```
pip install ucimlrepo
```

```
Collecting ucimlrepo
  Downloading ucimlrepo-0.0.7-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2.2.2)
Requirement already satisfied: certifi>=2020.12.5 in /usr/local/lib/python3.12/dist-packages (from ucimlrepo) (2026.1.4)
Requirement already satisfied: numpy>=1.26.0 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo) (2.0
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlr
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo) (2025
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas>=1.0.0->ucimlrepo) (20
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0
Downloading ucimlrepo-0.0.7-py3-none-any.whl (8.0 kB)
Installing collected packages: ucimlrepo
Successfully installed ucimlrepo-0.0.7
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

# data (as pandas dataframes)
X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets

# metadata
print(breast_cancer_wisconsin_diagnostic.metadata)

# variable information
print(breast_cancer_wisconsin_diagnostic.variables)
```

```
{'uci_id': 17, 'name': 'Breast Cancer Wisconsin (Diagnostic)', 'repository_url': 'https://archive.ics.uci.edu/dataset/17/b
                  name     role         type demographic description units  \
0                   ID       ID  Categorical        None        None  None
1            Diagnosis   Target  Categorical        None        None  None
2              radius1  Feature   Continuous        None        None  None
3             texture1  Feature   Continuous        None        None  None
4           perimeter1  Feature   Continuous        None        None  None
5                area1  Feature   Continuous        None        None  None
6          smoothness1  Feature   Continuous        None        None  None
7         compactness1  Feature   Continuous        None        None  None
8           concavity1  Feature   Continuous        None        None  None
9      concave_points1  Feature   Continuous        None        None  None
10           symmetry1  Feature   Continuous        None        None  None
11  fractal_dimension1  Feature   Continuous        None        None  None
12             radius2  Feature   Continuous        None        None  None
13            texture2  Feature   Continuous        None        None  None
14          perimeter2  Feature   Continuous        None        None  None
15               area2  Feature   Continuous        None        None  None
16         smoothness2  Feature   Continuous        None        None  None
17        compactness2  Feature   Continuous        None        None  None
18          concavity2  Feature   Continuous        None        None  None
19     concave_points2  Feature   Continuous        None        None  None
20           symmetry2  Feature   Continuous        None        None  None
21  fractal_dimension2  Feature   Continuous        None        None  None
22             radius3  Feature   Continuous        None        None  None
23            texture3  Feature   Continuous        None        None  None
24          perimeter3  Feature   Continuous        None        None  None
25               area3  Feature   Continuous        None        None  None
26         smoothness3  Feature   Continuous        None        None  None
27        compactness3  Feature   Continuous        None        None  None
28          concavity3  Feature   Continuous        None        None  None
29     concave_points3  Feature   Continuous        None        None  None
30           symmetry3  Feature   Continuous        None        None  None
31  fractal_dimension3  Feature   Continuous        None        None  None

   missing_values
0              no
1              no
2              no
3              no
4              no
5              no
6              no
7              no
8              no
9              no
10             no
11             no
12             no
13             no
14             no
15             no
16             no
17             no
```

```
18             no
19             no
20             no
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
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
)
```

```
# Convert y DataFrame to Series and encode
y = y.iloc[:, 0].map({'M': 1, 'B': 0})

print(y.value_counts())
```

```
Diagnosis
0    357
1    212
Name: count, dtype: int64
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Training samples:", X_train.shape[0])
print("Test samples:", X_test.shape[0])
```

```
Training samples: 455
Test samples: 114
```

```
scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
log_model = LogisticRegression(max_iter=500)
log_model.fit(X_train_scaled, y_train)

# Predictions
y_train_pred_log = log_model.predict(X_train_scaled)
y_test_pred_log = log_model.predict(X_test_scaled)

# Errors
train_error_log = 1 - accuracy_score(y_train, y_train_pred_log)
test_error_log = 1 - accuracy_score(y_test, y_test_pred_log)

print("LOGISTIC REGRESSION RESULTS")
print("Train Error:", round(train_error_log, 4))
print("Test Error:", round(test_error_log, 4))
print("Accuracy:", round(accuracy_score(y_test, y_test_pred_log), 4))
print("Precision:", round(precision_score(y_test, y_test_pred_log), 4))
print("Recall:", round(recall_score(y_test, y_test_pred_log), 4))
print("F1-score:", round(f1_score(y_test, y_test_pred_log), 4))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred_log))
```

```
LOGISTIC REGRESSION RESULTS
Train Error: 0.0132
Test Error: 0.0351
Accuracy: 0.9649
Precision: 0.975
Recall: 0.9286
F1-score: 0.9512
Confusion Matrix:
 [[71  1]
 [ 3 39]]
```

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

# Predictions
y_train_pred_tree = tree_model.predict(X_train)
y_test_pred_tree = tree_model.predict(X_test)

# Errors
train_error_tree = 1 - accuracy_score(y_train, y_train_pred_tree)
test_error_tree = 1 - accuracy_score(y_test, y_test_pred_tree)

print("DECISION TREE RESULTS")
print("Train Error:", round(train_error_tree, 4))
print("Test Error:", round(test_error_tree, 4))
print("Accuracy:", round(accuracy_score(y_test, y_test_pred_tree), 4))
print("Precision:", round(precision_score(y_test, y_test_pred_tree), 4))
print("Recall:", round(recall_score(y_test, y_test_pred_tree), 4))
print("F1-score:", round(f1_score(y_test, y_test_pred_tree), 4))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_test_pred_tree))
```

```
DECISION TREE RESULTS
Train Error: 0.0
Test Error: 0.0702
Accuracy: 0.9298
Precision: 0.9048
Recall: 0.9048
F1-score: 0.9048
Confusion Matrix:
 [[68  4]
 [ 4 38]]
```

```
comparison = pd.DataFrame({
    "Model": ["Logistic Regression", "Decision Tree"],
    "Train Error": [train_error_log, train_error_tree],
    "Test Error": [test_error_log, test_error_tree]
})

comparison
```

	Model	Train Error	Test Error
0	Logistic Regression	0.013187	0.035088
1	Decision Tree	0.000000	0.070175

Next steps: Generate code with comparison New interactive sheet

Conclusion

The Logistic Regression model performs consistently on both training and test data, showing only a small difference in error. This indicates that the model generalizes well and is able to make reliable predictions on unseen cases.

On the other hand, the Decision Tree performs extremely well on the training data but shows noticeably worse performance on the test set. This gap suggests overfitting, where the model learns the training data too closely and struggles to generalize.

Based on these results, Logistic Regression is the better choice for deployment since it provides more stable and dependable performance.

Relevant ML Considerations

- Logistic Regression requires feature scaling, whereas Decision Trees do not.
- Decision Trees are more likely to overfit if not properly constrained.
- Care was taken to avoid data leakage by fitting the scaler only on training data