```
import pandas as pd
df=pd.read_csv("/content/cancer_data (1).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.isna().sum()
```

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

```
df.describe()
```

	mean_radius	mean_texture	mean_perimeter	mean_area	mean_smooth
count	569.000000	569.000000	569.000000	569.000000	569.00
mean	14.127292	19.289649	91.969033	654.889104	0.09
std	3.524049	4.301036	24.298981	351.914129	0.01
min	6.981000	9.710000	43.790000	143.500000	0.05
25%	11.700000	16.170000	75.170000	420.300000	0.08
50%	13.370000	18.840000	86.240000	551.100000	0.09
75%	15.780000	21.800000	104.100000	782.700000	0.10

```
df.dtypes
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   mean_radius      569 non-null    float64
 1   mean_texture     569 non-null    float64
 2   mean_perimeter   569 non-null    float64
 3   mean_area        569 non-null    float64
 4   mean_smoothness  569 non-null    float64
 5   diagnosis        569 non-null    int64
```

```
dtypes: float64(5), int64(1)
memory usage: 26.8 KB
```

```
X=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=df.iloc[:,-1].values
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, 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, 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, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 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, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 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, 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, 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])
```

```
#split
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=8)
```

```
X_train.shape
```

```
(398, 5)
```

```
X_test.shape
```

```
(171, 5)
```

```
#scaling
from sklearn.preprocessing import MinMaxScaler
minmax=MinMaxScaler()
minmax.fit(X_train)
X_train_new=minmax.transform(X_train)
X_train_new
```

```
array([[0.15518955, 0.28506224, 0.15244282, 0.07520679, 0.41459218],
       [0.28392257, 0.4       , 0.28145947, 0.15715801, 0.49512447],
       [0.36485399, 0.17717842, 0.37613157, 0.21743372, 0.5789836 ],
       ...,
       [0.29906763, 0.49211618, 0.28643494, 0.17077413, 0.38075026],
       [0.35207535, 0.28340249, 0.34800636, 0.21111347, 0.51485603],
       [0.59676274, 0.63443983, 0.57984935, 0.44432662, 0.58013078]])
```

```
X_test_new=minmax.transform(X_test)
X_test_new
```

```
array([[0.26925079, 0.31493776, 0.25886255, 0.14693531, 0.57554204],
       [0.11557575, 0.75394191, 0.10690346, 0.05420997, 0.32809453],
       [0.27682332, 0.24979253, 0.27019556, 0.15414634, 0.43329127],
```

```
[0.3085333 , 0.12448133, 0.28954461, 0.17930011, 0.24239991],
[0.1113162 , 0.50746888, 0.10517587, 0.05111347, 0.45623494],
[0.18547967, 0.13485477, 0.17206827, 0.09471898, 0.25834576],
[0.35160206, 0.41576763, 0.36099786, 0.21527041, 0.40082597],
[0.22807516, 0.28506224, 0.24324511, 0.1223754 , 0.64781462],
[0.25647215, 0.33070539, 0.26038283, 0.13756098, 0.60536882],
[0.25363245, 0.10622407, 0.24289959, 0.1378579 , 0.42904669],
[0.32178522, 0.25228216, 0.3080644 , 0.18765642, 0.40770907],
[0.28723555, 0.17136929, 0.2689517 , 0.16419936, 0.35436503],
[0.29812107, 0.16639004, 0.29023564, 0.16895016, 0.35562694],
[0.43963273, 0.45643154, 0.43611361, 0.2842842 , 0.73729494],
[0.57877798, 0.36763485, 0.56464653, 0.42778367, 0.60422164],
[0.29291495, 0.35311203, 0.27980098, 0.16704136, 0.25272456],
[0.22523546, 0.17883817, 0.21042084, 0.12038176, 0.29253183],
[0.66065597, 0.5746888 , 0.65724553, 0.51770944, 0.55145119],
[0.31373941, 0.62157676, 0.30384908, 0.18201485, 0.45829987],
[0.51725117, 0.46929461, 0.55704512, 0.36076352, 0.80727314],
[0.18784609, 0.48298755, 0.19425057, 0.09654295, 0.80383159],
[0.51819774, 0.61286307, 0.4934697 , 0.36284199, 0.41206837],
[0.35964788, 0.16390041, 0.34897381, 0.218579  , 0.52414822],
[0.22523546, 0.206639  , 0.22472531, 0.11983033, 0.32534129],
[0.21245681, 0.26224066, 0.20199019, 0.10994698, 0.43742113],
[0.25883856, 0.24854772, 0.26798424, 0.14150583, 0.86233796],
[0.59013678, 0.39917012, 0.57155691, 0.4349947 , 0.58357233],
[0.41076246, 0.55975104, 0.45891784, 0.26723224, 0.66846392],
[0.25457901, 0.36639004, 0.24338332, 0.13709438, 0.36870483],
[0.23706754, 0.62987552, 0.2337088 , 0.12632025, 0.57783641],
[0.20772398, 0.38091286, 0.19611637, 0.10710498, 0.39577836],
[0.32604477, 0.4560166 , 0.31773893, 0.18718982, 0.49500975],
[0.26972408, 0.58630705, 0.26881349, 0.15079533, 0.68337731],
[0.47796867, 0.72365145, 0.46582821, 0.33399788, 0.53171963],
[0.55038099, 0.4373444 , 0.54115127, 0.40318134, 0.47917862],
[0.21151025, 0.46721992, 0.20744938, 0.10943796, 0.66043364],
[0.8173127 , 0.43526971, 0.84589869, 0.68610817, 1.056212  ],
[0.4012968 , 0.40539419, 0.40017967, 0.25679745, 0.6489618 ],
[0.35728146, 0.39958506, 0.34869739, 0.21896076, 0.3584949 ],
[0.59392304, 0.94439834, 0.58192247, 0.45790032, 0.36216588],
[0.1452506 , 0.32448133, 0.14249188, 0.07096501, 0.55145119],
[0.57357187, 0.6879668 , 0.58952388, 0.41930011, 0.79006539],
[0.28486914, 0.50248963, 0.30205238, 0.15961824, 0.85660204],
[0.25173932, 0.38630705, 0.2355746 , 0.13611877, 0.34989102],
[0.38567845, 0.8340249 , 0.36569691, 0.24432662, 0.35069405],
[0.24747977, 0.34481328, 0.23854606, 0.1335737 , 0.45520248],
[0.32036537, 0.58340249, 0.30923917, 0.18939555, 0.31891706],
[0.48080837, 0.27759336, 0.498998  , 0.32627784, 0.75679706],
[0.63699181, 0.50082988, 0.62200263, 0.48759279, 0.44591029],
[0.55132756, 0.63900415, 0.55980927, 0.40063627, 0.61684066],
[0.3814189 , 0.29170124, 0.37965586, 0.23155885, 0.52999885],
[0.41265559, 0.43983402, 0.39672448, 0.26430541, 0.4971894 ],
[0.38851815, 0.14522822, 0.37219266, 0.24106045, 0.30973959],
[0.32131194, 0.5219917 , 0.32368185, 0.18892895, 0.77285763],
[0.37053339, 0.20705394, 0.35153065, 0.2278685 , 0.36813124],
[0.47512897, 0.59211618, 0.4768848 , 0.32059385, 0.76941608],
[0.443419  , 0.56348548, 0.45062539, 0.29560976, 0.51382356],
[0.16986133, 0.35726141, 0.15914588, 0.08373277, 0.33027418],
```

```
from sklearn.neighbors import KNeighborsClassifier# k- nearest values,skykit
knn=KNeighborsClassifier(n_neighbors=5)# by default -5
knn.fit(X_train_new,y_train)
y_pred=knn.predict(X_test_new)
y_pred
```

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

```
from sklearn.metrics import classification_report,accuracy_score,ConfusionMatrixDisplay
print(accuracy_score(y_test,y_pred)*100)
print(classification_report(y_test,y_pred))
```

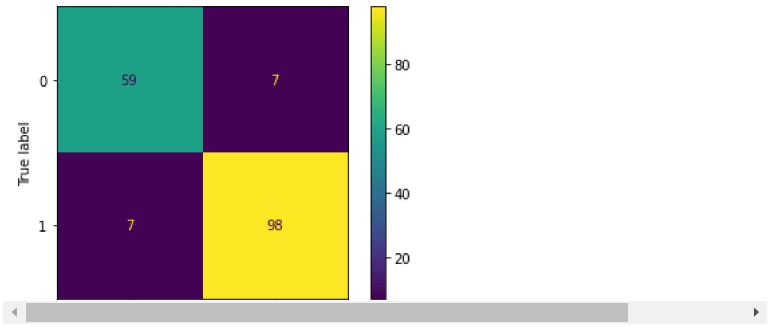
```
91.81286549707602
              precision    recall  f1-score   support

           0       0.89      0.89      0.89        66
           1       0.93      0.93      0.93       105

    accuracy                           0.92       171
   macro avg       0.91      0.91      0.91       171
weighted avg       0.92      0.92      0.92       171
```

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

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




```
#Nawebayeclassifier
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, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0])
```

```
print(accuracy_score(y_test,y_pred_nb)*100)
```

```
92.98245614035088
```

```
print(classification_report(y_test,y_pred_nb))
```

```
              precision    recall  f1-score   support

           0       0.98      0.83      0.90        66
           1       0.90      0.99      0.95       105

    accuracy                           0.93       171
   macro avg       0.94      0.91      0.92       171
weighted avg       0.93      0.93      0.93       171
```

```
#svm
from sklearn.svm import SVC
sv=SVC()#(kernel="poly")# default rbf-radial basis function
sv.fit(X_train_new,y_train)
y_pred_sv=sv.predict(X_test_new)
y_pred_sv
```

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

```
#to predict a new patient if he has cancer or not .we have already scaled it
sv.predict(minmax.transform([[18.5,19.8,130.1,516,0.05]]))
```

```
array([1])
```

```
print(accuracy_score(y_test,y_pred_sv))
```

```
0.9415204678362573
```

```
print(classification_report(y_test,y_pred_sv))
```

```
                precision    recall  f1-score   support

             0       0.95      0.89      0.92        66
             1       0.94      0.97      0.95       105

      accuracy                           0.94       171
     macro avg       0.94      0.93      0.94       171
  weighted avg       0.94      0.94      0.94       171
```

```
#decision tree classifier
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier(criterion="entropy")
dt.fit(X_train_new,y_train)
y_pred_decisiontree=dt.predict(X_test_new)
```

```
print(accuracy_score(y_test,y_pred_decisiontree))
print(classification_report(y_test,y_pred_decisiontree))
```

```
0.8771929824561403
              precision    recall  f1-score   support

           0       0.85      0.83      0.84        66
           1       0.90      0.90      0.90       105

    accuracy                           0.88       171
   macro avg       0.87      0.87      0.87       171
weighted avg       0.88      0.88      0.88       171
```

```
from sklearn import tree as tr
import matplotlib.pyplot as plt
plt.figure(figsize=(40,40))
tr.plot_tree(dt,feature_names=["mean_radius","mean_texture","mean_perimeter","mean_area","mean_smooth
```

```
[Text(0.49166666666666664, 0.9545454545454546, 'mean_perimeter <=
0.32\nentropy = 0.948\nsamples = 398\nvalue = [146, 252]'),
 Text(0.16666666666666666, 0.8636363636363636, 'mean_texture <=
0.349\nentropy = 0.329\nsamples = 232\nvalue = [14, 218]'),
 Text(0.06666666666666667, 0.7727272727272727, 'mean_smoothness <=
0.985\nentropy = 0.066\nsamples = 127\nvalue = [1, 126]'),
 Text(0.03333333333333333, 0.6818181818181818, 'entropy = 0.0\nsamples
= 126\nvalue = [0, 126]'),
 Text(0.1, 0.6818181818181818, 'entropy = 0.0\nsamples = 1\nvalue = [1,
0]'),
 Text(0.26666666666666666, 0.7727272727272727, 'mean_smoothness <=
0.54\nentropy = 0.54\nsamples = 105\nvalue = [13, 92]'),
 Text(0.16666666666666666, 0.6818181818181818, 'mean_radius <=
0.274\nentropy = 0.25\nsamples = 72\nvalue = [3, 69]'),
 Text(0.13333333333333333, 0.5909090909090909, 'entropy = 0.0\nsamples
= 47\nvalue = [0, 47]'),
 Text(0.2, 0.5909090909090909, 'mean_area <= 0.154\nentropy =
0.529\nsamples = 25\nvalue = [3, 22]'),
 Text(0.16666666666666666, 0.5, 'entropy = 0.0\nsamples = 1\nvalue =
[1, 0]'),
 Text(0.23333333333333334, 0.5, 'mean_area <= 0.178\nentropy =
0.414\nsamples = 24\nvalue = [2, 22]'),
 Text(0.2, 0.4090909090909091, 'entropy = 0.0\nsamples = 17\nvalue =
[0, 17]'),
 Text(0.26666666666666666, 0.4090909090909091, 'mean_area <=
0.181\nentropy = 0.863\nsamples = 7\nvalue = [2, 5]'),
 Text(0.23333333333333334, 0.3181818181818182, 'entropy = 0.0\nsamples
= 2\nvalue = [2, 0]'),
 Text(0.3, 0.3181818181818182, 'entropy = 0.0\nsamples = 5\nvalue = [0,
5]'),
 Text(0.36666666666666664, 0.6818181818181818, 'mean_perimeter <=
0.202\nentropy = 0.885\nsamples = 33\nvalue = [10, 23]'),
 Text(0.3333333333333333, 0.5909090909090909, 'entropy = 0.0\nsamples =
16\nvalue = [0, 16]'),
 Text(0.4, 0.5909090909090909, 'mean_texture <= 0.46\nentropy =
0.977\nsamples = 17\nvalue = [10, 7]')]
```