```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix

bnotes = pd.read_csv(r"C:\Users\Admin\Downloads\archive\bank_note_data.csv")
print("First 10 records:\n", bnotes.head(10))

# Split into input features (X) and labels (y)
x = bnotes.drop('Class', axis=1)
y = bnotes['Class']
print("\nFeature sample:\n", x.head(2))
print("\nLabel sample:\n", y.head(2))
```

```
First 10 records:
    Image.Var  Image.Skew  Image.Curt  Entropy  Class
0    3.62160      8.6661    -2.80730 -0.44699      0
1    4.54590      8.1674    -2.45860 -1.46210      0
2    3.86600     -2.6383     1.92420  0.10645      0
3    3.45660      9.5228    -4.01120 -3.59440      0
4    0.32924     -4.4552     4.57180 -0.98880      0
5    4.36840      9.6718    -3.96060 -3.16250      0
6    3.59120      3.0129     0.72888  0.56421      0
7    2.09220     -6.8100     8.46360 -0.60216      0
8    3.20320      5.7588    -0.75345 -0.61251      0
9    1.53560      9.1772    -2.27180 -0.73535      0

Feature sample:
    Image.Var  Image.Skew  Image.Curt  Entropy
0     3.6216      8.6661     -2.8073 -0.44699
1     4.5459      8.1674     -2.4586 -1.46210

Label sample:
 0    0
1    0
Name: Class, dtype: int64
```

```
# ------------------------------
# Function to train and evaluate model
# ------------------------------
def evaluate_activation(activation_name, x_train, x_test, y_train, y_test):
    mlp = MLPClassifier(max_iter=500, activation=activation_name, random_state=42)
    mlp.fit(x_train, y_train)
    pred = mlp.predict(x_test)

    cm = confusion_matrix(y_test, pred)
    cr = classification_report(y_test, pred, zero_division=0)

    print(f"\n{'='*40}")
    print(f"Activation Function: {activation_name.upper()}")
    print(pred)
    print(f"Confusion Matrix:\n{cm}")
    print(f"Classification Report:\n{cr}")
```

```
# ------------------------------
# Evaluate for test_size = 0.2
# ------------------------------
print("\n==============================")
print("TRAIN-TEST SPLIT (TEST SIZE = 0.2)")
print("==============================")

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

for activation in ['relu', 'logistic', 'tanh', 'identity']:
    evaluate_activation(activation, x_train, x_test, y_train, y_test)
```

```
==============================
TRAIN-TEST SPLIT (TEST SIZE = 0.2)
==============================

========================================
Activation Function: RELU
[0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1
 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0
 1 0 0 1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1
 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1
```

```
 1 0 1 1 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1
 0 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 0 0 0 1
 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0]
Confusion Matrix:
[[148   0]
 [  0 127]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       148
           1       1.00      1.00      1.00       127

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275


========================================
Activation Function: LOGISTIC
[0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1
 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0
 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1
 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1
 1 0 1 1 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1
 0 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1
 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0
 1 1 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0
 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0
 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 1
 1 0 1 0 0]
Confusion Matrix:
[[148   0]
 [  1 126]]
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       148
           1       1.00      0.99      1.00       127

    accuracy                           1.00       275
   macro avg       1.00      1.00      1.00       275
weighted avg       1.00      1.00      1.00       275


========================================
Activation Function: TANH
[0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1
 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0
 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1
```

```
# ------------------------------
# Evaluate for test_size = 0.3
# ------------------------------
print("\n==============================")
print("TRAIN-TEST SPLIT (TEST SIZE = 0.3)")
print("==============================")

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)

for activation in ['relu', 'logistic', 'tanh', 'identity']:
    evaluate_activation(activation, x_train, x_test, y_train, y_test)
```

```
==============================
TRAIN-TEST SPLIT (TEST SIZE = 0.3)
==============================

========================================
Activation Function: RELU
[0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1
 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0
 1 0 0 1 1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1
 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1
 1 0 1 1 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1
 0 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 1
 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0
 1 1 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 1 0 0 0 0 1 0 0 0 0 0
 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0
 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1 1
 1 0 1 0 0]
Confusion Matrix:
[[229   0]
 [  0 183]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       229
           1       1.00      1.00      1.00       183
```

```
    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412


========================================
Activation Function: LOGISTIC
[0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 0 0 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1
 1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 1 0 1 0
 1 0 0 1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1
 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1
 1 0 1 1 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 1 0 1
 0 1 1 0 0 1 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
 1 0 1 0 0 1 1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1
 1 0 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0
 1 1 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0
 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1 0 0
 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1 1 1
 1 0 1 0 0]
Confusion Matrix:
[[229   0]
 [  1 182]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       229
           1       1.00      0.99      1.00       183

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
```

Start coding or generate with AI.