import numpy as np

import tensorflow as tf

from tensorflow.keras import layers, models

from sklearn.datasets import load\_iris

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

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

# Load and preprocess the Iris dataset

X, y = load\_iris(return\_X\_y=True)

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

X\_train, X\_test = StandardScaler().fit\_transform(X\_train), StandardScaler().fit\_transform(X\_test)

# Define and compile a single-layer neural network

model\_single\_layer = models.Sequential([layers.Dense(64, 'relu', input\_shape=(4,)), layers.Dense(3, 'softmax')])

model\_single\_layer.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model\_single\_layer.fit(X\_train, y\_train, epochs=15, validation\_data=(X\_test, y\_test))

# Evaluate the single-layer model

single\_layer\_accuracy = accuracy\_score(y\_test, np.argmax(model\_single\_layer.predict(X\_test), axis=1))

print(f"\nSingle-layer Neural Network - Accuracy: {single\_layer\_accuracy}")

# Define and compile a multi-layer neural network

model\_multi\_layer = models.Sequential([layers.Dense(64, 'relu', input\_shape=(4,)), layers.Dense(32, 'relu'), layers.Dense(3, 'softmax')])

model\_multi\_layer.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model\_multi\_layer.fit(X\_train, y\_train, epochs=15, validation\_data=(X\_test, y\_test))

# Evaluate the multi-layer model

multi\_layer\_accuracy = accuracy\_score(y\_test, np.argmax(model\_multi\_layer.predict(X\_test), axis=1))

print(f"\nMulti-layer Neural Network - Accuracy: {multi\_layer\_accuracy}")