**Practical -10**

**Aim: To implement multiclass classification with neural network on Iris flower species.**

# Code:

import pandas as pd

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.neural\_network import MLPClassifier

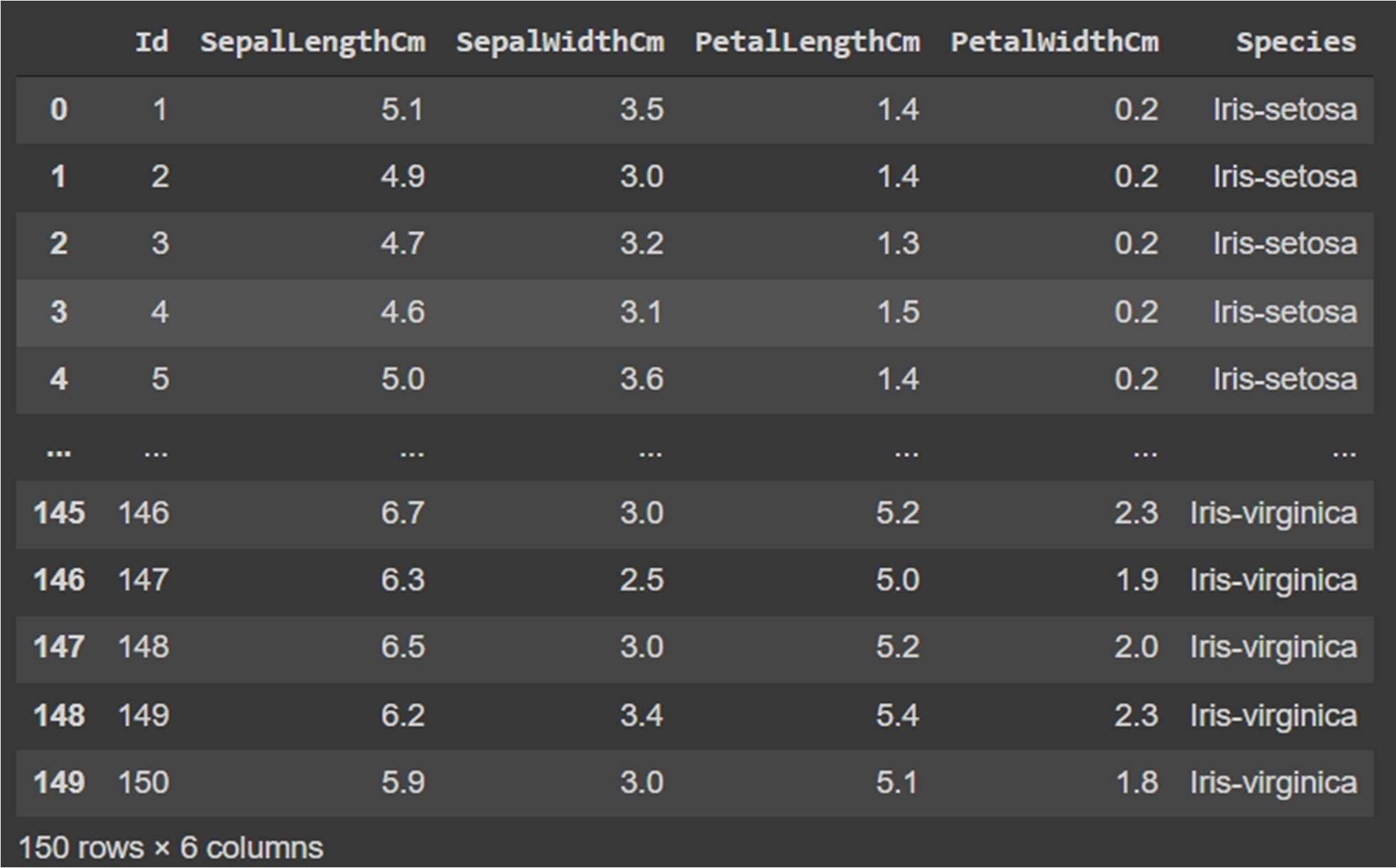
from sklearn.metrics import accuracy\_score

from sklearn.metrics import plot\_confusion\_matrix

from sklearn.metrics import classification\_report

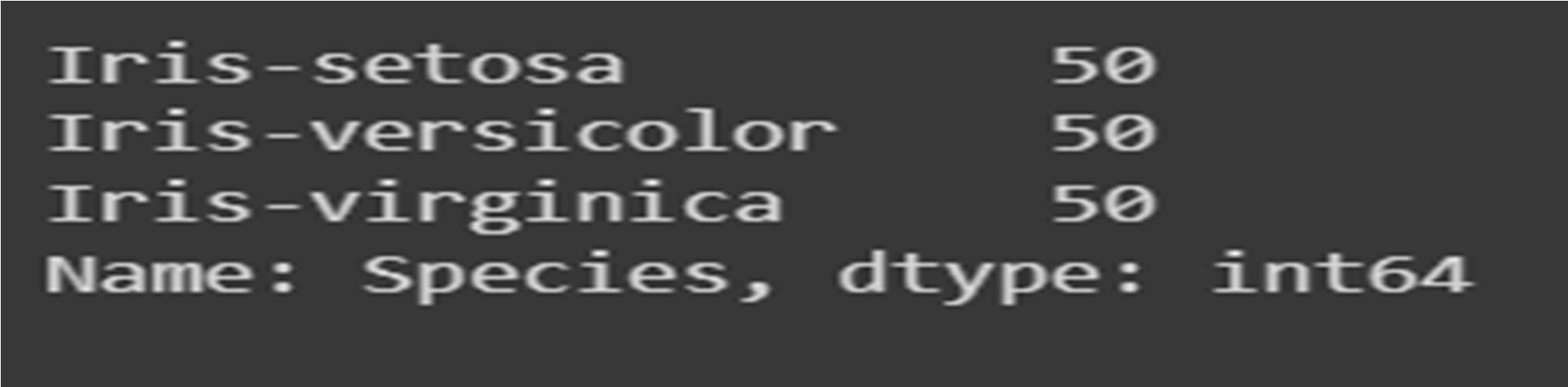
from sklearn.model\_selection import GridSearchCV

df = pd.read\_csv('Iris.csv')



# Code:

df['Species'].value\_counts()



**Code:**

x = df.drop('Species', axis=1) y = df['Species']

trainX, testX, trainY, testY = train\_test\_split(x, y, test\_size = 0.

sc=StandardScaler()

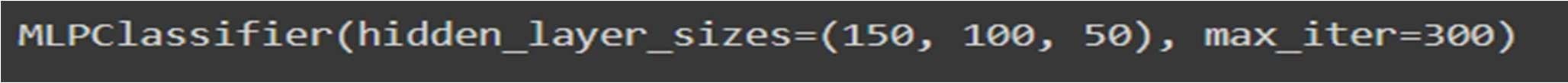
scaler = sc.fit(trainX)

trainX\_scaled = scaler.transform(trainX)

testX\_scaled = scaler.transform(testX)

mlp\_clf = MLPClassifier(hidden\_layer\_sizes=(15 0,100,50),max\_iter = 300,activation = 'relu',solver = 'adam')

mlp\_clf.fit(trainX\_scaled, trainY)



# Code:

y\_pred = mlp\_clf.predict(testX\_scaled) print('Accuracy:

{:.2f}'.format(accuracy\_score(testY, y\_pred)))

Text

Description automatically generated

# Code:

fig = plot\_confusion\_matrix(mlp\_clf, testX\_scaled, testY, display\_labels=mlp\_clf.classes\_)

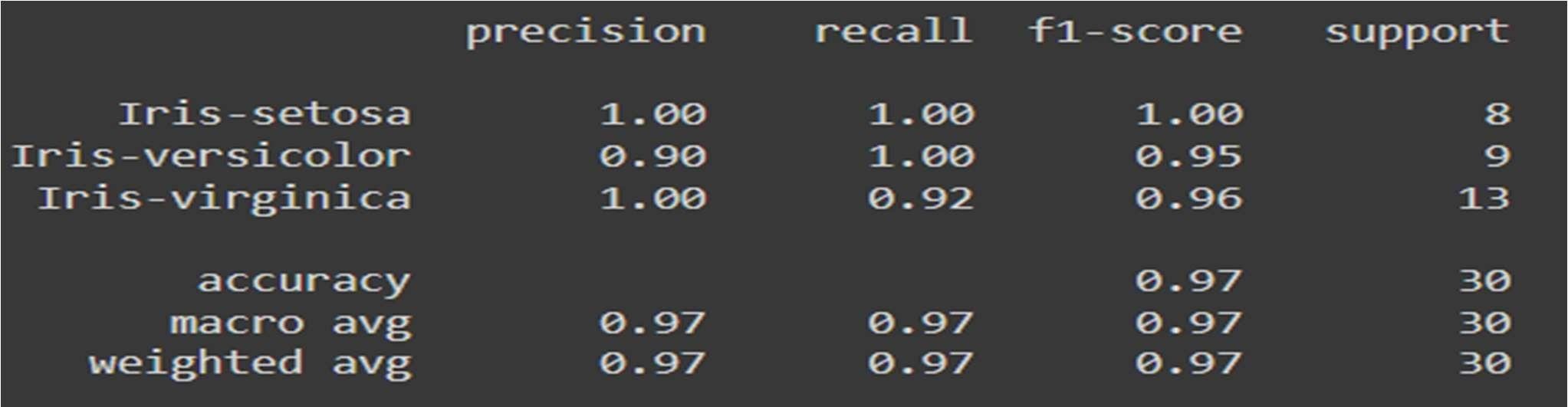
fig.figure\_.suptitle("Confusion Matrix”)

Chart

Description automatically generated

# Code:

print(classification\_report(testY, y\_pred))



plt.plot(mlp\_clf.loss\_curve\_) plt.title("Loss Curve", fontsize=14) plt.xlabel('Iterations') plt.ylabel('Cost')

plt.show()

Chart

Description automatically generated with medium confidence

# Code:

param\_grid = {

'hidden\_layer\_sizes': [(150,100,50),

(120,80,40), (100,50,30)],

'max\_iter': [50, 100, 150],

'activation': ['tanh', 'relu'],

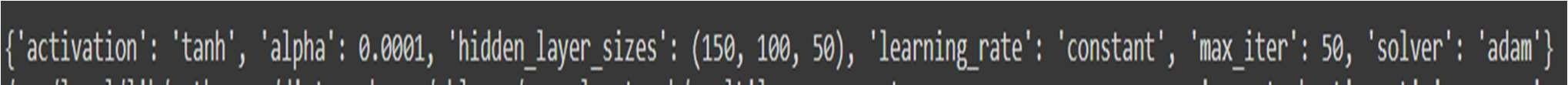
'solver': ['sgd', 'adam'],

'alpha': [0.0001, 0.05],

'learning\_rate': ['constant','adaptive'],}

grid = GridSearchCV(mlp\_clf, param\_grid, n\_jobs= -1, cv=5)

grid.fit(trainX\_scaled, trainY) print(grid.best\_params\_)



# Code:

grid\_predictions = grid.predict(testX\_scaled)

print('Accuracy:

{:.2f}'.format(accuracy\_score(testY, grid\_predictions)))



**Practical - 11**

**Aim: To implement a convolutional neural network for object recognition inan image with CIFAR 10 dataset.**

# Code:

from future import print\_function import tensorflow as tf

from tensorflow.keras.datasets import cifar10

from tensorflow.keras.preprocessing.image import ImageDataGenerator from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten from tensorflow.keras.layers import BatchNormalization

from tensorflow.keras.layers import Conv2D, MaxPooling2D from tensorflow.keras.regularizers import l2

import numpy as npimportos import matplotlib.pyplot as plt

%matplotlib inline

# Defining the parameters batch\_size = 16

num\_classes = 10

epochs = 10

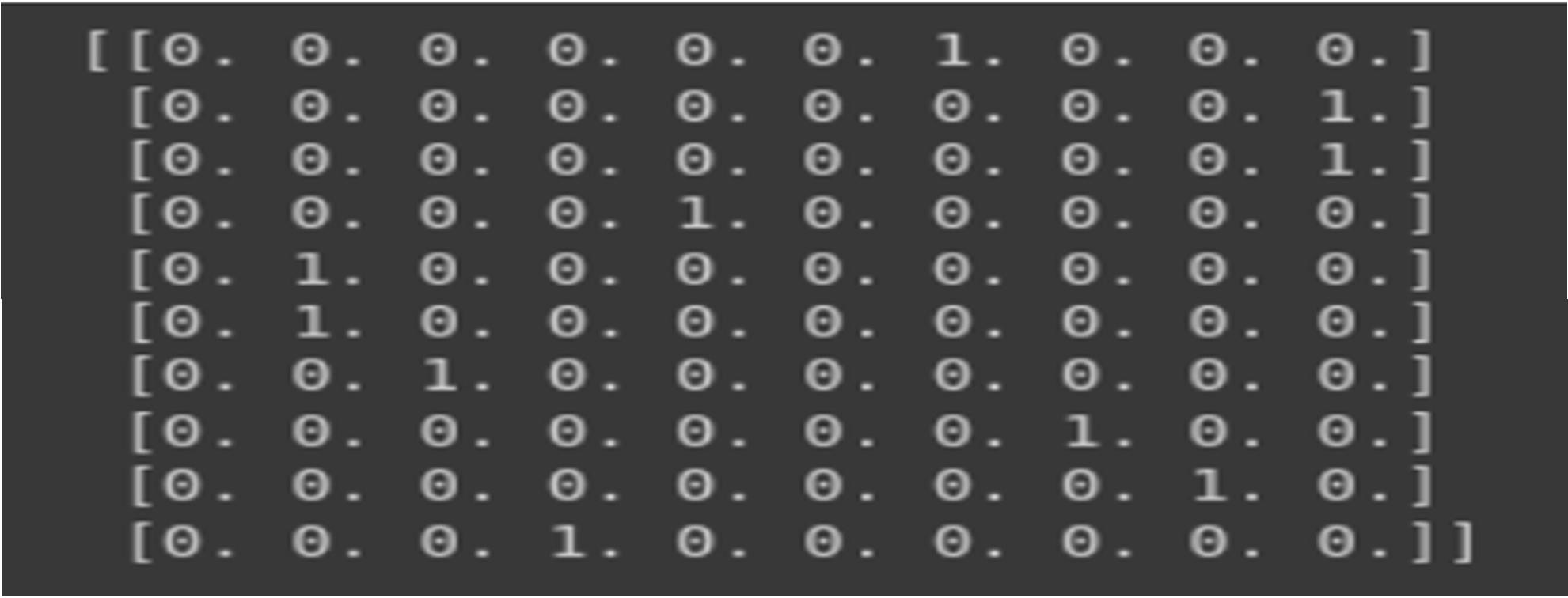
# Splitting the data between train and test (x\_train, y\_train), (x\_test, y\_test) = cifar10.load\_data()print('x\_train shape:', x\_train.shape) print(x\_train.shape[0], 'train samples') print(x\_test.shape[0], 'test samples')



# Code:

# Convert class vectors to binary class matrices. y\_train = tf.keras.utils.to\_categorical(y\_train, num\_classes)y\_test = tf.keras.utils.to\_categorical(y\_test, num\_classes)

# Printing sample data print(y\_train[:10])



# Code:

model = Sequential() model.add(Conv2D(64, (3, 3), padding='same', input\_shape=x\_train.shape[1:])) model.add(Activation('relu')) model.add(BatchNormalization()) model.add(Conv2D(64, (3, 3))) model.add(Activation('relu'))

model.add(BatchNormalization()) model.add(MaxPooling2D(pool\_size=(2,2)) model.add(Dropout(0.25)) model.add(Conv2D(128, (3, 3), padding='same'))model.add(Activation('relu')) model.add(BatchNormalization()) model.add(Conv2D(128, (3, 3))) model.add(Activation('relu')) model.add(BatchNormalization()) model.add(MaxPooling2D(pool\_size=(2, 2))) model.add(Dropout(0.25)) model.add(Flatten()) model.add(Dense(512,kernel\_regularizer=l2(0.01)) model.add(Activation('relu')) model.add(Dropout(0.5)) model.add(Dense(num\_classes)) model.add(Activation('softmax'))

# compile model.compile(loss='categorical\_crossentro’, optimizer='sgd',

metrics=['accuracy']) x\_train=x\_train.astype('float3') x\_test=x\_test.astype('float32') # Normalizing the input imagex\_train /= 255

x\_test /= 255

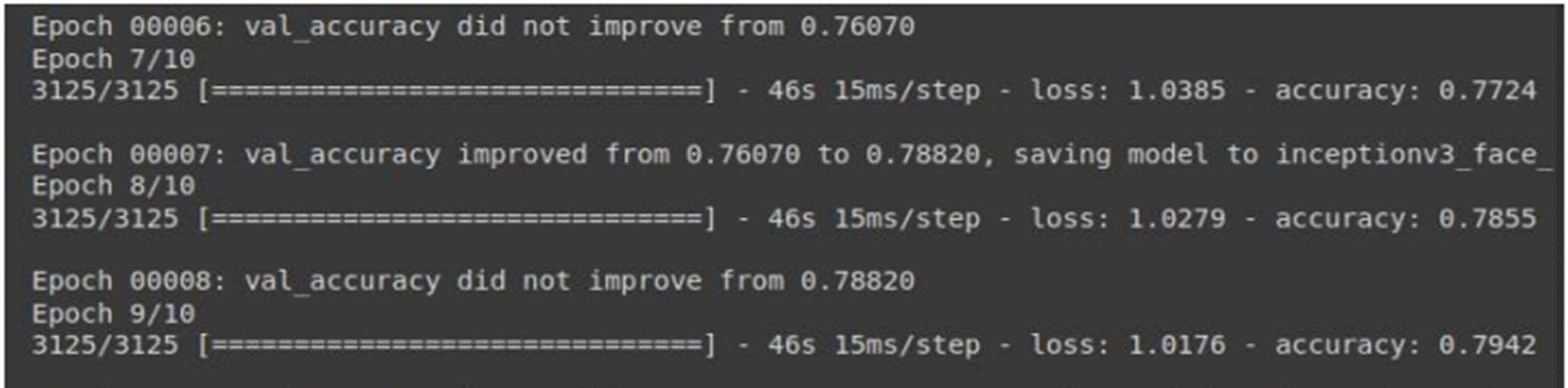
from keras.callbacksimportModelCheckpoit model\_path = 'inceptionv3\_face\_bg.h5'

checkpoint = ModelCheckpoint(model\_path, monitor='val\_accuracy', verbose=1,

save\_best\_only=True, mode='max') callbacks\_list=[checkpoint]

# Training the model history = model.fit(x\_train,

y\_train,batch\_size=batch\_size, epochs=epochs, validation\_data=(x\_test, y\_test), callbacks=callbacks\_list

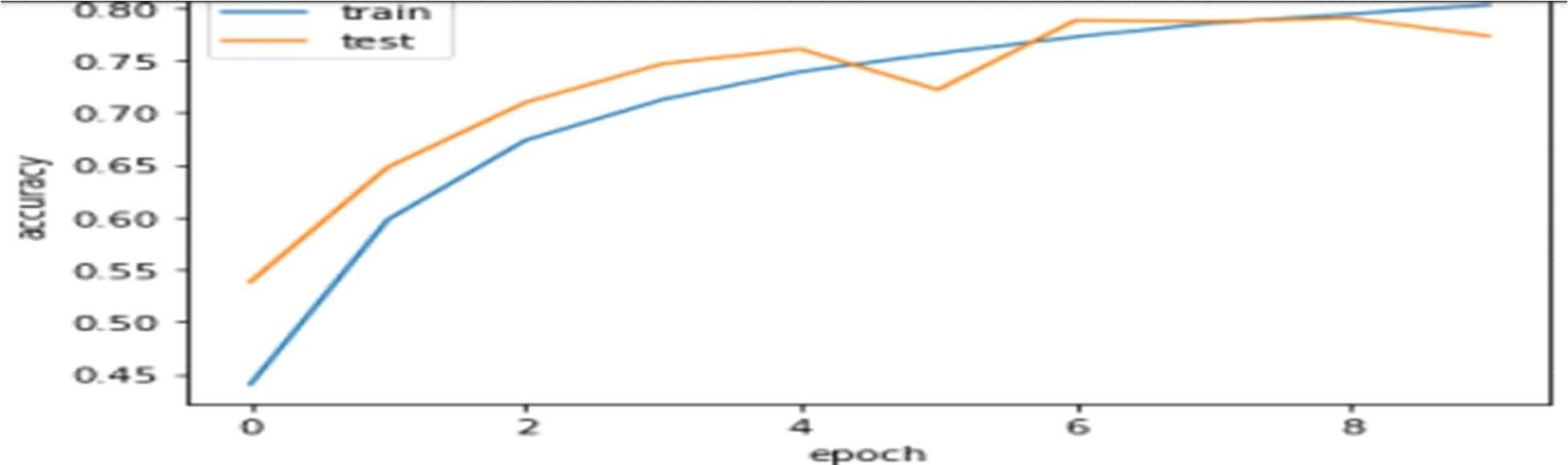


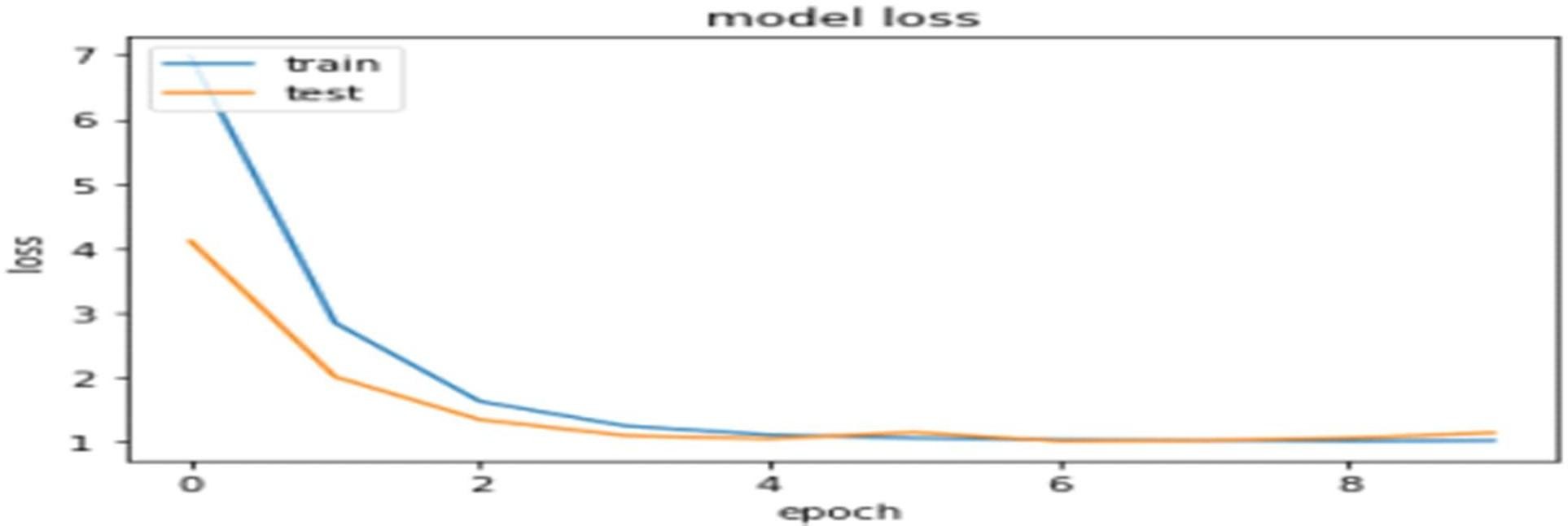
# Code:

# list all data in history print(history.history.keys())

# summarize history for accuracy plt.plot(history.history['accuracy']) plt.plot(history.history['val\_accuracy '])plt.title('model accuracy') plt.ylabel('accuracy') plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')plt.show()





# Code:

import keras

from keras.models import load\_model from keras.preprocessing import image import numpy as np

import keras

from tensorflow.keras.models import load\_model from tensorflow.keras.preprocessing import image from matplotlib import pyplot as plt

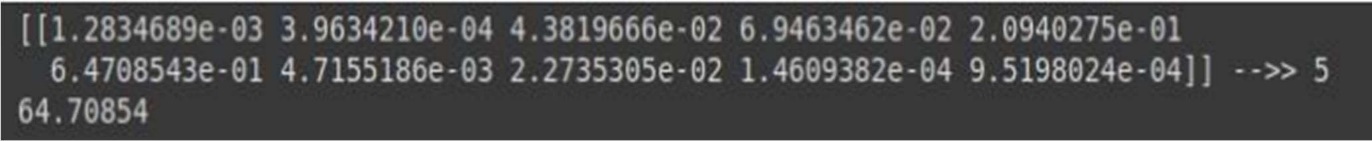
import numpy as np import cv2

import numpy as np import cv2

#img = cv2.imread('/content/index.jpeg')

model = keras.models.load\_model('/content/inceptionv3\_face\_bg.h5') t\_img = image.load\_img("/content/index.jpeg", target\_size = (32,32)) t\_img = image.img\_to\_array(t\_img)/255

t\_img = np.expand\_dims(t\_img, axis = 0) result = model.predict(t\_img)

pred = np.argmax(result) print(result, "-->>",pred) print(max(result[0]\*1003

**Practical-12**

**Aim: To implement sequence classification of Movie reviews with LSTM network.**

# Code:

import numpy

from keras.datasets import imdb from keras.models import Sequential from keras.layers import Dense

from keras.layers import LSTM

from keras.layers.embeddings import Embedding from keras.preprocessing import sequence

# fix random seed for reproducibility numpy.random.seed(7)

# load the dataset but only keep the top n words, zero the rest top\_words = 5000

(X\_train, y\_train), (X\_test, y\_test) = imdb.load\_data(num\_words=top\_words)

# truncate and pad input sequences max\_review\_length = 500

X\_train = sequence.pad\_sequences(X\_train, maxlen=max\_review\_length) X\_test = sequence.pad\_sequences(X\_test, maxlen=max\_review\_length)

# create the model embedding\_vecor\_length = 32 model = Sequential()

model.add(Embedding(top\_words, embedding\_vecor\_length, input\_length=max\_review\_length)) model.add(LSTM(100))

model.add(Dense(1, activation='sigmoid')) model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy']) print(model.summary())

model.fit(X\_train, y\_train, epochs=3, batch\_size=64)

# Final evaluation of the model

scores = model.evaluate(X\_test, y\_test, verbose=0) print("Accuracy: %.2f%%" % (scores[1]\*100))

