Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import Adam
```

Load the data set

```
# assuming loanpred.csv is in the current directory
loan_data = pd.read_excel("/content/Copy of loan.xlsx")
```

Data Preprocessing

```
loan_data = loan_data.dropna()  ## Remove missing values
```

Feature selection and label encoding

```
features = loan_data[['ApplicantIncome', 'LoanAmount', 'Credit_History']]
labels = loan_data['Loan_Status']
labels = labels.map({'Y': 1, 'N': 0})  # Map 'Y' to 1 and 'N' to 0
print(labels)
```

```
1      0
2      1
3      1
4      1
5      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 480, dtype: int64
```

Scaling the features

```
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
##print(loan_data)
```

Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(features_scaled, labels, test_size=0.2,
##print(loan_data)
```

```
      Loan_ID  Gender Married Dependents     Education Self_Employed  \
1    LP001003    Male     Yes          1      Graduate            No
2    LP001005    Male     Yes          0      Graduate           Yes
3    LP001006    Male     Yes          0  Not Graduate            No
4    LP001008    Male      No          0      Graduate            No
5    LP001011    Male     Yes          2      Graduate           Yes
..        ...     ...     ...        ...           ...           ...
609  LP002978  Female      No          0      Graduate            No
610  LP002979    Male     Yes         3+      Graduate            No
611  LP002983    Male     Yes          1      Graduate            No
612  LP002984    Male     Yes          2      Graduate            No
613  LP002990  Female      No          0      Graduate           Yes

     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  \
1               4583             1508.0       128.0             360.0
2               3000                0.0        66.0             360.0
3               2583             2358.0       120.0             360.0
4               6000                0.0       141.0             360.0
5               5417             4196.0       267.0             360.0
..               ...                ...         ...               ...
609             2900                0.0        71.0             360.0
610             4106                0.0        40.0             180.0
611             8072              240.0       253.0             360.0
612             7583                0.0       187.0             360.0
613             4583                0.0       133.0             360.0

     Credit_History Property_Area Loan_Status
1               1.0         Rural           N
2               1.0         Urban           Y
3               1.0         Urban           Y
4               1.0         Urban           Y
5               1.0         Urban           Y
..              ...           ...         ...
609             1.0         Rural           Y
610             1.0         Rural           Y
611             1.0         Urban           Y
612             1.0         Urban           Y
613             0.0     Semiurban           N

[480 rows x 13 columns]
```

Build the classification model

```
model = Sequential([
    Dense(32, input_dim=3, activation='relu'),  # First hidden layer with 32 units
    Dense(16, activation='relu'),  # Second hidden layer with 16 units
    Dense(1, activation='sigmoid')  # Output layer for binary classification
])

model.compile(optimizer=Adam(), loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning:
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Train the model

```
X_train, X_test, y_train, y_test = train_test_split(features_scaled, labels, test_size=0.2,
```

```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=50, batch_size=32)
```

```
Epoch 38/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step - accuracy: 0.7911 - loss: 0.4868 - val_accura
Epoch 39/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.7747 - loss: 0.5171 - val_accura
Epoch 40/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.7941 - loss: 0.4962 - val_accura
Epoch 41/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.8171 - loss: 0.4673 - val_accura
Epoch 42/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - accuracy: 0.7773 - loss: 0.5251 - val_accura
Epoch 43/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 15ms/step - accuracy: 0.7840 - loss: 0.5039 - val_accura
Epoch 44/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.7937 - loss: 0.4999 - val_accura
Epoch 45/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step - accuracy: 0.7799 - loss: 0.5079 - val_accura
Epoch 46/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step - accuracy: 0.7580 - loss: 0.5349 - val_accura
Epoch 47/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step - accuracy: 0.7874 - loss: 0.4942 - val_accura
Epoch 48/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step - accuracy: 0.7781 - loss: 0.5000 - val_accura
Epoch 49/50
10/10 ━━━━━━━━━━━━━━━━━━━━ 0s 11ms/step - accuracy: 0.7898 - loss: 0.4902 - val_accura
Epoch 50/50
```

Model Evaluation

```
# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {test_accuracy * 100:.2f}%')
print(f'Test Loss: {test_loss:.4f}')
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step - accuracy: 0.8216 - loss: 0.4608
Test Accuracy: 82.29%
Test Loss: 0.4672
```

Confusion Matrix and Classification Report

```
y_pred = (model.predict(X_test) > 0.5).astype(int)  # Get predicted labels (0 or 1)
cm = confusion_matrix(y_test, y_pred)
print(f'Confusion Matrix:\n{cm}')
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step
Confusion Matrix:
[[11 17]
 [ 0 68]]
```

Generate classification report

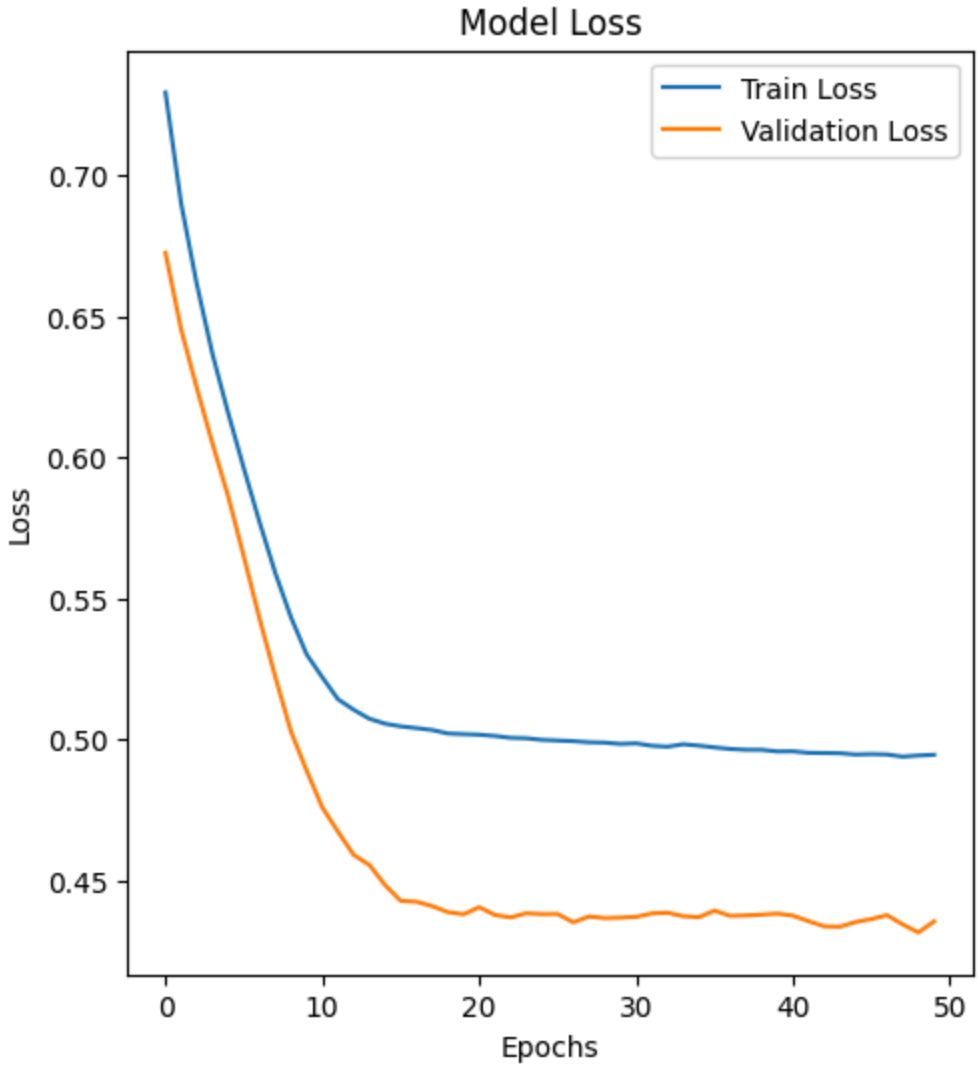
```
report = classification_report(y_test, y_pred)
print(f'Classification Report:\n{report}')
```

```
Classification Report:
                precision    recall  f1-score   support

           0       1.00      0.39      0.56        28
           1       0.80      1.00      0.89        68

    accuracy                           0.82        96
   macro avg       0.90      0.70      0.73        96
weighted avg       0.86      0.82      0.79        96
```

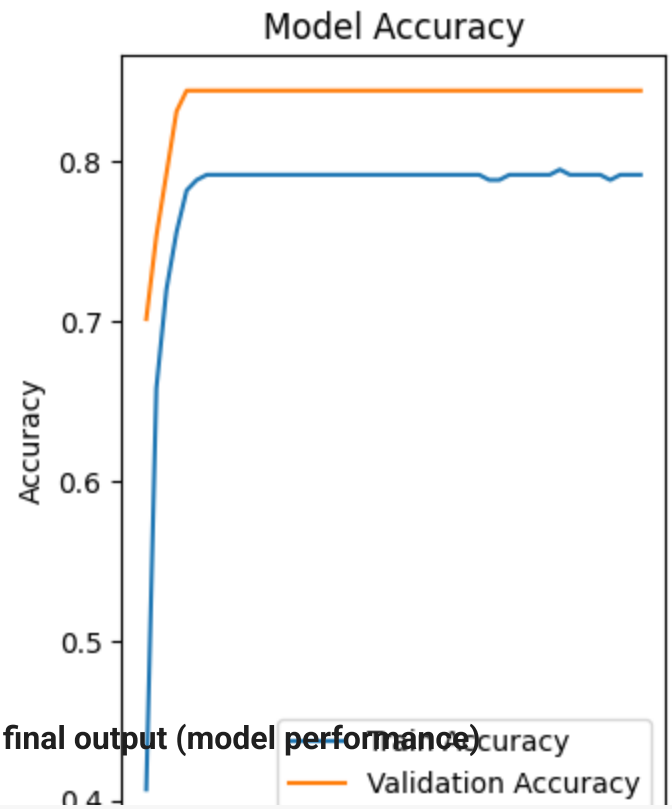
Visualize training history (loss and accuracy)

```
# Plotting loss curve
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x79da8c41fcd0>
```




```
# Plotting accuracy curve
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

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



Save final output (model performance)