import tensorflow as tf

from tensorflow.keras.models import Sequential #type:ignore

from tensorflow.keras.layers import Dense #type:ignore

from sklearn.datasets import make\_classification

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

# Generate synthetic dataset (binary classification)

X, y = make\_classification(

n\_samples=1000,

n\_features=20,

n\_informative=15,

n\_redundant=5,

random\_state=42

)

# Split into training and testing datasets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Normalize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Build the feedforward neural network model

model = Sequential([

Dense(64, activation='sigmoid', input\_shape=(X\_train.shape[1],)), # Input layer

Dense(32, activation='sigmoid'), # Hidden layer

Dense(1, activation='sigmoid') # Output layer for binary classification

])

# Compile the model

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

# Train the model

history = model.fit(X\_train, y\_train,

epochs=20,

batch\_size=32,

validation\_split=0.2,

verbose=1)

# Evaluate the model on test data

test\_loss, test\_accuracy = model.evaluate(X\_test, y\_test, verbose=0)

print(f"Test Accuracy: {test\_accuracy \* 100:.2f}%")

# Predict and evaluate

predictions = (model.predict(X\_test) > 0.5).astype("int32")

accuracy = accuracy\_score(y\_test, predictions)

print(f"Accuracy Score: {accuracy \* 100:.2f}%")

# Save the model

# model.save("feedforward\_model.h5")

# Visualize training history

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

Building and training a feedforward neural network using TensorFlow/Keras is a

foundational step in deep learning. The modular structure of these frameworks allows

developers to experiment with various configurations, enabling faster prototyping and

deployment of models in real-world applications.