```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
df=pd.read_csv('/content/BankNoteAuthentication.csv')
df.head()
```

| | variance | skewness | curtosis | entropy | class |
|---|---|---|---|---|---|
| **0** | 3.62160 | 8.6661 | -2.8073 | -0.44699 | 0 |
| **1** | 4.54590 | 8.1674 | -2.4586 | -1.46210 | 0 |
| **2** | 3.86600 | -2.6383 | 1.9242 | 0.10645 | 0 |
| **3** | 3.45660 | 9.5228 | -4.0112 | -3.59440 | 0 |
| **4** | 0.32924 | -4.4552 | 4.5718 | -0.98880 | 0 |

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
```

```
X = df.drop('class', axis=1)
y = df['class']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
activations = ['tanh', 'logistic', 'identity']
```

```
mlp_relu = MLPClassifier(hidden_layer_sizes=(10, 10),
                         activation='relu',
                         solver='adam',
                         max_iter=500,
                         early_stopping=True,
                         validation_fraction=0.1,
                         random_state=42)
mlp_relu.fit(X_train, y_train)

# Step 8: Evaluate
y_pred_relu = mlp_relu.predict(X_test)

cm = confusion_matrix(y_test, y_pred_relu)
TN, FP, FN, TP = cm.ravel()

print(f"Confusion Matrix:\n{cm}")
print(f"TN={TN}, FP={FP}, FN={FN}, TP={TP}")
print(f"Accuracy: {accuracy_score(y_test, y_pred_relu):.4f}")
print(f"Precision: {precision_score(y_test, y_pred_relu):.4f}")
print(f"Recall: {recall_score(y_test, y_pred_relu):.4f}")
print(f"F1-Score: {f1_score(y_test, y_pred_relu):.4f}")
```

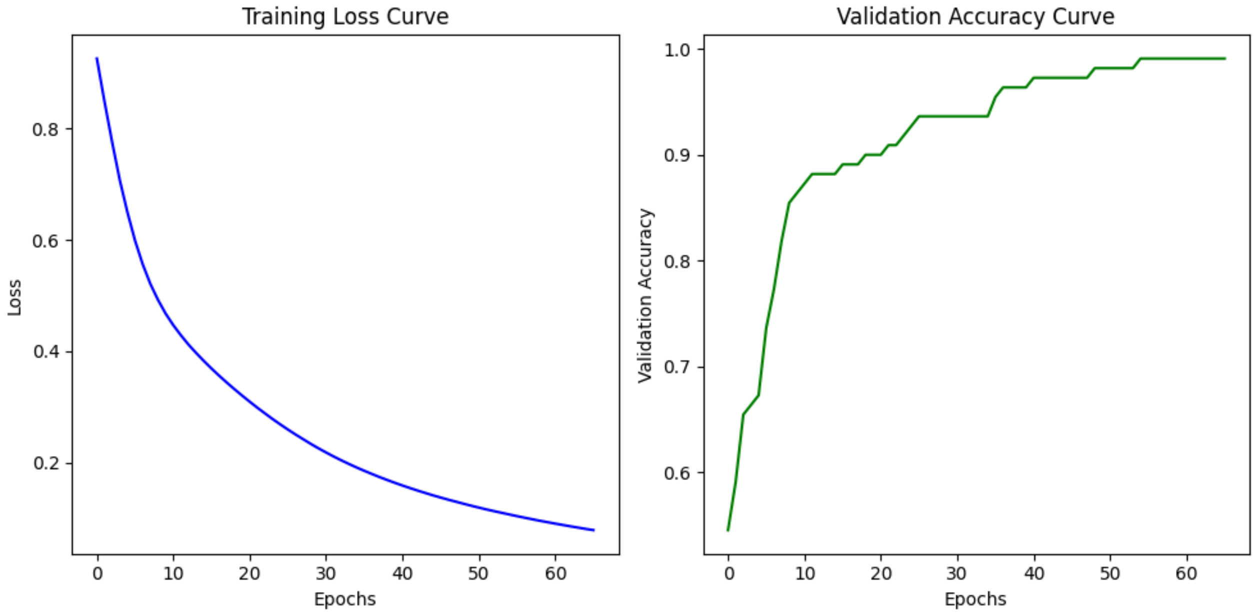
```
Confusion Matrix:
[[151   2]
 [  0 122]]
TN=151, FP=2, FN=0, TP=122
Accuracy: 0.9927
Precision: 0.9839
Recall: 1.0000
F1-Score: 0.9919
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,5))

plt.subplot(1,2,1)
plt.plot(mlp_relu.loss_curve_, color='blue')
plt.title('Training Loss Curve')
plt.xlabel('Epochs')
plt.ylabel('Loss')

plt.subplot(1,2,2)
plt.plot(mlp_relu.validation_scores_, color='green')
plt.title('Validation Accuracy Curve')
plt.xlabel('Epochs')
plt.ylabel('Validation Accuracy')

plt.tight_layout()
plt.show()
```



```
for act in activations:
    mlp = MLPClassifier(hidden_layer_sizes=(10, 10),
                        activation=act,
                        solver='adam',
                        max_iter=500,
                        early_stopping=True,
                        validation_fraction=0.1,
                        random_state=42)
    mlp.fit(X_train, y_train)
    y_pred = mlp.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"\nActivation: {act} | Accuracy: {acc:.4f}")
```

```
Activation: tanh | Accuracy: 0.9855

Activation: logistic | Accuracy: 0.5564

Activation: identity | Accuracy: 0.9745
```

```
Start coding or generate with AI.
```