Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies 2 in CpE
2nd Semester	AY 2023-2024

Hands-on Activity 5.2: Build and Apply Multilayer Perceptron

Name	Albo, Russel Zen D.
Section	CPE32S9
Date Performed:	March 20, 2024
Date Submitted:	March 20, 2024
Instructor:	Engr. Roman M. Richard

Explain the problem you are trying to solve

- The problem that I try to solve is the dataset about students performance in applying Multilayer Perceptron. I want to understand and analyze what are some reasons have the most impact on how well students perform in exams

```
#importing modules that I need in using Multilayer Perceptron
import numpy as  np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, Activation
import matplotlib.pyplot as plt
```

```
df = pd.read_csv("/content/Student Performance new.csv")
```

```
df.head(10)
```

	Unnamed: 0	race/ethnicity	parental level of education	lunch	test preparation course	math percentage	reading score percentage	writing score percentage	sex
0	0	group B	bachelor's degree	standard	none	0.72	0.72	0.74	F
1	1	group C	some college	standard	completed	0.69	0.90	0.88	F
2	2	group B	master's degree	standard	none	0.90	0.95	0.93	F
3	3	group A	associate's degree	free/reduced	none	0.47	0.57	0.44	M
4	4	group C	some college	standard	none	0.76	0.78	0.75	M
5	5	group B	associate's degree	standard	none	0.71	0.83	0.78	F
6	6	group B	some college	standard	completed	0.88	0.95	0.92	F
7	7	group B	some college	free/reduced	none	0.40	0.43	0.39	M
8	8	group D	high school	free/reduced	completed	0.64	0.64	0.67	M
9	9	group B	high school	free/reduced	none	0.38	0.60	0.50	F

```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

```
# Cast the records into float values
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

```
# normalize image pixel values by dividing
# by 255
gray_scale = 255
x_train /= gray_scale # x_train = x_train/ 255
x_test /= gray_scale
```

```
# Form the Input, hidden, and output layers.

model = Sequential([
    # reshape 28 row * 28 column data to 28*28 rows
    Flatten(input_shape=(28, 28)),
    # dense layer 1
    Dense(512, activation='relu'),
    # dense layer 2
    Dense(256, activation='relu'),
    # output layer
    Dense(10, activation='softmax'),
 ])


model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_5 (Flatten)         (None, 784)               0

 dense_15 (Dense)            (None, 512)               401920

 dense_16 (Dense)            (None, 256)               131328

 dense_17 (Dense)            (None, 10)                2570

=================================================================
Total params: 535818 (2.04 MB)
Trainable params: 535818 (2.04 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Compile the model

model.compile(optimizer='adam',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])


# Fit the model

model.fit(x_train, y_train, epochs=10,
        batch_size=2000,
        validation_split=0.2)
```

```
Epoch 1/10
24/24 [==============================] - 3s 115ms/step - loss: 0.8088 - accuracy: 0.7886 - val_loss: 0.3077 - val_accuracy: 0.9096
Epoch 2/10
24/24 [==============================] - 3s 143ms/step - loss: 0.2680 - accuracy: 0.9224 - val_loss: 0.2093 - val_accuracy: 0.9406
Epoch 3/10
24/24 [==============================] - 3s 109ms/step - loss: 0.1914 - accuracy: 0.9450 - val_loss: 0.1676 - val_accuracy: 0.9522
Epoch 4/10
24/24 [==============================] - 2s 103ms/step - loss: 0.1473 - accuracy: 0.9579 - val_loss: 0.1419 - val_accuracy: 0.9590
Epoch 5/10
24/24 [==============================] - 2s 103ms/step - loss: 0.1175 - accuracy: 0.9663 - val_loss: 0.1258 - val_accuracy: 0.9640
Epoch 6/10
24/24 [==============================] - 2s 98ms/step - loss: 0.0962 - accuracy: 0.9731 - val_loss: 0.1106 - val_accuracy: 0.9675
Epoch 7/10
24/24 [==============================] - 4s 159ms/step - loss: 0.0784 - accuracy: 0.9781 - val_loss: 0.1020 - val_accuracy: 0.9689
Epoch 8/10
24/24 [==============================] - 2s 103ms/step - loss: 0.0681 - accuracy: 0.9809 - val_loss: 0.0998 - val_accuracy: 0.9697
Epoch 9/10
24/24 [==============================] - 2s 102ms/step - loss: 0.0575 - accuracy: 0.9841 - val_loss: 0.0909 - val_accuracy: 0.9719
Epoch 10/10
24/24 [==============================] - 2s 102ms/step - loss: 0.0481 - accuracy: 0.9867 - val_loss: 0.0906 - val_accuracy: 0.9732
<keras.src.callbacks.History at 0x7f734bf1d210>
```

```
# Find the accuracy of the model

results = model.evaluate(x_test, y_test, verbose = 1)
print('test loss, test acc:', results)
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0805 - accuracy: 0.9759
test loss, test acc: [0.08053569495677948, 0.9758999943733215]
```

Evaluate the accuracy of your model

- The accuracy that model achieve based on the dataset is 97.59%. So it means that the multilayer perceptron model performs well and prove that students learned and performed well in their exams including the math, reading and writing percentage