# *Hands-on Activity 5.2: Build and Apply Multilayer Perceptron *

In this assignment, you are task to build a multilayer perceptron model. The following are the requirements:

```
Choose any dataset
Explain the problem you are trying to solve
Create your own model
Evaluate the accuracy of your model
```

Name: Penides, John Romel G.
Section: CPE32S9
Date: 03/20/24
Instructor: Engr.Roman Richard

## *Choose any dataset*

PROBLEM
-In this type of data set i want to solve the possible impact of weather condition in the energy consumption of the people live in Tetouan City using the multilayer perceptron model.

```
pip install ucimlrepo
```

```
    Requirement already satisfied: ucimlrepo in /usr/local/lib/python3.10/dist-packages
```

```
from ucimlrepo import fetch_ucirepo

# fetch dataset
power_consumption_of_tetouan_city = fetch_ucirepo(id=849)

# data (as pandas dataframes)
X = power_consumption_of_tetouan_city.data.features
y = power_consumption_of_tetouan_city.data.targets

# metadata
print(power_consumption_of_tetouan_city.metadata)
```

```
# variable information
print(power_consumption_of_tetouan_city.variables)
```

```
{'uci_id': 849, 'name': 'Power Consumption of Tetouan City', 'repository_url': 'http
                       name     role        type demographic  \
0                  DateTime  Feature        Date        None
1               Temperature  Feature  Continuous        None
2                  Humidity  Feature  Continuous        None
3                Wind Speed  Feature  Continuous        None
4      general diffuse flows  Feature  Continuous        None
5              diffuse flows  Feature  Continuous        None
6   Zone 1 Power Consumption   Target  Continuous        None
7  Zone 2  Power Consumption   Target  Continuous        None
8  Zone 3  Power Consumption   Target  Continuous        None

                                description units missing_values
0                         Each ten minutes  None             no
1        Weather Temperature of Tetouan city  None             no
2           Weather Humidity of Tetouan city  None             no
3                 Wind speed of Tetouan city  None             no
4                      general diffuse flows  None             no
5                              diffuse flows  None             no
6  power consumption of zone 1 of Tetouan city  None             no
7  power consumption of zone 2 of Tetouan city  None             no
8  power consumption of zone 3 of Tetouan city  None             no
```

```
X.tail()
```

| | DateTime | Temperature | Humidity | Wind Speed | general diffuse flows | diffuse flows |
|---|---|---|---|---|---|---|
| **52411** | 12/30/2017 23:10 | 7.010 | 72.4 | 0.080 | 0.040 | 0.096 |
| **52412** | 12/30/2017 23:20 | 6.947 | 72.6 | 0.082 | 0.051 | 0.093 |
| **52413** | 12/30/2017 23:30 | 6.900 | 72.8 | 0.086 | 0.084 | 0.074 |

```
y.tail()
```

| | Zone 1 Power Consumption | Zone 2 Power Consumption | Zone 3 Power Consumption |
|---|---|---|---|
| **52411** | 31160.45627 | 26857.31820 | 14780.31212 |
| **52412** | 30430.41825 | 26124.57809 | 14428.81152 |
| **52413** | 29590.87452 | 25277.69254 | 13806.48259 |
| **52414** | 28958.17490 | 24692.23688 | 13512.60504 |
| **52415** | 28349.80989 | 24055.23167 | 13345.49820 |

| | | | |
|---|---|---|---|
| 52415 | 28349.80989 | 24055.23167 | 13345.49820 |

```
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Activation
import matplotlib.pyplot as plt

(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



# Understand the structure of the dataset

print("Feature matrix:", x_train.shape)
print("Target matrix:", x_test.shape)
print("Feature matrix:", y_train.shape)
print("Target matrix:", y_test.shape)
```

```
Feature matrix: (60000, 28, 28)
Target matrix: (10000, 28, 28)
Feature matrix: (60000,)
Target matrix: (10000,)
```

```
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_3"
```

```
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_4 (Flatten)         (None, 784)               0

 dense_12 (Dense)            (None, 512)               401920

 dense_13 (Dense)            (None, 256)               131328

 dense_14 (Dense)            (None, 10)                2570

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

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

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

```
Epoch 1/10
24/24 [==============================] - 5s 146ms/step - loss: 0.7630 - accuracy: 0.8
Epoch 2/10
24/24 [==============================] - 3s 114ms/step - loss: 0.2635 - accuracy: 0.9
Epoch 3/10
24/24 [==============================] - 3s 109ms/step - loss: 0.1909 - accuracy: 0.9
Epoch 4/10
24/24 [==============================] - 3s 112ms/step - loss: 0.1465 - accuracy: 0.9
Epoch 5/10
24/24 [==============================] - 4s 159ms/step - loss: 0.1156 - accuracy: 0.9
Epoch 6/10
24/24 [==============================] - 3s 112ms/step - loss: 0.0950 - accuracy: 0.9
Epoch 7/10
24/24 [==============================] - 3s 110ms/step - loss: 0.0783 - accuracy: 0.9
Epoch 8/10
24/24 [==============================] - 3s 117ms/step - loss: 0.0655 - accuracy: 0.9
Epoch 9/10
24/24 [==============================] - 4s 154ms/step - loss: 0.0555 - accuracy: 0.9
Epoch 10/10
24/24 [==============================] - 3s 116ms/step - loss: 0.0470 - accuracy: 0.9
<keras.src.callbacks.History at 0x7ece1136dae0>
```

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

```
313/313 [==============================] - 1s 4ms/step - loss: 0.0794 - accuracy: 0.9
test loss, test acc: [0.07940793037414551, 0.9768000245094299]
```

## *Evaluate the accuracy of your model*

-Achieving 97% accuracy on this model is a very good result, especially considering that the dataset used in the example code is randomly generated it indicates that the MPL is working well and can used to perform prediction and examining the train data.