import tensorflow as tf

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

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

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

from sklearn.preprocessing import StandardScaler

from sklearn.datasets import load\_iris

from sklearn.metrics import classification\_report

from tensorflow.keras.utils import to\_categorical#type:ignore

# Load the Iris dataset

data = load\_iris()

X, y = data.data, data.target

# One-hot encode the labels

y = to\_categorical(y, num\_classes=3)

# 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 neural network model

model = Sequential([

Dense(128, activation='relu', input\_shape=(X\_train.shape[1],)),

Dropout(0.3),

Dense(64, activation='relu'),

Dropout(0.3),

Dense(3, activation='softmax') # Output layer with 3 classes

])

# Compile the model

model.compile(optimizer='adam',

loss='categorical\_crossentropy',

metrics=['accuracy'])

# Train the model

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

epochs=30,

batch\_size=8,

validation\_split=0.2,

verbose=1)

# Evaluate the model on the test data

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

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

# Generate predictions

predictions = model.predict(X\_test)

y\_pred = predictions.argmax(axis=1)

y\_true = y\_test.argmax(axis=1)

# Classification report

print("Classification Report:\n")

print(classification\_report(y\_true, y\_pred, target\_names=data.target\_names))

#OUTPUT

Test Accuracy: 100.00%

1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step

Classification Report:

precision recall f1-score support

setosa 1.00 1.00 1.00 10

versicolor 1.00 1.00 1.00 9

virginica 1.00 1.00 1.00 11

accuracy 1.00 30

macro avg 1.00 1.00 1.00 30

weighted avg 1.00 1.00 1.00 30