***ASSIGNMENT:-2***

import tensorflow as tf

from tensorflow import keras

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import random

get\_ipython().run\_line\_magic("matplotlib","inline")

mnist = tf.keras.datasets.mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

len(x\_train)

len(x\_test)

x\_train.shape

x\_test.shape

x\_train[0]

plt.matshow(x\_train[11]) #we can change it by changing the argument

x\_train = x\_train/255

x\_test = x\_test/255

x\_train[11]

model = keras.Sequential([

    keras.layers.Flatten(input\_shape=(28, 28)),

    keras.layers.Dense(128, activation='relu'),

    keras.layers.Dense(10, activation='softmax')

])

model.summary()

model.compile(optimizer='sgd',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

history=model.fit(x\_train, y\_train,validation\_data=(x\_test,y\_test),epochs=10)

test\_loss, test\_acc=model.evaluate(x\_test,y\_test)

print("Loss=%.3f" %test\_loss)

print("Accuracy=%.3f" %test\_acc)

n=random.randint(0,9999)

plt.imshow(x\_test[n])

plt.show()

predicted\_value=model.predict(x\_test)

print("Handwritten nuber in the image is= %d" %np.argmax(predicted\_value))

get\_ipython().run\_line\_magic('pinfo2','history.history')

history.history.keys()

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['Train', 'Validation'], loc='upper left')

plt.show()

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('Training Loss and accuracy')

plt.ylabel('accuracy/Loss')

plt.xlabel('epoch')

plt.legend(['accuracy', 'val\_accuracy','loss','val\_loss'])

plt.show()

keras\_model\_path="/content/sample\_data"

model.save(keras\_model\_path)

restored\_keras\_model = tf.keras.models.load\_model(keras\_model\_path)