```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df=pd.read_csv('/content/data.csv')
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoot
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	
...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

569 rows × 33 columns

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

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
Unnamed: 32                569
dtype: int64
```

```
df=df.drop(columns='id')
```

```
df.head()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	M	17.99	10.38	122.80	1001.0	0.11840
1	M	20.57	17.77	132.90	1326.0	0.08474
2	M	19.69	21.25	130.00	1203.0	0.10960
3	M	11.42	20.38	77.58	386.1	0.14250
4	M	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

```
df.dtypes
```

```
diagnosis                 object
radius_mean              float64
texture_mean             float64
perimeter_mean           float64
area_mean                float64
smoothness_mean          float64
compactness_mean         float64
concavity_mean           float64
concave points_mean      float64
symmetry_mean            float64
fractal_dimension_mean   float64
radius_se                float64
texture_se               float64
perimeter_se             float64
area_se                  float64
smoothness_se            float64
compactness_se           float64
concavity_se             float64
concave points_se        float64
symmetry_se              float64
fractal_dimension_se     float64
radius_worst             float64
texture_worst            float64
perimeter_worst          float64
area_worst               float64
smoothness_worst         float64
compactness_worst        float64
concavity_worst          float64
concave points_worst     float64
symmetry_worst           float64
fractal_dimension_worst  float64
Unnamed: 32              float64
dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['diagnosis']=le.fit_transform(df['diagnosis'])         # 'B'&'M' is converted to 0 & 1
```

```
df.head()
```

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
0	1	17.99	10.38	122.80	1001.0	0.11840
1	1	20.57	17.77	132.90	1326.0	0.08474
2	1	19.69	21.25	130.00	1203.0	0.10960
3	1	11.42	20.38	77.58	386.1	0.14250
4	1	20.29	14.34	135.10	1297.0	0.10030

5 rows × 32 columns

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

```
diagnosis                 0
radius_mean               0
texture_mean              0
perimeter_mean            0
area_mean                 0
smoothness_mean           0
compactness_mean          0
concavity_mean            0
concave points_mean       0
```

```
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

```
df=df.drop(columns='Unnamed: 32')
```

```
X=df.iloc[:,2:].values
y=df.iloc[:,0].values
```

Scaling

```
from sklearn.preprocessing import MinMaxScaler
minmax=MinMaxScaler()
X=minmax.fit_transform(X)
X
```

```
array([[0.0226581 , 0.54598853, 0.36373277, ..., 0.91202749, 0.59846245,
        0.41886396],
       [0.27257355, 0.61578329, 0.50159067, ..., 0.63917526, 0.23358959,
        0.22287813],
       [0.3902604 , 0.59574321, 0.44941676, ..., 0.83505155, 0.40370589,
        0.21343303],
       ...,
       [0.62123774, 0.44578813, 0.30311771, ..., 0.48728522, 0.12872068,
        0.1519087 ],
       [0.66351031, 0.66553797, 0.4757158 , ..., 0.91065292, 0.49714173,
        0.45231536],
       [0.50152181, 0.02853984, 0.01590668, ..., 0.        , 0.25744136,
        0.10068215]])
```

```
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)
```

KNeighbors Classifier

```
from sklearn.neighbors import KNeighborsClassifier
clf=KNeighborsClassifier(n_neighbors=5)
clf.fit(X_train,y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
y_pred=clf.predict(X_test)
y_pred
```

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

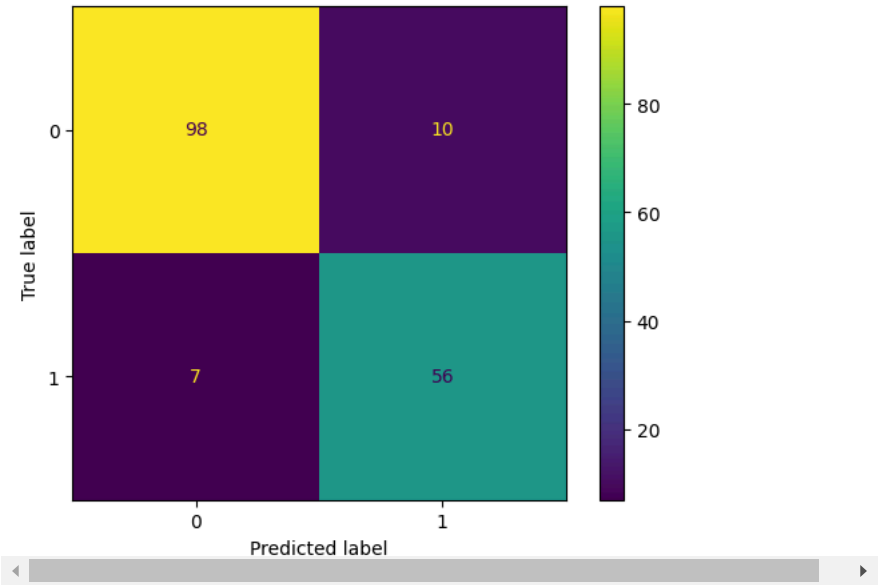
```
from sklearn.metrics import accuracy_score,ConfusionMatrixDisplay
```

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

```
90.05847953216374
```

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

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x79bd5d3a00
```



Linear Regression

```
from sklearn.linear_model import LinearRegression
linear_model = LinearRegression()
linear_model.fit(X_train,y_train)
```

```
LinearRegression
LinearRegression()
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
lr.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
y_pred = lr.predict(X_test)
```

DecisionTree Classifier

```
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion='entropy')
dtg=DecisionTreeClassifier(criterion='gini')
dtc.fit(X_train,y_train)
dtg.fit(X_train,y_train)
y_predc=dtc.predict(X_test)
y_predg=dtg.predict(X_test)
```

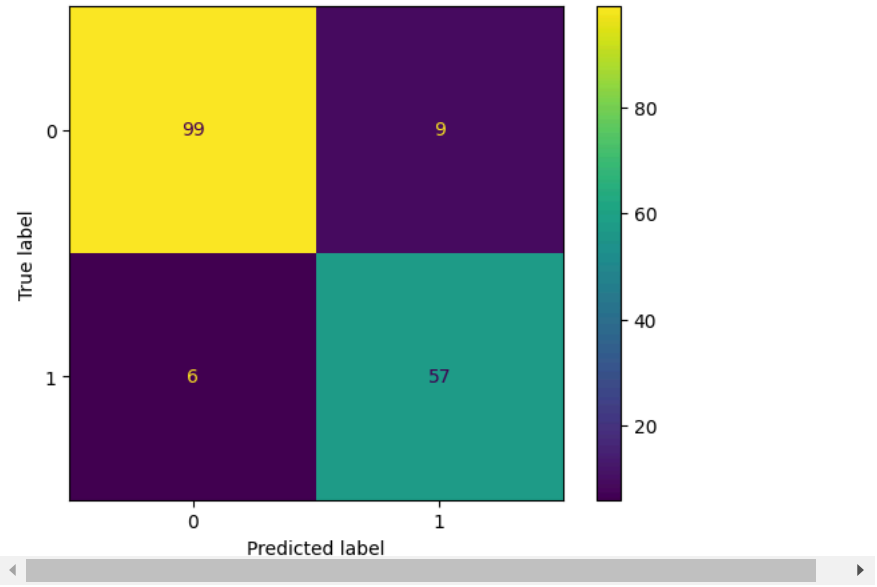
```
from sklearn.metrics import classification_report,ConfusionMatrixDisplay
print(classification_report(y_test,y_predc))
print(ConfusionMatrixDisplay.from_predictions(y_test,y_predc))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       108
           1       0.86      0.90      0.88        63

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

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x79bd5d2a20
```



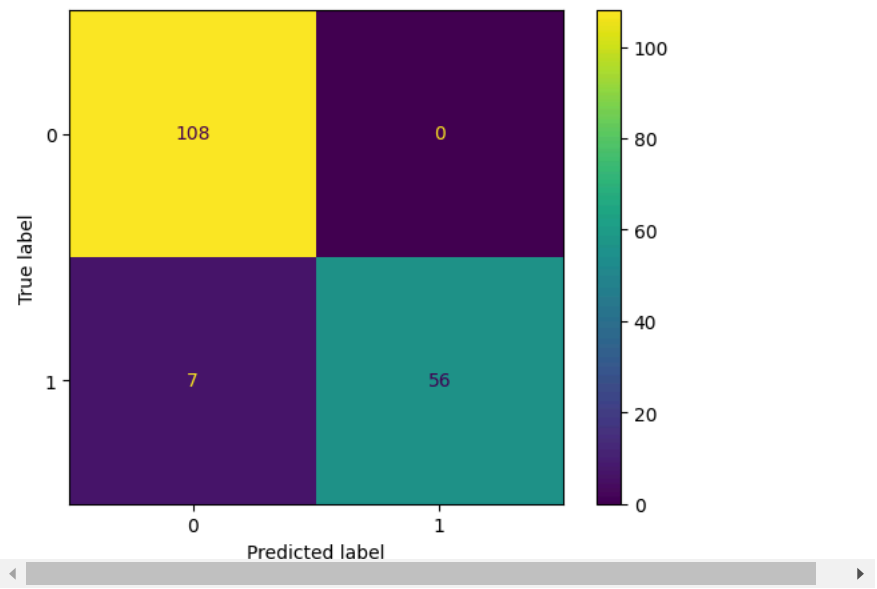

```
from sklearn.metrics import classification_report,ConfusionMatrixDisplay,accuracy_score
print('acc',accuracy_score(y_test,y_pred))
print(classification_report(y_test,y_predg))
print(ConfusionMatrixDisplay.from_predictions(y_test,y_predg))
```

```
acc 0.9649122807017544
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       108
           1       1.00      0.89      0.94        63

    accuracy                           0.96       171
   macro avg       0.97      0.94      0.95       171
weighted avg       0.96      0.96      0.96       171
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x79bd5d60f4
```



RandomForest Classifier

```
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier(n_estimators=15,criterion='entropy')
rfc.fit(X_train,y_train)
y_pred=rfc.predict(X_test)
print(y_pred)
```

```
[0 1 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 1 0 1 1 0 0 1
 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1
 0 1 0 0 0 1 0 1 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1
 0 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 0 1 0 0
 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 1 0 0]
```

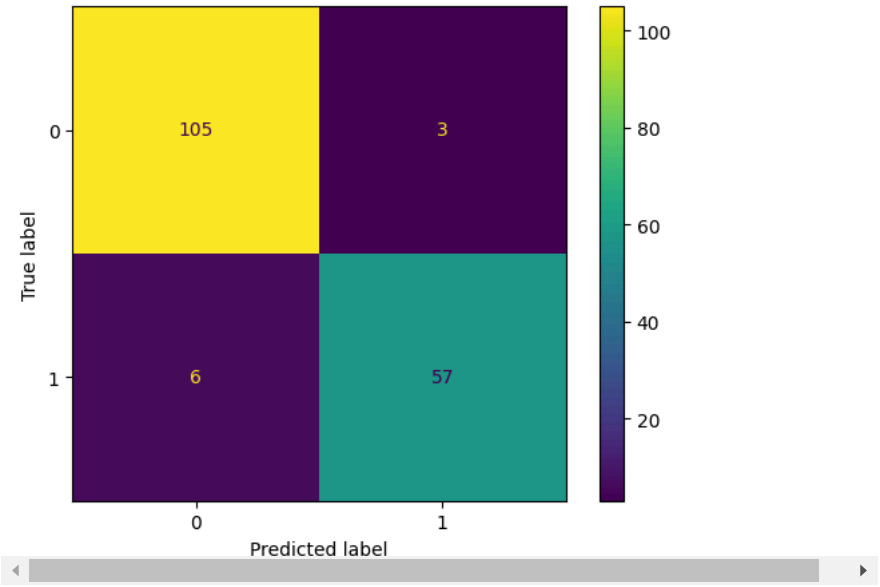
```
print(classification_report(y_test,y_pred))
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.97      0.96       108
           1       0.95      0.90      0.93        63

    accuracy                           0.95       171
   macro avg       0.95      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x79bd5f3564
```




```
# With tuning
from sklearn.model_selection import GridSearchCV
params={'criterion':['entropy','gini','log_loss'],'splitter':['best','random']}     # criterion and splitter is the par
clf=GridSearchCV(dtc,params,cv=10,scoring='accuracy')
clf.fit(X_train,y_train)
```

```
GridSearchCV
estimator: DecisionTreeClassifier
DecisionTreeClassifier
```

```
print(clf.best_params_)
```

```
{'criterion': 'entropy', 'splitter': 'best'}
```

```
print(clf.best_score_)
```

```
0.9523717948717948
```

```
dtc=DecisionTreeClassifier(criterion='entropy',splitter='best',random_state=1)
dtc.fit(X_train,y_train)
y_pred=dtc.predict(X_test)
y_pred
```

```
array([0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
```

```
       0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0])
```

```
print('acc',accuracy_score(y_test,y_pred))
print('classification report',classification_report(y_test,y_pred))
print(ConfusionMatrixDisplay.from_predictions(y_test,y_pred))
```

```
acc 0.9064327485380117
classification report               precision    recall  f1-score   support

           0       0.93      0.92      0.93       108
           1       0.86      0.89      0.88        63

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

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x79bd5d0fd3
```