```
#Task-4 Classification with Logistic regression

#Objective: Build a binary classifier using logistic regression.
#Tools: Scikit-learn, Pandas, Matplotlib
import numpy as np
import pandas as pd
df=pd.read_csv('/content/data.csv')
df
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280
...	...	...	...	...	...	...	...	...
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362

569 rows × 33 columns

```
#Splitting the dataset into Training and Testing data
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Load data
X, y = load_breast_cancer(return_X_y=True)

# Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train
X_test
```

```
array([[-0.46649743, -0.13728933, -0.44421138, ..., -0.19435087,
         0.17275669,  0.20372995],
       [ 1.36536344,  0.49866473,  1.30551088, ...,  0.99177862,
        -0.561211  , -1.00838949],
       [ 0.38006578,  0.06921974,  0.40410139, ...,  0.57035018,
        -0.10783139, -0.20629287],
       ...,
       [-0.73547237, -0.99852603, -0.74138839, ..., -0.27741059,
        -0.3820785 , -0.32408328],
       [ 0.02898271,  2.0334026 ,  0.0274851 , ..., -0.49027026,
        -1.60905688, -0.33137507],
       [ 1.87216885,  2.80077153,  1.80354992, ...,  0.7925579 ,
        -0.05868885, -0.09467243]])
```

```
#Fit the model
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

```
#Evaluate ConfusionMAtrix,precision,recall,ROC-Auc
from sklearn.metrics import confusion_matrix, precision_score, recall_score, roc_auc_score

# Predictions
y_pred = model.predict(X_test)
y_proba = model.predict_proba(X_test)[:, 1]

# Metrics
cm = confusion_matrix(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_proba)

print("Confusion Matrix:\n", cm)
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print(f"ROC-AUC: {roc_auc:.2f}")
```

```
Confusion Matrix:
 [[41  2]
 [ 1 70]]
Precision: 0.97
Recall: 0.99
ROC-AUC: 1.00
```

```
#.Tune threshold and explain sigmoid function.

import numpy as np

# Custom threshold
custom_threshold = 0.3
y_pred_custom = (y_proba >= custom_threshold).astype(int)

# Re-evaluate
precision_custom = precision_score(y_test, y_pred_custom)
recall_custom = recall_score(y_test, y_pred_custom)

print(f"Precision (threshold=0.3): {precision_custom:.2f}")
print(f"Recall (threshold=0.3): {recall_custom:.2f}")
```

```
Precision (threshold=0.3): 0.97
Recall (threshold=0.3): 1.00
```