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

Assignment 5.2	**Build and Apply Multilayer Perceptron**
Name	Guevarra, Hans Angelo C.
Section	CPE32S3
Date Performed:	3-25-24
Date Submitted:	3-25-24
Instructor:	Engr. Roman M. Richard

Explain the problem you are trying to solve.

I chose the SUDS dataset, which stands for Soap Underwater Detection System. The problem that it is trying to solve is the inefficiency and wastage of water rinsing dishes that are already clean caused by automated dishwashers in homes, which run on preset timers by detecting the cleanliness of dishwasher water to end dishwashing cycles when all detergent is washed off the dishes.

This dataset aims to predict whether a sample is soapy or clean.

In [22]:

```
import numpy as np
import pandas as pd

SUDS = pd.read_csv("/content/combined.csv")
SUDS.head()
```

Out[22]:

	Function	Sample num	Time	SMUX config	ATIME	ASTEP	T_INT [ms]	WTIME_EN	WTIME	WTIME [ms]	...	Corr F1 (410nm)	Corr F2 (440nm)	Cor (470
0	clean	10	15.07.2020-22:49:18	4	0	65534	182.1873	False	0	2.78	...	0.035662	0.073566	0.106
1	clean	11	15.07.2020-22:49:19	4	0	65534	182.1873	False	0	2.78	...	0.034992	0.072179	0.104
2	clean	12	15.07.2020-22:49:20	4	0	65534	182.1873	False	0	2.78	...	0.035909	0.073981	0.107
3	clean	13	15.07.2020-22:49:21	4	0	65534	182.1873	False	0	2.78	...	0.036507	0.075237	0.109
4	clean	14	15.07.2020-22:49:21	4	0	65534	182.1873	False	0	2.78	...	0.036095	0.074362	0.108

5 rows × 45 columns

In [23]:

```
SUDS.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2040 entries, 0 to 2039
```

```
Data columns (total 45 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Function           2040 non-null   object
 1   Sample num         2040 non-null   int64
 2   Time               2040 non-null   object
 3   SMUX config        2040 non-null   int64
 4   ATIME              2040 non-null   int64
 5   ASTEP              2040 non-null   int64
 6   T_INT [ms]         2040 non-null   float64
 7   WTIME_EN           2040 non-null   bool
 8   WTIME              2040 non-null   int64
 9   WTIME [ms]         2040 non-null   float64
 10  AGAIN              2040 non-null   int64
 11  AGAIN [x]          2040 non-null   int64
 12  LED_EN             2040 non-null   bool
 13  LED_Current        2040 non-null   int64
 14  LED_Current [mA]   2040 non-null   int64
 15  F1 (410nm)         2040 non-null   int64
 16  F2 (440nm)         2040 non-null   int64
 17  F3 (470nm)         2040 non-null   int64
 18  F4 (510nm)         2040 non-null   int64
 19  F5 (550nm)         2040 non-null   int64
 20  F6 (583nm)         2040 non-null   int64
 21  F7 (620nm)         2040 non-null   int64
 22  F8 (670nm)         2040 non-null   int64
 23  CLEAR              2040 non-null   int64
 24  NIR                2040 non-null   int64
 25  Basic F1 (410nm)   2040 non-null   float64
 26  Basic F2 (440nm)   2040 non-null   float64
 27  Basic F3 (470nm)   2040 non-null   float64
 28  Basic F4 (510nm)   2040 non-null   float64
 29  Basic F5 (550nm)   2040 non-null   float64
 30  Basic F6 (583nm)   2040 non-null   float64
 31  Basic F7 (620nm)   2040 non-null   float64
 32  Basic F8 (670nm)   2040 non-null   float64
 33  Basic CLEAR        2040 non-null   float64
 34  Basic NIR          2040 non-null   float64
 35  Corr F1 (410nm)    2040 non-null   float64
 36  Corr F2 (440nm)    2040 non-null   float64
 37  Corr F3 (470nm)    2040 non-null   float64
 38  Corr F4 (510nm)    2040 non-null   float64
 39  Corr F5 (550nm)    2040 non-null   float64
 40  Corr F6 (583nm)    2040 non-null   float64
 41  Corr F7 (620nm)    2040 non-null   float64
 42  Corr F8 (670nm)    2040 non-null   float64
 43  Corr CLEAR         2040 non-null   float64
 44  Corr NIR           2040 non-null   float64
dtypes: bool(2), float64(22), int64(19), object(2)
memory usage: 689.4+ KB
```

Observation: Upon using the .info command, I noticed that there are some non-numerical data.

In [24]:

```
SUDS["LED_EN"] = SUDS["LED_EN"].apply(lambda toLabel: 0 if toLabel == "False" else 1)
```

In [25]:

```
SUDS["WTIME_EN"] = SUDS["WTIME_EN"].apply(lambda toLabel: 0 if toLabel == "False" else 1
)
```

By using lambda, I converted the boolean data into a numerical value, indicating "false" as 0.

In [27]:

```
SUDS
```

Out[27]:

	Function	Sample num	Time	SMUX config	ATIME	ASTEP	T_INT [ms]	WTIME_EN	WTIME	WTIME [ms]	...	Corr F1 (410nm)	Corr F2 (440nm)	
0	clean	10	15.07.2020-22:49:18	4	0	65534	182.1873	1	0	2.78	...	0.035662	0.073566	0.
1	clean	11	15.07.2020-22:49:19	4	0	65534	182.1873	1	0	2.78	...	0.034992	0.072179	0.
2	clean	12	15.07.2020-22:49:20	4	0	65534	182.1873	1	0	2.78	...	0.035909	0.073981	0.
3	clean	13	15.07.2020-22:49:21	4	0	65534	182.1873	1	0	2.78	...	0.036507	0.075237	0.
4	clean	14	15.07.2020-22:49:21	4	0	65534	182.1873	1	0	2.78	...	0.036095	0.074362	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2035	liquid	656	22.07.2020-17:07:28	4	0	65534	182.1873	1	0	2.78	...	0.010347	0.021174	0.
2036	liquid	657	22.07.2020-17:07:29	4	0	65534	182.1873	1	0	2.78	...	0.010368	0.021217	0.
2037	liquid	658	22.07.2020-17:07:30	4	0	65534	182.1873	1	0	2.78	...	0.010389	0.021259	0.
2038	liquid	659	22.07.2020-17:07:31	4	0	65534	182.1873	1	0	2.78	...	0.010410	0.021280	0.
2039	liquid	660	22.07.2020-17:07:31	4	0	65534	182.1873	1	0	2.78	...	0.010411	0.021323	0.

2040 rows × 45 columns

In [28]:

```
SUDS["Function"] = SUDS["Function"].apply(lambda toLabel: 1 if toLabel == "clean" else 0
)
```

By using lambda, I converted the "Function" columns which represents whether the water is clean or not into numerical value.

In [29]:

```
SUDS.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2040 entries, 0 to 2039
Data columns (total 45 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Function          2040 non-null   int64  
 1   Sample num        2040 non-null   int64  
 2   Time              2040 non-null   object 
 3   SMUX config       2040 non-null   int64  
 4   ATIME             2040 non-null   int64  
 5   ASTEP             2040 non-null   int64  
 6   T_INT [ms]        2040 non-null   float64
 7   WTIME_EN          2040 non-null   int64  
 8   WTIME             2040 non-null   int64  
 9   WTIME [ms]        2040 non-null   float64
 10  AGAIN             2040 non-null   int64  
 11  AGAIN [x]         2040 non-null   int64  
 12  LED_EN            2040 non-null   int64  
 13  LED_Current       2040 non-null   int64  
 14  LED_Current [mA]  2040 non-null   int64  
 15  F1 (410nm)        2040 non-null   int64  
 16  F2 (440nm)        2040 non-null   int64  
 17  F3 (470nm)        2040 non-null   int64  
 18  F4 (510nm)        2040 non-null   int64  
 19  F5 (550nm)        2040 non-null   int64  
```

```
 20  F6 (583nm)         2040 non-null   int64
 21  F7 (620nm)         2040 non-null   int64
 22  F8 (670nm)         2040 non-null   int64
 23  CLEAR              2040 non-null   int64
 24  NIR                2040 non-null   int64
 25  Basic F1 (410nm)   2040 non-null   float64
 26  Basic F2 (440nm)   2040 non-null   float64
 27  Basic F3 (470nm)   2040 non-null   float64
 28  Basic F4 (510nm)   2040 non-null   float64
 29  Basic F5 (550nm)   2040 non-null   float64
 30  Basic F6 (583nm)   2040 non-null   float64
 31  Basic F7 (620nm)   2040 non-null   float64
 32  Basic F8 (670nm)   2040 non-null   float64
 33  Basic CLEAR        2040 non-null   float64
 34  Basic NIR          2040 non-null   float64
 35  Corr F1 (410nm)    2040 non-null   float64
 36  Corr F2 (440nm)    2040 non-null   float64
 37  Corr F3 (470nm)    2040 non-null   float64
 38  Corr F4 (510nm)    2040 non-null   float64
 39  Corr F5 (550nm)    2040 non-null   float64
 40  Corr F6 (583nm)    2040 non-null   float64
 41  Corr F7 (620nm)    2040 non-null   float64
 42  Corr F8 (670nm)    2040 non-null   float64
 43  Corr CLEAR         2040 non-null   float64
 44  Corr NIR           2040 non-null   float64
dtypes: float64(22), int64(22), object(1)
memory usage: 717.3+ KB
```

In [30]:

```
SUDS
```

Out[30]:

	Function	Sample num	Time	SMUX config	ATIME	ASTEP	T_INT [ms]	WTIME_EN	WTIME	WTIME [ms]	...	Corr F1 (410nm)	Corr F2 (440nm)	
0	1	10	15.07.2020-22:49:18	4	0	65534	182.1873	1	0	2.78	...	0.035662	0.073566	0.
1	1	11	15.07.2020-22:49:19	4	0	65534	182.1873	1	0	2.78	...	0.034992	0.072179	0.
2	1	12	15.07.2020-22:49:20	4	0	65534	182.1873	1	0	2.78	...	0.035909	0.073981	0.
3	1	13	15.07.2020-22:49:21	4	0	65534	182.1873	1	0	2.78	...	0.036507	0.075237	0.
4	1	14	15.07.2020-22:49:21	4	0	65534	182.1873	1	0	2.78	...	0.036095	0.074362	0.
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2035	0	656	22.07.2020-17:07:28	4	0	65534	182.1873	1	0	2.78	...	0.010347	0.021174	0.
2036	0	657	22.07.2020-17:07:29	4	0	65534	182.1873	1	0	2.78	...	0.010368	0.021217	0.
2037	0	658	22.07.2020-17:07:30	4	0	65534	182.1873	1	0	2.78	...	0.010389	0.021259	0.
2038	0	659	22.07.2020-17:07:31	4	0	65534	182.1873	1	0	2.78	...	0.010410	0.021280	0.
2039	0	660	22.07.2020-17:07:31	4	0	65534	182.1873	1	0	2.78	...	0.010411	0.021323	0.

2040 rows × 45 columns

In [31]:

```
y = pd.get_dummies(SUDS["Function"])
```

```
y
```

Out[31]:

	0	1
0	0	1
1	0	1
2	0	1
3	0	1
4	0	1
...	...	...
2035	1	0
2036	1	0
2037	1	0
2038	1	0
2039	1	0

2040 rows × 2 columns

This represents the "y" values wherein 1 represents the clean samples.

In [32]:

```
x = SUDS.drop(["Function","Sample num","Time"],axis=1)
x
```

Out[32]:

	SMUX config	ATIME	ASTEP	T_INT [ms]	WTIME_EN	WTIME	WTIME [ms]	AGAIN	AGAIN [x]	LED_EN	...	Corr F1 (410nm)	Corr F2 (440nm)	Corr F (470nm
0	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.035662	0.073566	0.10690
1	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.034992	0.072179	0.10497
2	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.035909	0.073981	0.10759
3	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.036507	0.075237	0.10940
4	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.036095	0.074362	0.10816
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2035	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.010347	0.021174	0.03131
2036	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.010368	0.021217	0.03139
2037	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.010389	0.021259	0.03142
2038	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.010410	0.021280	0.03148
2039	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.010411	0.021323	0.03148

2040 rows × 42 columns

This represents the "x" values.

In [33]:

```
# split dataset
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [34]:

```
x_train
```

Out[34]:

	SMUX config	ATIME	ASTEP	T_INT [ms]	WTIME_EN	WTIME	WTIME [ms]	AGAIN	AGAIN [x]	LED_EN	...	Corr F1 (410nm)	Corr F2 (440nm)	Corr F (470nm
1060	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.035767	0.073438	0.10754
917	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.039208	0.081349	0.11784
756	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.036428	0.075562	0.10773
1553	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.036165	0.075126	0.10707
417	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.031950	0.065555	0.09680
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
249	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.034228	0.070863	0.10185
127	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.025656	0.053516	0.07627
94	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.033917	0.070464	0.09993
1225	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.041852	0.086093	0.12635
424	4	0	65534	182.1873	1	0	2.78	9	256	1	...	0.010146	0.020696	0.03054

1428 rows × 42 columns

In [35]:

```
y_train
```

Out[35]:

	0	1
1060	0	1
917	0	1
756	0	1
1553	1	0
417	1	0
...	...	...
249	1	0
127	1	0
94	1	0
1225	0	1
424	1	0

1428 rows × 2 columns

In [46]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical

# Convert y_train to one-hot encoded format
y_train_encoded = to_categorical(y_train, num_classes=3)

y_train_encoded = y_train_encoded[:, 0, :]

input_shape = x_train.shape[1]
```

```
# Define your model
model = Sequential()
model.add(Dense(6, activation="sigmoid", input_shape=(input_shape,)))  # Hidden layer
model.add(Dense(3, activation="softmax"))  # Output layer

# Compile the model
model.compile(loss="categorical_crossentropy", metrics=["accuracy"])

model.fit(x_train, y_train_encoded, epochs=25, batch_size=5)
```

```
Epoch 1/25
286/286 [==============================] - 1s 2ms/step - loss: 0.8917 - accuracy: 0.5042
Epoch 2/25
286/286 [==============================] - 1s 2ms/step - loss: 0.7237 - accuracy: 0.5889
Epoch 3/25
286/286 [==============================] - 1s 2ms/step - loss: 0.6502 - accuracy: 0.7570
Epoch 4/25
286/286 [==============================] - 1s 2ms/step - loss: 0.6157 - accuracy: 0.7346
Epoch 5/25
286/286 [==============================] - 1s 3ms/step - loss: 0.5770 - accuracy: 0.7570
Epoch 6/25
286/286 [==============================] - 1s 3ms/step - loss: 0.5441 - accuracy: 0.7906
Epoch 7/25
286/286 [==============================] - 1s 3ms/step - loss: 0.5332 - accuracy: 0.7829
Epoch 8/25
286/286 [==============================] - 1s 4ms/step - loss: 0.5137 - accuracy: 0.7955
Epoch 9/25
286/286 [==============================] - 1s 5ms/step - loss: 0.5214 - accuracy: 0.7738
Epoch 10/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5316 - accuracy: 0.7465
Epoch 11/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5090 - accuracy: 0.7773
Epoch 12/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5133 - accuracy: 0.7626
Epoch 13/25
286/286 [==============================] - 0s 2ms/step - loss: 0.5156 - accuracy: 0.7514
Epoch 14/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5077 - accuracy: 0.7605
Epoch 15/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5097 - accuracy: 0.7598
Epoch 16/25
286/286 [==============================] - 1s 2ms/step - loss: 0.4960 - accuracy: 0.7766
Epoch 17/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5144 - accuracy: 0.7710
Epoch 18/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5072 - accuracy: 0.7696
Epoch 19/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5021 - accuracy: 0.7577
Epoch 20/25
286/286 [==============================] - 0s 2ms/step - loss: 0.5068 - accuracy: 0.7654
Epoch 21/25
286/286 [==============================] - 1s 2ms/step - loss: 0.4956 - accuracy: 0.7878
Epoch 22/25
286/286 [==============================] - 1s 2ms/step - loss: 0.5320 - accuracy: 0.7514
Epoch 23/25
286/286 [==============================] - 1s 3ms/step - loss: 0.4954 - accuracy: 0.7843
Epoch 24/25
286/286 [==============================] - 1s 4ms/step - loss: 0.4845 - accuracy: 0.7899
Epoch 25/25
286/286 [==============================] - 1s 3ms/step - loss: 0.5182 - accuracy: 0.7661
```

Out[46]:

```
<keras.src.callbacks.History at 0x7d35335fd5a0>
```

OBSERVATION: As the iteration goes forward, the loss function decreases while the accuracy increases.

In [48]:

```
from keras.utils import to_categorical
y_test_encoded = to_categorical(y_test, num_classes=3)
```

```
y_test_encoded = y_test_encoded[:, 0, :]

score = model.evaluate(x_test, y_test_encoded)
print("ACCURACY:", score)
```

```
20/20 [==============================] - 0s 5ms/step - loss: 0.5350 - accuracy: 0.7516
ACCURACY: [0.535005509853363, 0.7516340017318726]
```

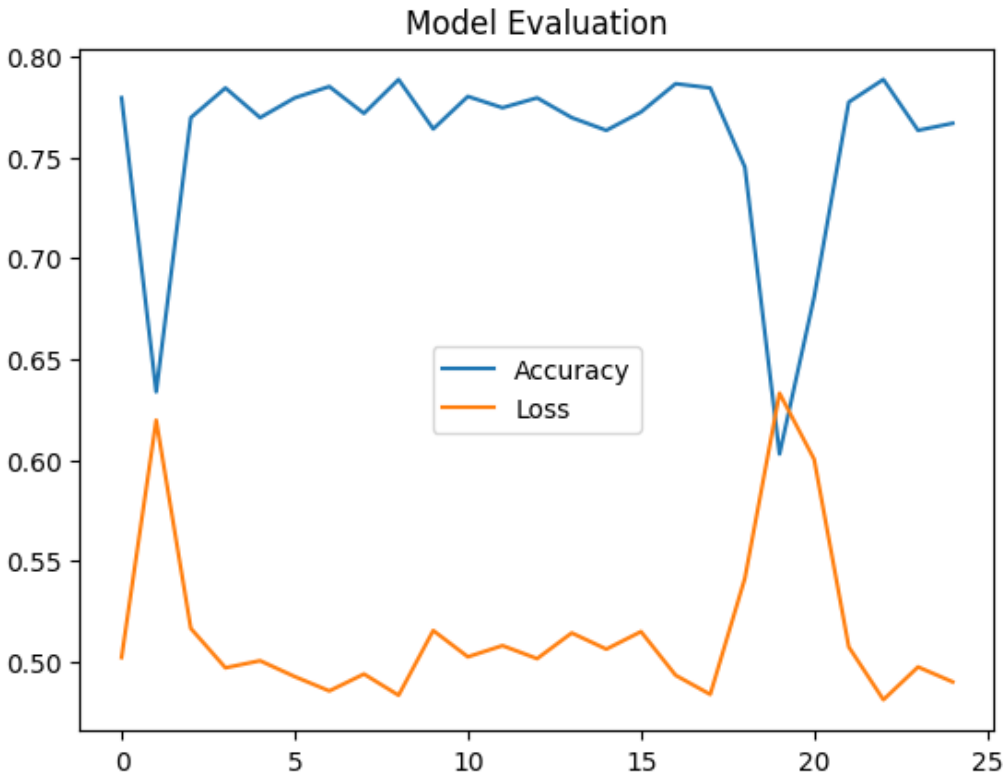
In [ ]:

```
plot = model.fit(x_train, y_train_encoded, epochs=25, batch_size=5)
```

In [56]:

```
import matplotlib.pyplot as plt

plt.plot(plot.history['accuracy'], label = 'Accuracy')
plt.plot(plot.history['loss'], label = 'Loss')
plt.title('Model Evaluation')
plt.legend()
plt.show()
```



CONCLUSION/EVALUATION:
After performing MLP in the SUDS dataset that predicts whether a water sample is soapy or clean, the accuracy score of the model is 0.7516, or 75.16%, and upon looking at the graph, we can see that as the accuracy increases, the loss also decreases.