```
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Datasets/cancer_data.csv')
df
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smoothness	diagnosis
0	17.99	10.38	122.80	1001.0	0.11840	0
1	20.57	17.77	132.90	1326.0	0.08474	0
2	19.69	21.25	130.00	1203.0	0.10960	0
3	11.42	20.38	77.58	386.1	0.14250	0
4	20.29	14.34	135.10	1297.0	0.10030	0
...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0
565	20.13	28.25	131.20	1261.0	0.09780	0
566	16.60	28.08	108.30	858.1	0.08455	0
567	20.60	29.33	140.10	1265.0	0.11780	0
568	7.76	24.54	47.92	181.0	0.05263	1

569 rows × 6 columns

```
df.shape
```

```
(569, 6)
```

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

```
mean_radius        0
mean_texture       0
mean_perimeter     0
mean_area          0
mean_smoothness    0
diagnosis          0
dtype: int64
```

```
df.dtypes
```

```
mean_radius        float64
mean_texture       float64
mean_perimeter     float64
mean_area          float64
mean_smoothness    float64
diagnosis            int64
dtype: object
```

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

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

```
x
```

```
array([[1.799e+01, 1.038e+01, 1.228e+02, 1.001e+03, 1.184e-01],
       [2.057e+01, 1.777e+01, 1.329e+02, 1.326e+03, 8.474e-02],
       [1.969e+01, 2.125e+01, 1.300e+02, 1.203e+03, 1.096e-01],
       ...,
       [1.660e+01, 2.808e+01, 1.083e+02, 8.581e+02, 8.455e-02],
       [2.060e+01, 2.933e+01, 1.401e+02, 1.265e+03, 1.178e-01],
       [7.760e+00, 2.454e+01, 4.792e+01, 1.810e+02, 5.263e-02]])
```

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
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_state=1)
x_train.shape
```

```
(398, 5)
```

```
x_test.shape
```

```
(171, 5)
```

```
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
sc.fit(x_train)
x_train_new=sc.transform(x_train)
x_train_new
```

```
array([[0.21482323, 0.17653027, 0.207864  , 0.11147402, 0.43937889],
       [0.28723555, 0.32465336, 0.26826066, 0.16275716, 0.25250519],
       [0.34166312, 0.3659114 , 0.33598231, 0.20144221, 0.33113659],
       ...,
       [0.48364807, 0.50084545, 0.48655933, 0.33336161, 0.49173964],
       [0.3336173 , 0.3902604 , 0.31787713, 0.19507953, 0.34368511],
       [0.28628899, 0.29455529, 0.26826066, 0.16131495, 0.335831  ]])
```

```
x_test_new=sc.transform(x_test)
x_test_new
```

```
array([[0.36485399, 0.14440311, 0.37613157, 0.21743372, 0.45562878],
       [0.29291495, 0.30267163, 0.29154861, 0.16589608, 0.57028076],
       [0.28250272, 0.21339195, 0.27192316, 0.15703075, 0.43215672],
       [0.5361825 , 0.29996618, 0.51696496, 0.38069989, 0.30017153],
       [0.38567845, 0.67974298, 0.36569691, 0.24432662, 0.27597725],
       [0.43442662, 0.40006764, 0.43127635, 0.2826299 , 0.43486504],
       [0.57783142, 0.21068651, 0.57017483, 0.42990456, 0.3097409 ],
       [0.52529699, 0.41021305, 0.50867252, 0.37348887, 0.19030423],
       [0.16560178, 0.3432533 , 0.15845484, 0.0823754 , 0.49083687],
       [0.35728146, 0.14440311, 0.34600235, 0.212386  , 0.51701724],
       [0.19210564, 0.24078458, 0.18747841, 0.09743372, 0.49715627],
       [0.34118983, 0.47683463, 0.33916108, 0.19817603, 0.37916403],
       [0.43821288, 0.30639161, 0.44924331, 0.28063627, 0.48722578],
       [0.25457901, 0.29861346, 0.24338332, 0.13709438, 0.29015076],
       [0.36722041, 0.5312817 , 0.35180706, 0.22273595, 0.27191478],
       [0.27918974, 0.28779168, 0.28097574, 0.14829268, 0.62444705],
       [0.24700648, 0.18599932, 0.23647295, 0.13336161, 0.30784508],
       [0.22239576, 0.21846466, 0.21905881, 0.11749735, 0.54319762],
       [0.16986133, 0.3554278 , 0.18215742, 0.0826299 , 0.34395594],
       [0.63888494, 0.39736219, 0.61301914, 0.4931071 , 0.27913695],
       [0.27351034, 0.30875888, 0.26314698, 0.14977731, 0.39839307],
       [0.21718964, 0.31552249, 0.21014443, 0.11291622, 0.29637989],
       [0.59676274, 0.51707812, 0.57984935, 0.44432662, 0.45653155],
       [0.27303706, 0.23638823, 0.26756962, 0.14858961, 0.5404893 ],
       [0.36911354, 0.48123098, 0.37046507, 0.2226087 , 0.58291956],
```

```
[0.204411  , 0.28677714, 0.20827863, 0.10430541, 0.39080979],
[0.36012116, 0.43862022, 0.36348559, 0.2178579 , 0.28978965],
[0.54328175, 0.2979371 , 0.5342409 , 0.39512195, 0.41626794],
[0.9578778 , 0.4112276 , 0.95577362, 0.89353128, 0.51250339],
[0.47465569, 0.35678052, 0.45546265, 0.32271474, 0.31678252],
[0.68999953, 0.42881299, 0.67866768, 0.56648993, 0.52694773],
[0.1930522 , 0.17754481, 0.19141732, 0.09773065, 0.45743432],
[0.43300677, 0.37098411, 0.44440605, 0.27796394, 0.58111402],
[0.33125089, 0.33513696, 0.32706793, 0.19342524, 0.48180915],
[0.11756354, 0.38214406, 0.11277728, 0.05340403, 0.46736481],
[0.33740357, 0.10720325, 0.31953562, 0.20063627, 0.36526135],
[0.51819774, 0.49949273, 0.4934697 , 0.36284199, 0.32427553],
[0.36154101, 0.48393642, 0.35090871, 0.2202333 , 0.33501851],
[0.18784609, 0.3936422 , 0.19425057, 0.09654295, 0.632572  ],
[0.22381561, 0.25295908, 0.21346141, 0.11741251, 0.40724023],
[0.10951772, 0.14169767, 0.11174072, 0.04848356, 0.76257109],
[0.30806001, 0.42576936, 0.29797526, 0.17709438, 0.31497698],
[0.22570874, 0.35407508, 0.22327413, 0.11720042, 0.50798953],
[0.30238061, 0.22590463, 0.29237786, 0.17391304, 0.16719328],
[0.31137299, 0.14169767, 0.30958469, 0.17722163, 0.47368421],
[0.39467083, 0.25566452, 0.41054523, 0.24169671, 0.73007132],
[0.0336031 , 0.53195807, 0.03144219, 0.01141039, 0.3073937 ],
[0.13067348, 0.20155563, 0.1223827 , 0.06209968, 0.34070597],
[0.36248758, 0.24146094, 0.34842098, 0.22163309, 0.30495622],
[0.50115954, 0.18058843, 0.49208762, 0.34426299, 0.41383046],
[0.521984  , 0.36692594, 0.51558289, 0.36627784, 0.42854564],
[0.52103744, 0.0226581 , 0.54598853, 0.36373277, 0.59375282],
[0.26735766, 0.37368955, 0.26508189, 0.14290562, 0.42159429],
[0.26499124, 0.29387893, 0.24904982, 0.14655355, 0.28256748],
[0.13067348, 0.31822793, 0.12535416, 0.06201485, 0.49535073],
[0.32746462, 0.23368279, 0.31221063, 0.19338282, 0.14128374],
[0.28060959, 0.22387555, 0.26770783, 0.15817603, 0.24176221],
[0.55984666, 0.34731146, 0.53285882, 0.40657476, 0.33041437],
```

```
#knn
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train_new,y_train)
y_pred=knn.predict(x_test_new)
y_pred
```

```
array([1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1])
```

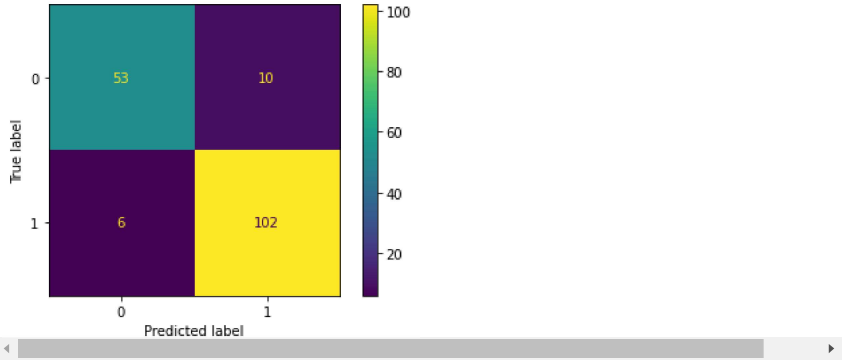
```
from sklearn.metrics import classification_report,ConfusionMatrixDisplay
print(classification_report(y_test,y_pred))
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.90      0.84      0.87        63
               1       0.91      0.94      0.93       108

        accuracy                           0.91       171
       macro avg       0.90      0.89      0.90       171
    weighted avg       0.91      0.91      0.91       171

    <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fa819d
```




```
#Naive bayes
from sklearn.naive_bayes import GaussianNB
```

```
nb=GaussianNB()
nb.fit(x_train_new,y_train)
y_pred_nb=nb.predict(x_test_new)
y_pred_nb
```

```
array([1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1])
```

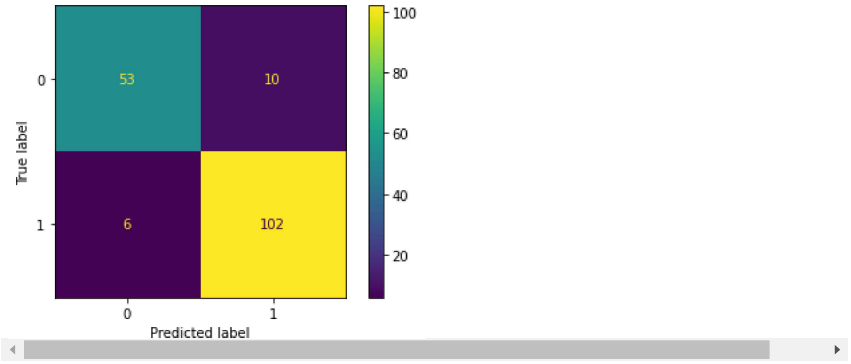
```
from sklearn.metrics import classification_report,ConfusionMatrixDisplay
print(classification_report(y_test,y_pred))
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.84      0.87        63
           1       0.91      0.94      0.93       108

    accuracy                           0.91       171
   macro avg       0.90      0.89      0.90       171
weighted avg       0.91      0.91      0.91       171

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fa81a5
```




```
#SVM
from sklearn.svm import SVC
svm=SVC()
svm.fit(x_train_new,y_train)
y_pred_svm=svm.predict(x_test_new)
y_pred_svm
```

```
array([1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1])
```

```
svm.predict(sc.transform([[18.5,19.8,130.1,516,0.5]])) # transforming values into 0 to 1 scale like training data
```

```
array([0])
```

```
print(classification_report(y_test,y_pred_svm))
```

```
              precision    recall  f1-score   support

           0       0.90      0.86      0.88        63
           1       0.92      0.94      0.93       108

    accuracy                           0.91       171
   macro avg       0.91      0.90      0.90       171
weighted avg       0.91      0.91      0.91       171
```

```
#Decision tree
from sklearn.tree import DecisionTreeClassifier
```

```
dt=DecisionTreeClassifier()
dt.fit(x_train_new,y_train)
y_pred_dt=dt.predict(x_test_new)
y_pred_dt
```

```
array([1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1])
```

```
print(classification_report(y_test,y_pred_dt))
```

```
              precision    recall  f1-score   support

           0       0.79      0.79      0.79        63
           1       0.88      0.88      0.88       108

    accuracy                           0.85       171
   macro avg       0.84      0.84      0.84       171
weighted avg       0.85      0.85      0.85       171
```

```
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred_dt))
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7fa8197
```