CH-6 Classification Analysis Using Python

KNN(K Nearest Neighbor) : -

In [14]:
```
import pandas as pd
df = pd.read_csv("KNN.csv")
df
```

Out[14]:

	points	x1	x2	class
0	A	1	2	0
1	B	2	3	0
2	C	3	1	1
3	D	6	5	1

In [15]:
```
df = df.drop("points",axis=1)
df
```

Out[15]:

	x1	x2	class
0	1	2	0
1	2	3	0
2	3	1	1
3	6	5	1

In [16]:
```
y = df["class"]
x = df.drop("class",axis=1)
df
```

Out[16]:

	x1	x2	class
0	1	2	0
1	2	3	0
2	3	1	1
3	6	5	1

In [27]:
```
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,random_stat
x_train
```

Out[27]:

	x1	x2
3	6	5
1	2	3
0	1	2

In [37]:
```
from sklearn.neighbors import KNeighborsClassifier
nn = KNeighborsClassifier(n_neighbors=1)
model = nn.fit(x_train,y_train)
```

In [38]:
```
y_pred = model.predict(x_test)
```

In [39]:
```
y_pred
```

Out[39]:
```
array([0], dtype=int64)
```

In [40]:
```
## Confusion Metrics : -
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[40]:
```
array([[0, 0],
       [1, 0]], dtype=int64)
```

In [48]:
```
TN = cm[0][0] # 0
FP = cm[0][1] # 0
FN = cm[1][0] # 1
TP = cm[1][1] # 0
print(TN)
print(FP)
print(FN)
print(TP)
```

```
0
0
1
0
```

In [42]:
```
accuracy = (TP+TN)/(TP+TN+FP+FN)
accuracy
```

Out[42]:
```
0.0
```

In [43]:
```
error_rate = (FP+FN)/(TP+TN+FP+FN)
error_rate
```

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

In [44]:
```
error_rate = 1-accuracy
error_rate
```

Out[44]:
```
1.0
```

In [45]:
```
sensitivity = TP/(TP+FN)
sensitivity
```

Out[45]:
```
0.0
```

In [46]:
```
specificity = (TN)/(TN+FP)
specificity
```

```
<ipython-input-46-b01751f7ed08>:1: RuntimeWarning: invalid value encountered
in longlong_scalars
  specificity = (TN)/(TN+FP)
```

Out[46]:
```
nan
```

Diabities

In [49]:
```
import pandas as pd
df = pd.read_csv("diabetes.csv")
df
```

Out[49]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

In [51]:
```
y = df["Outcome"]
x = df.drop("Outcome",axis=1)
df
```

Out[51]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

In [52]:
```
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,random_stat
x_train
```

Out[52]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
602	1	124	74	36	0	27.8	0.1
429	1	95	82	25	180	35.0	0.2
623	0	94	70	27	115	43.5	0.3
209	7	184	84	33	0	35.5	0.3
589	0	73	0	0	0	21.1	0.3
...	...	...	...	...	...	...	
534	1	77	56	30	56	33.3	1.2
584	8	124	76	24	600	28.7	0.6
493	4	125	70	18	122	28.9	1.1
527	3	116	74	15	105	26.3	0.1
168	4	110	66	0	0	31.9	0.4

614 rows × 8 columns

In [63]:
```
from sklearn.neighbors import KNeighborsClassifier
nn = KNeighborsClassifier(n_neighbors=27)
model = nn.fit(x_train,y_train)
```

In [64]:
```
y_pred = model.predict(x_test)
```

In [65]:
```
y_pred
```

Out[65]:
```
array([0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

Confusion_matrix

In [66]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[66]:
```
array([[98, 11],
       [24, 21]], dtype=int64)
```

In [67]:
```
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]
print(TN)
print(FP)
print(FN)
print(TP)
```

```
98
11
24
21
```

In [68]:
```
accuracy = (TP+TN)/(TP+TN+FP+FN)
accuracy
```

Out[68]:
```
0.7727272727272727
```

In [69]:
```
error_rate = (FP+FN)/(TP+TN+FP+FN)
error_rate
```

Out[69]:
```
0.22727272727272727
```

In [70]:
```
error_rate = 1-accuracy
error_rate
```

Out[70]:
```
0.2272727272727273
```

In [71]:
```
sensitivity = TP/(TP+FN)
sensitivity
```

Out[71]:
```
0.4666666666666667
```

In [62]:
```
specificity = (TN)/(TN+FP)
specificity
```

Out[62]:
```
0.7431192660550459
```

Last Code

In [1]:
```
import pandas as pd
df = pd.read_csv("diabetes.csv")
df
```

Out[1]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 9 columns

In [3]:
```
y = df["Outcome"]
x = df.drop("Outcome",axis=1)
x
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
0	6	148	72	35	0	33.6	0.6
1	1	85	66	29	0	26.6	0.3
2	8	183	64	0	0	23.3	0.6
3	1	89	66	23	94	28.1	0.1
4	0	137	40	35	168	43.1	2.2
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	0.1
764	2	122	70	27	0	36.8	0.3
765	5	121	72	23	112	26.2	0.2
766	1	126	60	0	0	30.1	0.3
767	1	93	70	31	0	30.4	0.3

768 rows × 8 columns

In [4]:
```
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,random_stat
x_train
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFuncti
602	1	124	74	36	0	27.8	0.1
429	1	95	82	25	180	35.0	0.2
623	0	94	70	27	115	43.5	0.3
209	7	184	84	33	0	35.5	0.3
589	0	73	0	0	0	21.1	0.3
...	...	...	...	...	...	...	
534	1	77	56	30	56	33.3	1.2
584	8	124	76	24	600	28.7	0.6
493	4	125	70	18	122	28.9	1.1
527	3	116	74	15	105	26.3	0.1
168	4	110	66	0	0	31.9	0.4

614 rows × 8 columns

In [18]:
```
from sklearn.tree import DecisionTreeClassifier
nn = DecisionTreeClassifier(criterion="entropy",max_depth=27)
```

In [19]:
```
model = nn.fit(x_train,y_train)
```

In [20]:
```
y_pred = nn.predict(x_test)
y_pred
```

Out[20]:
```
array([0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0],
      dtype=int64)
```

In [21]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred)
cm
```

Out[21]:
```
array([[80, 29],
       [23, 22]], dtype=int64)
```

In [22]:
```
TN = cm[0][0]
FP = cm[0][1]
FN = cm[1][0]
TP = cm[1][1]
print(TN)
print(FP)
print(FN)
print(TP)
```

```
80
29
23
22
```

In [24]:
```
accuracy = (TP+TN)/(TP+TN+FP+FN)
accuracy
```

Out[24]:
```
0.6623376623376623
```

In [25]:
```
error_rate = (FP+FN)/(TP+TN+FP+FN)
error_rate
```

Out[25]:
```
0.33766233766233766
```

In [26]:
```
error_rate = 1-accuracy
error_rate
```

Out[26]:
```
0.33766233766233766
```

In [27]:
```
sensitivity = TP/(TP+FN)
sensitivity
```

Out[27]:
```
0.4888888888888889
```

In [28]:
```
specificity = (TN)/(TN+FP)
specificity
```

Out[28]:
```
0.7339449541284404
```

In [30]:
```
from sklearn import tree
gr = tree.export_text(nn)
print(gr)
```

```
|   |   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- feature_7 >  39.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- feature_6 >  0.31
|   |   |   |   |   |   |   |--- feature_5 <= 45.30
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |--- feature_5 >  45.30
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- feature_7 >  66.50
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature_3 >  32.50
|   |   |   |   |   |--- class: 1
|   |   |   |--- feature_6 >  2.23
|   |   |   |   |--- class: 0
|   |   |--- feature_2 >  93.00
|   |   |   |--- feature_0 <= 7.50
|   |   |   |   |--- feature_0 <= 2.00
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- feature_0 >  2.00
```

In [31]:
```
pre = [[10,20,30,40,50,60,70,80]]
test_data =nn.predict(pre)
test_data
```

Out[31]:
```
array([0], dtype=int64)
```

In [ ]: