000e+000, 2.73275186e-018, 4.41693284e-026],
       [1.51945588e-061, 9.99923137e-001, 7.68625723e-005],
       [1.00000000e+000, 8.26460866e-018, 4.22137187e-026],
       [3.02488164e-086, 9.92244834e-001, 7.75516573e-003],
       [1.00000000e+000, 8.81103610e-016, 2.24202527e-023],
       [3.99747955e-099, 9.06660671e-001, 9.33393292e-002],
       [1.00000000e+000, 2.58711824e-017, 2.19200358e-025],
       [5.53458507e-036, 9.99999935e-001, 6.52591630e-008],
```

```
       [1.00000000e+000, 6.86794610e-018, 2.86664229e-026],
       [1.14205531e-204, 2.10615962e-009, 9.99999998e-001],
       [7.03103100e-154, 1.26816451e-003, 9.98731835e-001],
       [4.20369138e-121, 6.03664195e-002, 9.39633581e-001],
       [5.08828922e-147, 2.10380951e-004, 9.99789619e-001],
       [1.60969387e-127, 3.86517286e-002, 9.61348271e-001],
       [8.31638608e-191, 1.00655912e-007, 9.99999899e-001],
       [1.00000000e+000, 8.16669244e-015, 9.37105590e-023],
       [1.00000000e+000, 1.05466554e-018, 1.34808591e-026],
       [5.24820543e-131, 1.74461785e-002, 9.82553821e-001],
       [9.20633020e-077, 9.96012742e-001, 3.98725843e-003],
       [1.00000000e+000, 7.95949016e-018, 1.09210730e-025],
       [1.12352686e-117, 5.43401329e-001, 4.56598671e-001],
       [4.92935564e-114, 6.61681156e-001, 3.38318844e-001],
       [1.05079091e-058, 9.99969307e-001, 3.06931659e-005],
       [1.00000000e+000, 3.24839379e-018, 6.89227133e-026],
       [6.13857131e-175, 1.40932173e-007, 9.99999859e-001],
       [1.93239644e-145, 1.65950534e-003, 9.98340495e-001],
       [9.92920006e-087, 9.89966802e-001, 1.00331980e-002],
       [1.52045932e-100, 6.77850463e-001, 3.22149537e-001]])
```

[13]:
```
model.predict_log_proba(x_test)
```

[13]:
```
array([[-5.02448601e+02, -2.06036738e+01, -1.12703624e-09],
       [-4.55609407e+02, -1.36904218e+01, -1.13324963e-06],
       [ 0.00000000e+00, -4.09210281e+01, -5.87725016e+01],
       [-1.78560579e+02, -4.61236399e-04, -7.68183046e+00],
       [-5.72562195e+02, -2.61404601e+01, -4.43911574e-12],
       [ 0.00000000e+00, -4.14979499e+01, -5.97889711e+01],
       [ 0.00000000e+00, -4.10918434e+01, -6.03350137e+01],
       [-3.78950172e+02, -1.26939860e+01, -3.06953513e-06],
       [ 0.00000000e+00, -4.00182226e+01, -5.89099031e+01],
       [-1.55494402e+02, -2.85887285e-04, -8.16005587e+00],
       [-9.56763587e+01, -9.12176376e-07, -1.39074329e+01],
       [-5.10880905e+02, -2.45003540e+01, -2.28892461e-11],
       [-2.17975257e+02, -2.40899712e-02, -3.73798046e+00],
       [-1.26565425e-14, -3.19928131e+01, -5.09208686e+01],
       [-1.11564033e+02, -2.09886630e-05, -1.07715386e+01],
       [-1.25780767e+02, -2.56565706e-05, -1.05707237e+01],
       [-1.46551358e+02, -1.89298234e-04, -8.57228148e+00],
       [ 0.00000000e+00, -4.04412226e+01, -5.83817669e+01],
       [-1.40039338e+02, -7.68655264e-05, -9.47349151e+00],
       [ 0.00000000e+00, -3.93345493e+01, -5.84270523e+01],
       [-1.96915446e+02, -7.78539341e-03, -4.85939611e+00],
       [-8.88178420e-16, -3.46653564e+01, -5.21520775e+01],
       [-2.26570260e+02, -9.79870214e-02, -2.37151373e+00],
       [ 0.00000000e+00, -3.81934020e+01, -5.67798113e+01],
```

```
       [-8.11820467e+01, -6.52591652e-08, -1.65448994e+01],
       [ 0.00000000e+00, -3.95196666e+01, -5.88140710e+01],
       [-4.69594529e+02, -1.99783996e+01, -2.10615969e-09],
       [-3.52647771e+02, -6.67018470e+00, -1.26896931e-03],
       [-2.77176833e+02, -2.80732230e+00, -6.22652877e-02],
       [-3.36853067e+02, -8.46659062e+00, -2.10403084e-04],
       [-2.91952263e+02, -3.25316378e+00, -3.94185305e-02],
       [-4.37675525e+02, -1.61115580e+01, -1.00655917e-07],
       [-8.21565038e-15, -3.24387124e+01, -5.07218314e+01],
       [ 0.00000000e+00, -4.13933080e+01, -5.95685267e+01],
       [-2.99980761e+02, -4.04863465e+00, -1.76001566e-02],
       [-1.75079161e+02, -3.99522874e-03, -5.52465139e+00],
       [ 0.00000000e+00, -3.93721667e+01, -5.74765182e+01],
       [-2.69285983e+02, -6.09907136e-01, -7.83950456e-01],
       [-2.60899492e+02, -4.12971476e-01, -1.08376650e+00],
       [-1.33500392e+02, -3.06936370e-05, -1.03914705e+01],
       [ 0.00000000e+00, -4.02683710e+01, -5.79368117e+01],
       [-4.01137799e+02, -1.57749871e+01, -1.40932182e-07],
       [-3.33216078e+02, -6.40123571e+00, -1.66088385e-03],
       [-1.98029423e+02, -1.00838697e-02, -4.60185588e+00],
       [-2.29839497e+02, -3.88828571e-01, -1.13273944e+00]])
```

[14]:
```
print(metrics.classification_report(y_predict,y_test))
confusion_metrics=metrics.confusion_matrix(y_predict,y_test)
print(confusion_metrics)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       0.94      0.88      0.91        17
           2       0.87      0.93      0.90        14

    accuracy                           0.93        45
   macro avg       0.93      0.94      0.94        45
weighted avg       0.93      0.93      0.93        45

[[14  0  0]
 [ 0 15  2]
 [ 0  1 13]]
```

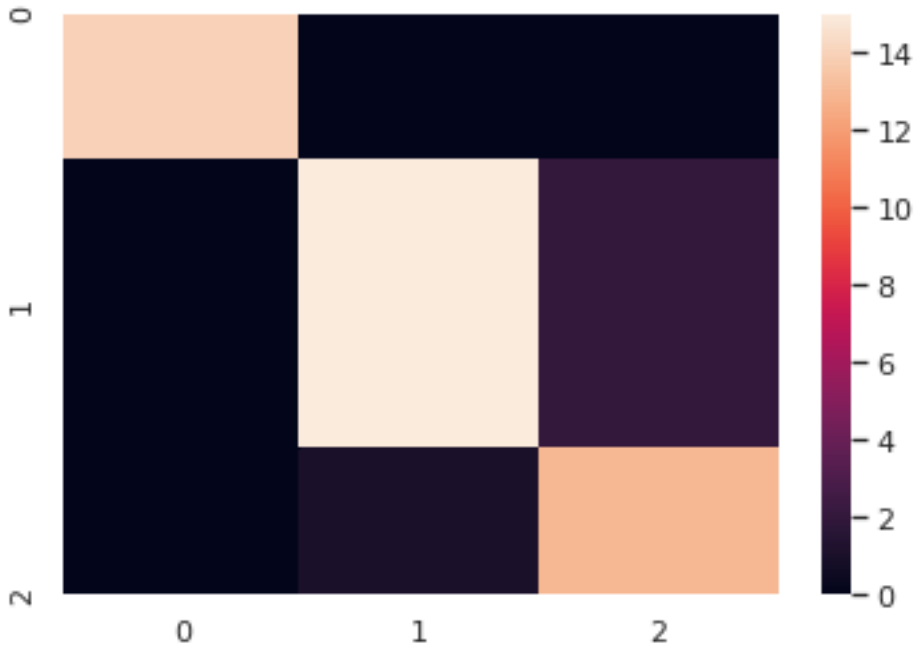
[15]:
```
import matplotlib.pyplot as plt
plt.imshow(confusion_metrics, cmap='binary',)
plt.show
```

[15]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```

[16]:
```
import seaborn as sns
```

```
/home/sushil/anaconda3/lib/python3.7/site-
packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is
deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[17]:
```
sns.set()
sns.heatmap(confusion_metrics)
plt.show()
```



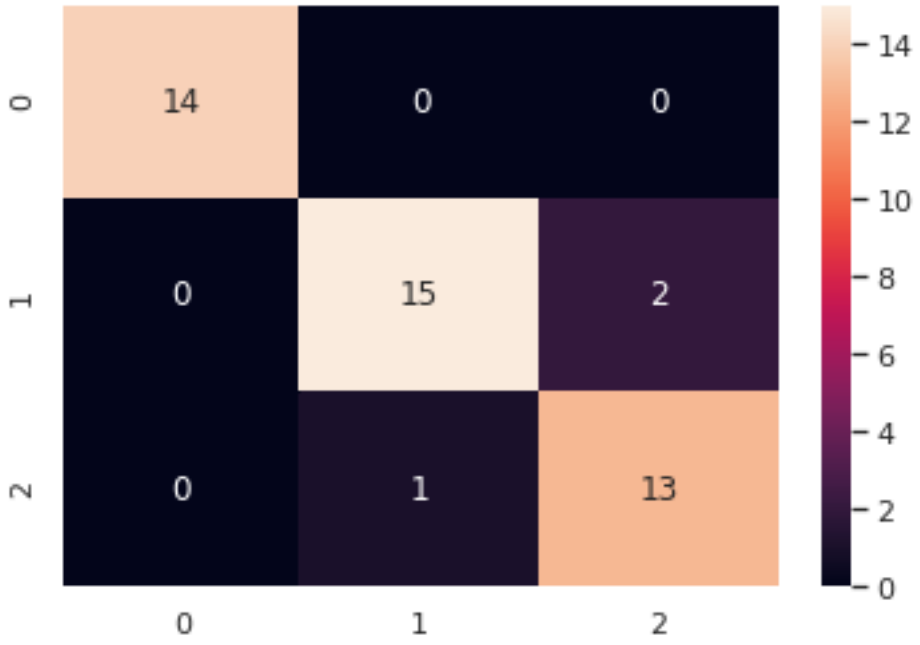
[18]:
```
df_cm = pd.DataFrame(confusion_metrics, range(3), range(3))
```

[19]:
```
df_cm
```

[19]:

	0	1	2
0	14	0	0
1	0	15	2
2	0	1	13

[32]:
```
ax=sns.heatmap(df_cm,annot=True,annot_kws={"size": 12})
sns.set(font_scale=1.1)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()
```



[24]:
```
import matplotlib
matplotlib.__version__
```

[24]: '3.1.1'

[158]:
```
datasets=pd.read_csv("/home/sushil/Downloads/databases/diabetes.csv")
```

[159]:
```
print(datasets.head())
print(datasets.dtypes)
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
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

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
```

```
SkinThickness                int64
Insulin                      int64
BMI                        float64
DiabetesPedigreeFunction   float64
Age                          int64
Outcome                      int64
dtype: object
```

[160]:
```
datasets.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
```

[161]:
```
#df['DataFrame Column'] = df['DataFrame Column'].astype(float)
#df['DataFrame Column'] = pd.to_numeric(df['DataFrame Column'],errors='coerce')
```

[162]:
```
datasets.shape
```

[162]:
```
(768, 9)
```

[163]:
```
datasets.iloc[:,:] = datasets.iloc[:,:].astype(float)
```

[164]:
```
datasets.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    float64
 1   Glucose                   768 non-null    float64
 2   BloodPressure             768 non-null    float64
 3   SkinThickness             768 non-null    float64
 4   Insulin                   768 non-null    float64
```

```
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    float64
 8   Outcome                   768 non-null    float64
dtypes: float64(9)
memory usage: 54.1 KB
```

the datasets is converted to floats value

[113]:
```
#alternate method of splitind data
df = pd.DataFrame(np.random.randn(100, 2))

msk = np.random.rand(len(df)) < 0.8

train = df[msk]

test = df[~msk]

print(train.shape)
print(test.shape)
```

```
(80, 2)
(20, 2)
```

[133]:
```
msk = np.random.rand(len(datasets)) < 0.8
train=datasets[msk]
print(train.shape)
test=datasets[~msk]
print(test.shape)
print("*"*100)
train
x_train=train.drop(columns="Outcome")
print(x_train.head(1))
print("*"*100)
y_train=train.Outcome
print(y_train.head(2))
print("*"*100)
x_test=test.drop(columns="Outcome")
print(x_test.shape)
y_test=test.Outcome
print("*"*100)
print(y_test.shape)
```

```
(617, 9)
(151, 9)
********************************************************************************
********************
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0          6.0    148.0           72.0           35.0      0.0  33.6
```

```
   DiabetesPedigreeFunction   Age
0                     0.627  50.0
********************************************************************************
********************
0    1.0
1    0.0
Name: Outcome, dtype: float64
********************************************************************************
********************
(151, 8)
********************************************************************************
********************
(151,)
```

[169]:
```
from sklearn.model_selection import train_test_split
```

[172]:
```
X=datasets.drop(columns="Outcome")
X.head()
y=datasets.Outcome
y.head()
```

[172]:
```
0    1.0
1    0.0
2    1.0
3    0.0
4    1.0
Name: Outcome, dtype: float64
```

[173]:
```
y.valuescounts()
```

[173]:
```
0.0    500
1.0    268
Name: Outcome, dtype: int64
```

[174]:
```
#hence we can say that our datasets is imbalanced datasets
```

[175]:
```
x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.30)
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(537, 8)
(231, 8)
(537,)
(231,)
```

[176]:
```
from sklearn.preprocessing import StandardScaler
#df = StandardScaler().fit_transform(df[['cost', 'sales']])
#here we are standardizing our data so that i wont be affected by scale
```

[177]:
```
scalar=StandardScaler()
x_train=scalar.fit_transform(x_train)
x_test=scalar.fit_transform(x_test)
```

[181]:
```
print(x_train.shape)
print(x_test.shape)
print(type(x_train))
```

```
(537, 8)
(231, 8)
<class 'numpy.ndarray'>
```

[182]:
```
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
```

[183]:
```
from sklearn.naive_bayes import GaussianNB
```

[184]:
```
from sklearn.naive_bayes import BernoulliNB
```

[185]:
```
model=GaussianNB()
```

[186]:
```
model.fit(x_train,y_train)
```

[186]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

[188]:
```
y_predict=model.predict(x_test)
y_predict
```

[188]:
```
array([0., 1., 1., 0., 0., 1., 1., 1., 0., 1., 0., 0., 1., 0., 1., 0., 0.,
       1., 1., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1., 0., 0., 0., 0.,
       0., 1., 0., 1., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       1., 0., 0., 1., 0., 0., 1., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0.,
       1., 0., 0., 0., 0., 1., 1., 0., 0., 0., 0., 0., 1., 1., 1., 0., 0.,
       0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 1., 1., 0., 0.,
       0., 0., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 0., 1., 1., 0., 0.,
       0., 1., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 1., 1.,
       0., 0., 0., 1., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1.,
       0., 1., 0., 0., 1., 0., 0., 1., 1., 0., 0., 0., 0., 0., 0., 0., 1.,
       0., 1., 0., 0., 0., 0., 1., 0., 1., 0., 1., 1., 1., 0., 1., 1., 0.,
       1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 1.,
       0., 0., 0., 0., 0., 0., 0., 0., 1., 1.])
```

[189]:
```
y_predict_pro=model.predict_proba(x_test)
```

[190]:
```
y_predict_pro
```

[190]:
```
array([[7.41655011e-01, 2.58344989e-01],
       [3.46458262e-01, 6.53541738e-01],
       [9.45829239e-02, 9.05417076e-01],
       [9.46178272e-01, 5.38217281e-02],
       [9.25183109e-01, 7.48168908e-02],
       [3.06806022e-01, 6.93193978e-01],
       [7.10437603e-02, 9.28956240e-01],
       [3.10692190e-01, 6.89307810e-01],
       [6.28222738e-01, 3.71777262e-01],
       [2.55899946e-01, 7.44100054e-01],
       [9.42339879e-01, 5.76601213e-02],
       [5.99424773e-01, 4.00575227e-01],
       [1.04989121e-01, 8.95010879e-01],
       [9.48632544e-01, 5.13674563e-02],
       [1.51476507e-01, 8.48523493e-01],
       [9.48192062e-01, 5.18079377e-02],
       [7.70087069e-01, 2.29912931e-01],
       [1.46984416e-01, 8.53015584e-01],
       [3.02398852e-01, 6.97601148e-01],
       [7.81904412e-01, 2.18095588e-01],
       [8.63924860e-01, 1.36075140e-01],
       [9.43135266e-01, 5.68647340e-02],
       [9.78894770e-01, 2.11052302e-02],
       [8.60342117e-01, 1.39657883e-01],
       [6.62603048e-01, 3.37396952e-01],
       [3.52079275e-01, 6.47920725e-01],
       [4.33420356e-01, 5.66579644e-01],
       [2.41982472e-02, 9.75801753e-01],
       [1.68300563e-01, 8.31699437e-01],
       [3.50208259e-01, 6.49791741e-01],
       [7.71992855e-01, 2.28007145e-01],
       [7.45385146e-01, 2.54614854e-01],
       [8.31676826e-01, 1.68323174e-01],
       [9.86553591e-01, 1.34464092e-02],
       [9.58945727e-01, 4.10542728e-02],
       [8.25353244e-02, 9.17464676e-01],
       [9.77270643e-01, 2.27293574e-02],
       [3.89712758e-02, 9.61028724e-01],
       [2.94223780e-01, 7.05776220e-01],
       [7.10738292e-05, 9.99928926e-01],
       [9.30553347e-01, 6.94466534e-02],
       [9.50828161e-01, 4.91718392e-02],
       [9.16662197e-01, 8.33378031e-02],
```

```
[9.43536716e-01, 5.64632838e-02],
[9.37215088e-01, 6.27849119e-02],
[8.34970931e-01, 1.65029069e-01],
[9.52306116e-01, 4.76938840e-02],
[1.92354934e-03, 9.98076451e-01],
[7.75554869e-01, 2.24445131e-01],
[8.34270958e-02, 9.16572904e-01],
[7.97380537e-01, 2.02619463e-01],
[9.84075414e-01, 1.59245863e-02],
[9.45039710e-01, 5.49602902e-02],
[6.96892562e-01, 3.03107438e-01],
[8.35281655e-01, 1.64718345e-01],
[9.64424478e-01, 3.55755224e-02],
[9.09908361e-01, 9.00916390e-02],
[8.70397555e-01, 1.29602445e-01],
[8.97698645e-01, 1.02301355e-01],
[9.36828767e-01, 6.31712331e-02],
[9.79464544e-01, 2.05354555e-02],
[9.93049733e-01, 6.95026658e-03],
[9.16122671e-01, 8.38773286e-02],
[8.58577733e-01, 1.41422267e-01],
[4.74859162e-01, 5.25140838e-01],
[8.03738191e-01, 1.96261809e-01],
[9.85617663e-01, 1.43823366e-02],
[8.99104759e-01, 1.00895241e-01],
[1.30992684e-01, 8.69007316e-01],
[9.59229544e-01, 4.07704563e-02],
[9.85516358e-01, 1.44836422e-02],
[4.34087531e-03, 9.95659125e-01],
[8.65858026e-01, 1.34141974e-01],
[9.85981247e-01, 1.40187531e-02],
[3.96144104e-03, 9.96038559e-01],
[4.71533363e-01, 5.28466637e-01],
[5.79240445e-01, 4.20759555e-01],
[5.13414160e-02, 9.48658584e-01],
[9.32321460e-01, 6.76785404e-02],
[9.55988492e-01, 4.40115080e-02],
[6.38142741e-01, 3.61857259e-01],
[9.86395284e-01, 1.36047157e-02],
[8.20635483e-01, 1.79364517e-01],
[7.96664598e-01, 2.03335402e-01],
[9.73608596e-01, 2.63914041e-02],
[2.40441068e-03, 9.97595589e-01],
[9.85909666e-01, 1.40903338e-02],
[9.87475373e-01, 1.25246275e-02],
[6.85913052e-01, 3.14086948e-01],
[8.26834717e-01, 1.73165283e-01],
```

```
[9.08346618e-02, 9.09165338e-01],
[4.50451494e-01, 5.49548506e-01],
[8.04209274e-01, 1.95790726e-01],
[9.13364024e-01, 8.66359758e-02],
[9.10909075e-01, 8.90909247e-02],
[7.62675567e-01, 2.37324433e-01],
[8.89565293e-01, 1.10434707e-01],
[7.13391803e-02, 9.28660820e-01],
[3.39249590e-01, 6.60750410e-01],
[2.44166757e-02, 9.75583324e-01],
[6.78520130e-01, 3.21479870e-01],
[7.80225891e-01, 2.19774109e-01],
[9.41792796e-01, 5.82072037e-02],
[9.87026559e-01, 1.29734412e-02],
[9.49355154e-01, 5.06448456e-02],
[9.51480709e-01, 4.85192906e-02],
[9.94184778e-01, 5.81522207e-03],
[1.66707037e-01, 8.33292963e-01],
[9.74520071e-01, 2.54799291e-02],
[7.06503094e-01, 2.93496906e-01],
[9.52979464e-01, 4.70205365e-02],
[5.49450774e-01, 4.50549226e-01],
[2.34781412e-01, 7.65218588e-01],
[9.82288159e-01, 1.77118409e-02],
[8.62721571e-01, 1.37278429e-01],
[2.85440094e-01, 7.14559906e-01],
[3.42405627e-01, 6.57594373e-01],
[9.88245407e-01, 1.17545932e-02],
[6.15622157e-01, 3.84377843e-01],
[9.30566112e-01, 6.94338876e-02],
[9.83420519e-01, 1.65794806e-02],
[8.80616378e-01, 1.19383622e-01],
[9.83432241e-01, 1.65677592e-02],
[7.37783568e-01, 2.62216432e-01],
[2.52953862e-06, 9.99997470e-01],
[1.30018196e-03, 9.98699818e-01],
[2.83585781e-01, 7.16414219e-01],
[1.06610701e-02, 9.89338930e-01],
[5.68044899e-01, 4.31955101e-01],
[2.34156288e-02, 9.76584371e-01],
[2.76023134e-01, 7.23976866e-01],
[6.82756583e-01, 3.17243417e-01],
[1.73649391e-03, 9.98263506e-01],
[1.53903824e-01, 8.46096176e-01],
[8.57747629e-01, 1.42252371e-01],
[9.52968715e-01, 4.70312846e-02],
[9.72146575e-01, 2.78534254e-02],
```

```
[3.31442189e-01, 6.68557811e-01],
[9.57516504e-01, 4.24834959e-02],
[8.65388430e-01, 1.34611570e-01],
[6.75418354e-03, 9.93245816e-01],
[9.89078598e-01, 1.09214016e-02],
[9.87156495e-01, 1.28435050e-02],
[7.80415013e-01, 2.19584987e-01],
[8.51417501e-01, 1.48582499e-01],
[9.50368723e-01, 4.96312773e-02],
[9.89378475e-01, 1.06215250e-02],
[9.17902234e-01, 8.20977661e-02],
[7.33502431e-01, 2.66497569e-01],
[5.62643677e-03, 9.94373563e-01],
[9.81218757e-01, 1.87812433e-02],
[1.19932715e-01, 8.80067285e-01],
[4.65161749e-01, 5.34838251e-01],
[9.79430217e-01, 2.05697826e-02],
[8.92882078e-01, 1.07117922e-01],
[8.41077810e-01, 1.58922190e-01],
[4.37550027e-01, 5.62449973e-01],
[9.06998580e-01, 9.30014196e-02],
[8.23150297e-02, 9.17684970e-01],
[5.23768543e-01, 4.76231457e-01],
[7.93376041e-01, 2.06623959e-01],
[8.94102632e-01, 1.05897368e-01],
[9.58962384e-01, 4.10376155e-02],
[9.62707912e-01, 3.72920880e-02],
[5.21007933e-01, 4.78992067e-01],
[9.91466705e-01, 8.53329477e-03],
[7.72753754e-01, 2.27246246e-01],
[9.64175109e-01, 3.58248908e-02],
[2.11483229e-01, 7.88516771e-01],
[5.22820496e-02, 9.47717950e-01],
[7.16932021e-01, 2.83067979e-01],
[3.05264808e-01, 6.94735192e-01],
[9.88503257e-01, 1.14967430e-02],
[6.40535037e-01, 3.59464963e-01],
[2.98288479e-01, 7.01711521e-01],
[9.88773262e-01, 1.12267379e-02],
[9.35415962e-01, 6.45840378e-02],
[1.07972102e-01, 8.92027898e-01],
[5.90327852e-02, 9.40967215e-01],
[9.84503799e-01, 1.54962011e-02],
[9.89642271e-01, 1.03577291e-02],
[6.73686891e-01, 3.26313109e-01],
[9.77353029e-01, 2.26469710e-02],
[8.52011851e-01, 1.47988149e-01],
```

```
[9.05421614e-01, 9.45783857e-02],
[9.42949278e-01, 5.70507220e-02],
[6.59307632e-02, 9.34069237e-01],
[9.77786347e-01, 2.22136535e-02],
[4.62602973e-01, 5.37397027e-01],
[7.05975781e-01, 2.94024219e-01],
[9.80627195e-01, 1.93728054e-02],
[7.62575244e-01, 2.37424756e-01],
[9.50687816e-01, 4.93121840e-02],
[4.60820796e-03, 9.95391792e-01],
[6.41213516e-01, 3.58786484e-01],
[2.32392530e-03, 9.97676075e-01],
[9.57775538e-01, 4.22244621e-02],
[1.56258995e-01, 8.43741005e-01],
[6.91408879e-03, 9.93085911e-01],
[5.35688734e-02, 9.46431127e-01],
[9.10986714e-01, 8.90132863e-02],
[3.48538110e-02, 9.65146189e-01],
[1.26336666e-01, 8.73663334e-01],
[9.49668918e-01, 5.03310820e-02],
[4.29515979e-01, 5.70484021e-01],
[7.69138654e-01, 2.30861346e-01],
[9.84883234e-01, 1.51167661e-02],
[9.88256234e-01, 1.17437656e-02],
[8.36680030e-01, 1.63319970e-01],
[2.02970465e-01, 7.97029535e-01],
[6.15057901e-01, 3.84942099e-01],
[9.69466336e-01, 3.05336643e-02],
[9.26080856e-01, 7.39191436e-02],
[9.88837458e-01, 1.11625422e-02],
[1.77419169e-01, 8.22580831e-01],
[9.68700627e-01, 3.12993733e-02],
[7.57789314e-01, 2.42210686e-01],
[9.90105729e-01, 9.89427115e-03],
[9.54580657e-01, 4.54193431e-02],
[8.12860347e-01, 1.87139653e-01],
[1.62129070e-01, 8.37870930e-01],
[7.55644589e-01, 2.44355411e-01],
[9.69096878e-01, 3.09031222e-02],
[9.82993099e-01, 1.70069011e-02],
[9.13830526e-01, 8.61694743e-02],
[5.41187247e-01, 4.58812753e-01],
[9.72655447e-01, 2.73445533e-02],
[6.05418147e-01, 3.94581853e-01],
[9.62262221e-01, 3.77377793e-02],
[2.53358190e-01, 7.46641810e-01],
[6.35679948e-02, 9.36432005e-01]])
```

in above predict_probability array, it is comparing the value (0:7.41655011e-01) and (1:2.58344989e-01) so here we can see the probability of 0 is high hence the patience has no diabetes

[197]:
```
#print("the probability of correct ans is ",)
print("the total length of y_test:",len(y_test))
count=(y_test==y_predict).sum()
print("total correct y_predicted:",count)
print("the predict in percentage is :",(count/len(y_test)))
```

```
the total length of y_test: 231
total correct y_predicted: 169
the predict in percentage is : 0.7316017316017316
```

[198]:
```
from sklearn import metrics
```

[199]:
```
from sklearn.metrics import accuracy_score
```

[208]:
```
accuracy=accuracy_score(y_test,y_predict)
classification_reports=metrics.classification_report(y_test,y_predict)
print("the accuracy :",accuracy)
print("+"*50)
print("the classification reports",classification_reports)
confusion_mat=metrics.confusion_matrix(y_test,y_predict,labels=[0,1])
print("+"*50)
print("the confusion matrix:");print(confusion_mat)
```

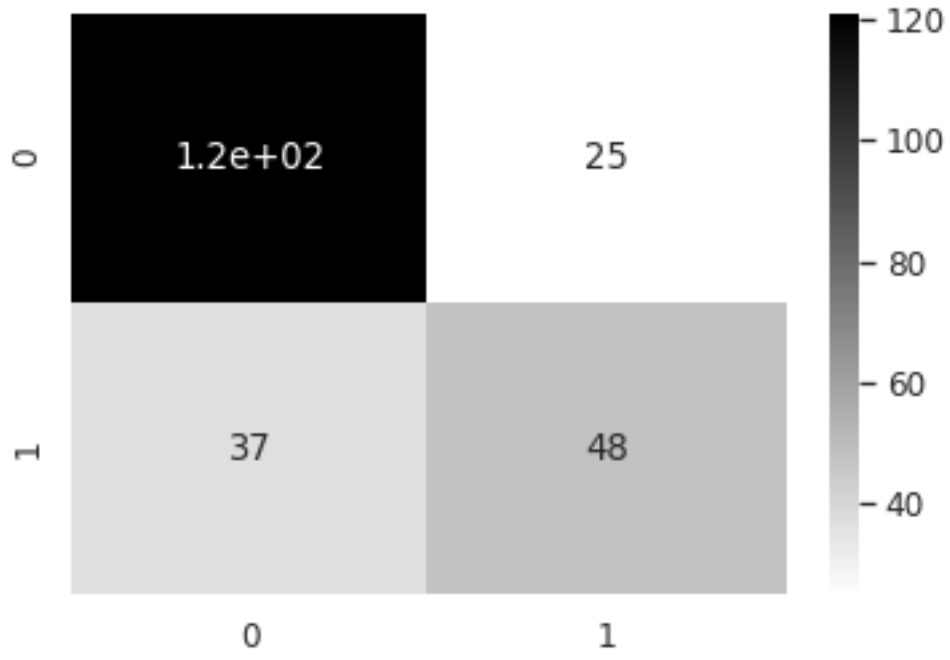
```
the accuracy : 0.7316017316017316
++++++++++++++++++++++++++++++++++++++++++++++++++
the classification reports              precision    recall  f1-score   support

         0.0       0.77      0.83      0.80       146
         1.0       0.66      0.56      0.61        85

    accuracy                           0.73       231
   macro avg       0.71      0.70      0.70       231
weighted avg       0.73      0.73      0.73       231

++++++++++++++++++++++++++++++++++++++++++++++++++
the confusion matrix:
[[121  25]
 [ 37  48]]
```

[212]:
```
ax=sns.heatmap(confusion_mat,cmap="binary",annot=True)
sns.set(font_scale=1.1)
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
plt.show()
```


0
1
1.2e+02
25
37
48
0
1
120
100
80
60
40

[ ]: