```
from google.colab import files
_ = files.upload()
```

Choose Files 2 files
wdbc.data(n/a) - 124103 bytes, last modified: 15/1/2026 - 100% done
wdbc.names(n/a) - 4708 bytes, last modified: 15/1/2026 - 100% done

```
Saving wdbc.data to wdbc.data
Saving wdbc.names to wdbc.names
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
```

```
# Column names
columns = ["id", "diagnosis",
           "radius_mean","texture_mean","perimeter_mean","area_mean","smoothness_mean",
           "compactness_mean","concavity_mean","concave_points_mean","symmetry_mean","fractal_dimension_mean",
           "radius_se","texture_se","perimeter_se","area_se","smoothness_se",
           "compactness_se","concavity_se","concave_points_se","symmetry_se","fractal_dimension_se",
           "radius_worst","texture_worst","perimeter_worst","area_worst","smoothness_worst",
           "compactness_worst","concavity_worst","concave_points_worst","symmetry_worst","fractal_dimension_worst"]

# Load dataset
data = pd.read_csv("wdbc.data", header=None, names=columns)
data.head()
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave_points_mean	...	radius_worst	texture
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	25.38	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	24.99	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	23.57	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	14.91	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	22.54	

5 rows × 32 columns

```
le = LabelEncoder()
data['diagnosis'] = le.fit_transform(data['diagnosis'])  # M=1, B=0
```

```
X = data.drop(["id", "diagnosis"], axis=1)
y = data["diagnosis"]
```

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

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

```
lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)
```

```
▾    LogisticRegression    i ?
LogisticRegression(max_iter=1000)
```

```
dt = DecisionTreeClassifier(random_state=42)
dt.fit(X_train, y_train)
```

```
▾    DecisionTreeClassifier    i ?
DecisionTreeClassifier(random_state=42)
```

```
def evaluate(model, X_train, X_test, y_train, y_test):
    train_pred = model.predict(X_train)
    test_pred = model.predict(X_test)

    print("Train Error:", 1 - accuracy_score(y_train, train_pred))
    print("Test Error:", 1 - accuracy_score(y_test, test_pred))
    print("Accuracy:", accuracy_score(y_test, test_pred))
    print("Precision:", precision_score(y_test, test_pred))
    print("Recall:", recall_score(y_test, test_pred))
    print("F1-score:", f1_score(y_test, test_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, test_pred))
    print("="*50)
```

```
print("Logistic Regression Evaluation:")
evaluate(lr, X_train, X_test, y_train, y_test)

print("Decision Tree Evaluation:")
evaluate(dt, X_train, X_test, y_train, y_test)
```

```
Logistic Regression Evaluation:
Train Error: 0.013186813186813187
Test Error: 0.02631578947368418
Accuracy: 0.9736842105263158
Precision: 0.9761904761904762
Recall: 0.9534883720930233
F1-score: 0.9647058823529412
Confusion Matrix:
 [[70  1]
 [ 2 41]]
==================================================
Decision Tree Evaluation:
Train Error: 0.0
Test Error: 0.052631578947368474
Accuracy: 0.9473684210526315
Precision: 0.9302325581395349
Recall: 0.9302325581395349
F1-score: 0.9302325581395349
Confusion Matrix:
 [[68  3]
 [ 3 40]]
==================================================
```

```
print('submission successfull')
```

```
submission successfull
```