นางสาวฉันท์สินี เมืองหนู 63070501221

Homework2 – Where did this glass come from?

- **Task:** Classify the type of glass using a fully-connected neural network.
- **Dataset:**
  - The dataset has 6 types of glass, which contain different ingredient. There are 9 types of ingredients. And note that the type of glass was labeled 1,2,3,5,6,7 (Samples type 4 are missing).
  - There are 214 samples in the dataset, in which the number of classes does not balance.
  - Features:

    1. Id number: 1 to 214
    2. RI: refractive index
    3. Na: Sodium (unit measurement: weight in corresponding oxide, as are attributes 4-10)
    4. Mg: Magnesium
    5. Al: Aluminum
    6. Si: Silicon
    7. K: Potassium
    8. Ca: Calcium
    9. Ba: Barium
    10. Fe: Iron
    11. Type of glass: (class attribute)
        -- 1 building_windows_float_processed
        -- 2 building_windows_non_float_processed
        -- 3 vehicle_windows_float_processed
        -- 4 vehicle_windows_non_float_processed (none in this database)
        -- 5 containers
        -- 6 tableware
        -- 7 headlamps

นางสาวฉันท์สินี เมืองหนู 63070501221

การสร้าง model NN จำแนก Glass Type

Step1: Import และตรวจสอบข้อมูลเบื้องต้น

- มี target คือคอลัมน์ Glass Type (ชนิดตัวเลข) → Classes are not balance
- ไม่มีข้อมูลสูญหาย (Missing Data)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('glass.csv')
```

```
df
```

Id	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe	Glass Type
1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0	1
2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0	1
3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0	1
4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0	1
5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0	1
...	...	...	...	...	...	...	...	...	...	...
210	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0	7
211	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0	7
212	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0	7
213	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0	7
214	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0	7

214 rows × 10 columns

```
df.isnull().sum()
```

```
RI            0
Na            0
Mg            0
Al            0
Si            0
K             0
Ca            0
Ba            0
Fe            0
Glass Type    0
dtype: int64
```

```
df['Glass Type'].value_counts()
```

```
2    76
1    70
7    29
3    17
5    13
6     9
Name: Glass Type, dtype: int64
```

→ class is not balance

Step2: Preparing Data

- ข้อมูลประกอบไปด้วย Feature ทั้งหมด 9 Features (ในที่นี้ทำการเปลี่ยนคอลัมน์ Id ซึ่งไม่ใช่ข้อมูล Feature และทำการ Drop คอลัมน์ Glass Type ออก เพราะเป็น Target เพื่อเตรียมค่า Feature สำหรับเทรนโมเดล)
- มี Class ของ Glass Type ทั้งหมด 6 ชนิด คือ 1, 2, 3, 5, 6, 7 ทำให้อาจสับสน เช่น ถ้าทำการ Classification แล้วได้ผลลัพธ์เป็น Class 3 นั่นหมายถึง Glass Type คือ 5 (จะอิงตามการ encoding เป็นเลข 0, 1, 2, 3, …)

```
df.set_index('Id', inplace=True)
```

```
X = df.drop('Glass Type', axis=1)
y = df['Glass Type']
```

```
X
```

Id	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
1	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0
2	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0
3	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0
4	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0
5	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0
...	...	...	...	...	...	...	...	...	...
210	1.51623	14.14	0.00	2.88	72.61	0.08	9.18	1.06	0.0
211	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0
212	1.52065	14.36	0.00	2.02	73.42	0.00	8.44	1.64	0.0
213	1.51651	14.38	0.00	1.94	73.61	0.00	8.48	1.57	0.0
214	1.51711	14.23	0.00	2.08	73.36	0.00	8.62	1.67	0.0

214 rows × 9 columns

```
y
```

```
Id
1      1
2      1
3      1
4      1
5      1
      ..
210    7
211    7
212    7
213    7
214    7
Name: Glass Type, Length: 214, dtype: int64
```

ดังนั้นในที่นี้เราจะทำการเตรียมข้อมูลซึ่งประกอบไปด้วย 4 ขั้นตอนหลัก ๆ คือ

1. Encoding
2. Perform class balancing
3. Perform feature scaling
4. Converted target output to one-hot of six types

1) Encoding

ในที่นี้จะทำให้ค่าคลาสหรือ Target (y) เป็นตัวเลขที่เรียงกันตั้งแต่ 0,1,2,... เพื่อให้ง่ายต่อการทำความเข้าใจสำหรับผลลัพธ์ในการทำ Classification แต่จากการตรวจสอบก่อนหน้านี้ พบว่าคอลัมน์ Glass Type (Target) เป็นชนิดตัวเลขแล้วแต่ไม่ได้เป็นเลขที่เรียงกัน (1 2 3 5 6) เราจึงทำการเปลี่ยนหรือแทนที่ตัวเลขเหล่านั้นให้กลายเป็นตัวเลขที่เรียงกันตั้งแต่ 0 1 2 3 4 5 แล้วเก็บไว้ที่ตัวแปร y

```
pd.unique(df['Glass Type'])
```

```
array([1, 2, 3, 5, 6, 7])
```

```
y, class_names = pd.factorize(df['Glass Type'],sort=True) #ทำการเปลี่ยน 1,2,3,5,6,7 --> 0,1,2,3,4,5
```

```
class_names
```

```
Int64Index([1, 2, 3, 5, 6, 7], dtype='int64')
```

```
y #ตรวจสอบข้อมูล target ทั้งหมด
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5])
```

```
pd.unique(y) #ตรวจสอบค่า y (target)
```

```
array([0, 1, 2, 3, 4, 5])
```

2) Perform Class Balancing

จาก Step1 ที่เราทำการตรวจสอบข้อมูลจะพบว่า Classes is not Balance ซึ่งเป็นปัญหามากกับการทำ Classification เพราะว่าสิ่งที่จะเกิดขึ้นก็คือว่า ถ้าจำนวน Class ที่เราจะจำแนกมีไม่เท่ากันจะทำให้ความแม่นยำตอนที่เราจะจำแนกมีค่าคลาดเคลื่อนไป ไม่น่าเชื่อถือ ในที่นี้จึงทำ Class Balancing โดยใช้วิธี Over-sampling ซึ่งเป็นการสุ่มเพิ่มจำนวนข้อมูลกลุ่มน้อย ให้มีจำนวนเท่ากับข้อมูลกลุ่มหลัก (แต่ก็อาจเสี่ยงต่อการเกิด Over-fitting) ซึ่งมีขั้นตอนการทำ ดังนี้

```
from imblearn import under_sampling, over_sampling
from collections import Counter
from imblearn.over_sampling import RandomOverSampler
```

```
ros = RandomOverSampler(random_state=0)
X_resampled, y_resampled = ros.fit_resample(X, y)

print(sorted(Counter(y_resampled).items()),y_resampled.shape)
```

```
[(0, 76), (1, 76), (2, 76), (3, 76), (4, 76), (5, 76)] (456,)
```

จำนวน rows

จำนวนข้อมูลเดิม ; ของแต่ละ class

[(0, 70), (1, 76), (2, 17), (3, 13), (4, 9), (5, 29)] (214)

max

class1 class2 class3 class5 class6 class7

⇒ Over-sampling → เพิ่มจำนวนข้อมูลของแต่ละ class ให้เท่ากับ class ที่มีจำนวนข้อมูลมากที่สุด " class2 "

เมื่อตรวจสอบข้อมูลหลังทำ Over-sampling แล้วพบว่าข้อมูลทั้งหมดจะมี 456 rows (จากเดิม 214 rows)

```
X = X_resampled
```

```
X
```

	RI	Na	Mg	Al	Si	K	Ca	Ba	Fe
0	1.52101	13.64	4.49	1.10	71.78	0.06	8.75	0.00	0.0
1	1.51761	13.89	3.60	1.36	72.73	0.48	7.83	0.00	0.0
2	1.51618	13.53	3.55	1.54	72.99	0.39	7.78	0.00	0.0
3	1.51766	13.21	3.69	1.29	72.61	0.57	8.22	0.00	0.0
4	1.51742	13.27	3.62	1.24	73.08	0.55	8.07	0.00	0.0
...	...	...	...	...	...	...	...	...	...
451	1.51732	14.95	0.00	1.80	72.99	0.00	8.61	1.55	0.0
452	1.52365	15.79	1.83	1.31	70.43	0.31	8.61	1.68	0.0
453	1.51653	11.95	0.00	1.19	75.18	2.70	8.93	0.00	0.0
454	1.51685	14.92	0.00	1.99	73.06	0.00	8.40	1.59	0.0
455	1.52247	14.86	2.20	2.06	70.26	0.76	9.76	0.00	0.0

456 rows × 9 columns

```
y = y_resampled
```

```
y
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 0, 0, 0, 0, 0,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
       3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
       4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
       5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5])
```

3) Perform feature scaling

เนื่องจากชุดข้อมูลมี Feature บางตัวที่มีช่วง Scale แตกต่างกัน ซึ่งจะมีผลทำให้การคำนวณค่า Weight ไม่สมดุล ในที่นี้เราทำการปรับค่าข้อมูล (ปรับสเกลของค่า x) เป็น Standard scale โดยใช้ scikit-learn และทำการแบ่งข้อมูลสำหรับ Train และ Test โดยตัว Train ใช้ค่า X_sc (ที่ผ่าน Standard scale แล้ว) ซึ่งมีขั้นตอนดังต่อไปนี้

```
from sklearn.preprocessing import StandardScaler
```

```
sc = StandardScaler()
X_sc = sc.fit_transform(X)
```

```
#split data to train & test (ratio 75:25)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_sc, y, test_size=0.25, random_state=40)
```

4) Converted target output to one-hot of six types: การทำ Class (ตัวเลข) ให้เป็นแบบ One-hot encoding

เป็นการแปลงค่าคลาส ให้มีลักษณะเป็นค่าฐานสอง (binary) สำหรับเป็นรหัสเพื่อใช้อ้างอิงการ output ของ NN ซึ่งเมื่อตรวจสอบขนาดมิติของข้อมูล y จะมีจำนวน rows เท่าเดิม แต่จำนวนคอลัมน์กลายเป็น 6 (เทียบกับ y ก่อนหน้านี้มี 1 คอลัมน์) โดยในที่นี้จะใช้ Pandas ตาม code ต่อไปนี้

```
y_train_1h = pd.get_dummies(y_train)
y_test_1h = pd.get_dummies(y_test)
```

```
y_test_1h.head(10)
```

	0	1	2	3	4	5
0	0	0	0	1	0	0
1	1	0	0	0	0	0
2	0	0	0	1	0	0
3	1	0	0	0	0	0
4	0	0	0	0	0	1
5	1	0	0	0	0	0
6	0	1	0	0	0	0
7	1	0	0	0	0	0
8	0	0	0	0	1	0
9	0	0	1	0	0	0

```
y_train_1h.shape, y_test_1h.shape
```

```
((342, 6), (114, 6))
```

****** หมายเหตุ:** จากที่ทำการแบ่งชุดข้อมูล Train-Test เนื่องจากต้องการใช้ Test Set สำหรับทดสอบ Model ว่าเกิด Over Fitting หรือไม่ (หลังจากนี้จะทำการพลอทกราฟ Loss และกราฟ Accuracy ของ Training set เทียบกับ Test set)

Step3: สร้าง Model

3.1) สร้าง Model NN และตรวจสอบสรุปโครงสร้าง:

hidden layer จำนวน 1 layer → มี 64 nodes ใช้ activation fn. = relu
output layer → มี 6 node ใช้ activation fn. = softmax
↳เนื่องจากต้องการแยก type of glass → 6 types

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

```
model = Sequential() #เริ่มต้นสร้าง Instance

#เพิ่มชั้น layer มีจำนวน 16 Node และใช้ activation fn. 'relu'
#input_shape คือ มิติรับอินพุต ในที่นี้คือ 9 Features
model.add(Dense(16, activation='relu',input_shape=(9,)))

#เพิ่มชั้น output layer มีจำนวน 6 nodes (ใช้ 6 เนื่องจาก y_train_1h มีมิติ 6 คอลัมน์ เพื่อรองรับการจำแนก 6 class)
model.add(Dense(6, activation='softmax'))
```

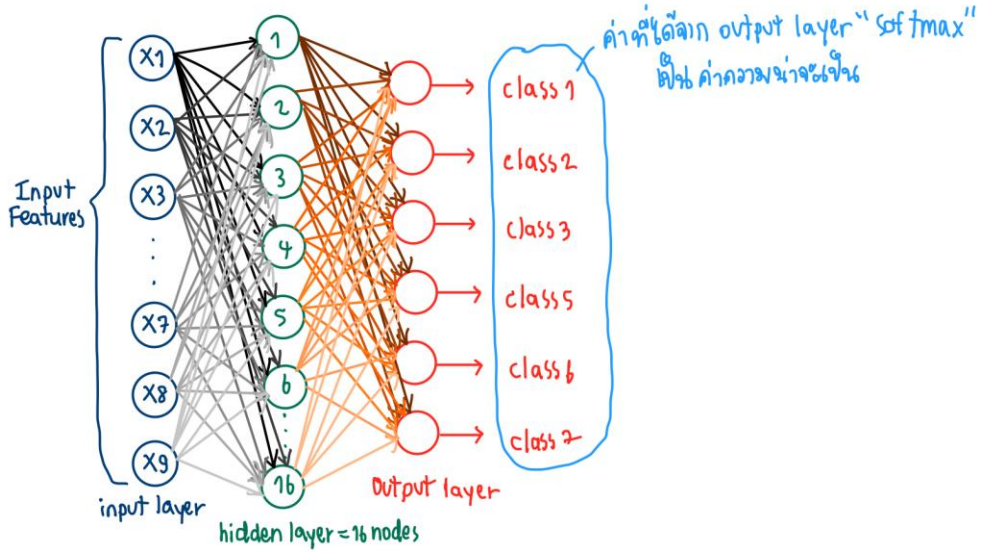
```
model.summary() #ตรวจสอบสรุปโครงสร้าง
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 16)                160

 dense_3 (Dense)             (None, 6)                 102

=================================================================
Total params: 262 (1.02 KB)
Trainable params: 262 (1.02 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

โครงสร้าง NN นี้มีชั้น Hidden Layer 1 ชั้น โดยในชั้น Hidden Layer นี้ มี 16 Nodes ส่วน Output Layer มี 6 Nodes ดังรูป



3.2) Compile:

เนื่องจาก Model นี้เป็นแบบ Multi-class classification จำแนก 6 class การ compile จะกำหนดตัววัด loss เป็น Categorical Crossentropy

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

3.3) Train **และประเมิน** Model: y ใช้ข้อมูลแบบ One-hot ในการ train

- กำหนด epochs = 600
- กำหนด batch size = 8
- กำหนด validation spit = 0.2
- ได้ Test loss = 0.3978 และ Test accuracy = 0.9123
- เมื่อ Plot กราฟ loss และ accuracy ของ training set และ test set เทียบกันพบว่า model ไม่เกิด overfitting

```
history = model.fit(X_train, y_train_1h, epochs=600, batch_size=8, verbose=1, validation_split=0.2)
```

```
Epoch 572/600
35/35 [==============================] - 0s 6ms/step - loss: 0.1185 - accuracy: 0.9560 - val_loss: 0.1922 - val_accuracy: 0.9275
Epoch 573/600
35/35 [==============================] - 0s 5ms/step - loss: 0.1210 - accuracy: 0.9524 - val_loss: 0.1984 - val_accuracy: 0.9275
Epoch 574/600
35/35 [==============================] - 0s 5ms/step - loss: 0.1188 - accuracy: 0.9524 - val_loss: 0.1851 - val_accuracy: 0.9275
Epoch 575/600
35/35 [==============================] - 0s 5ms/step - loss: 0.1210 - accuracy: 0.9597 - val_loss: 0.1795 - val_accuracy: 0.9275
Epoch 576/600
35/35 [==============================] - 0s 5ms/step - loss: 0.1189 - accuracy: 0.9487 - val_loss: 0.1852 - val_accuracy: 0.9275
Epoch 577/600
35/35 [==============================] - 0s 5ms/step - loss: 0.1181 - accuracy: 0.9560 - val_loss: 0.1919 - val_accuracy: 0.9275
```

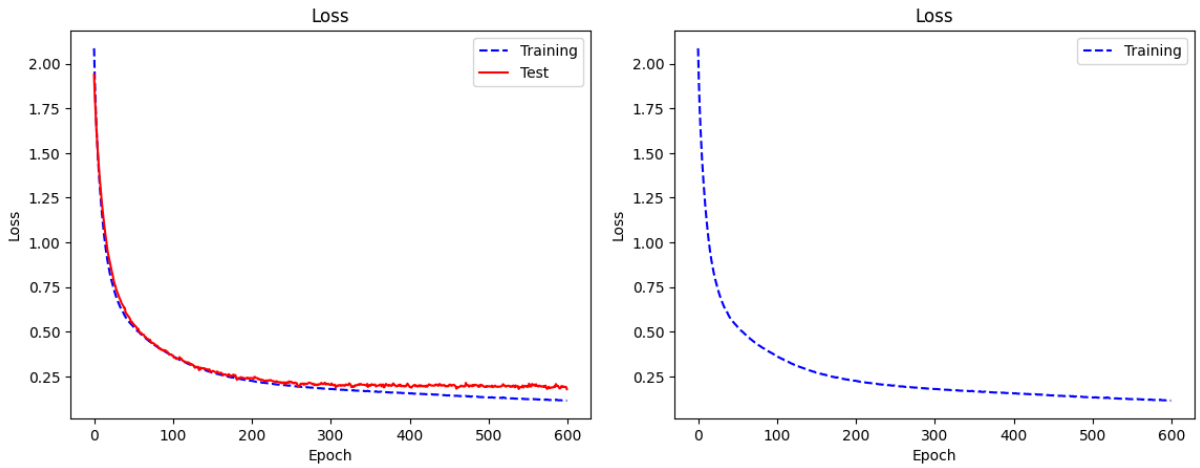
.....

```
Epoch 598/600
35/35 [==============================] - 0s 9ms/step - loss: 0.1151 - accuracy: 0.9560 - val_loss: 0.1923 - val_accuracy: 0.9275
Epoch 599/600
35/35 [==============================] - 0s 7ms/step - loss: 0.1148 - accuracy: 0.9597 - val_loss: 0.1903 - val_accuracy: 0.9275
Epoch 600/600
35/35 [==============================] - 0s 7ms/step - loss: 0.1158 - accuracy: 0.9524 - val_loss: 0.1785 - val_accuracy: 0.9275
```

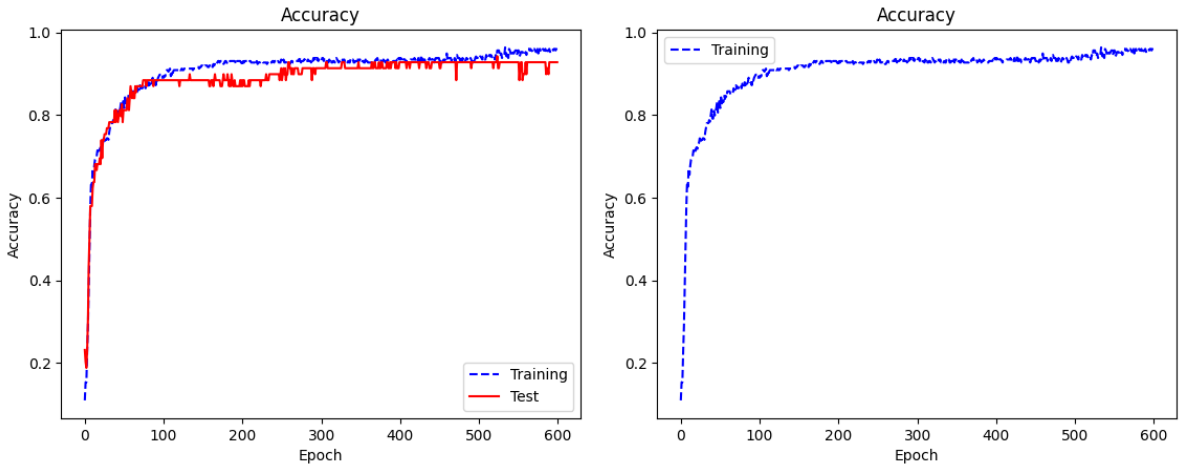
```
score = model.evaluate(X_test, y_test_1h, verbose=0)
print("Test loss: ", score[0])
print("Test accuracy: ", score[1])
```

```
Test loss:  0.39783188700675964
Test accuracy:  0.9122806787490845
```

```
df_hist = pd.DataFrame.from_dict(history.history)
df_hist['loss'].plot(style='b--', label='Training')
df_hist['val_loss'].plot(style='r-', label='Test')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.title('Loss')
plt.show()
```



```
df_hist['accuracy'].plot(style='b--', label='Training')
df_hist['val_accuracy'].plot(style='r-', label='Test')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Accuracy')
plt.show()
```



จากกราฟ Loss และ Accuracy ข้างต้น ที่พล็อตเทียบ training set กับ test set เนื่องจากต้องการดูความเหมาะสมของการปรับค่าต่าง ๆ ใน Model ว่าทำให้ Model เกิด Over fitting หรือไม่

ซึ่งพบว่า Model ค่อนข้างดี และจากการทดลองพบว่าจำนวน Epochs นั้นมีผลต่อความแม่นยำของ Model โดยถ้าหากตั้งค่าไว้น้อยเกินไปก็อาจได้โมเดลที่มีประสิทธิภาพไม่ดีพอ แต่ถ้ามากไปก็อาจเกิด overfitting ซึ่งในที่นี้เราได้ทำการ Train Model เพียง 600 Epochs ซึ่งในที่นี้เราถือว่าเป็นจำนวน Epochs ที่เหมาะสมแล้ว เนื่องจากลักษณะของกราฟ Loss พล็อตที่ได้ เส้น Train จะอยู่ล่าง Validation เพราะการ Train ย่อมได้ค่า Loss น้อยกว่า Validation อยู่แล้ว ลักษณะของเส้น Loss จะลดลงชันช่วงแรก และเริ่มขนานไปกับแกน x แสดงว่า loss เริ่มคงที่ และจากกราฟ Accuracy ที่จะเห็นได้ว่าเส้นกราฟขึ้นสูงสุดและค่อนข้างคงที่แล้ว จึงถือว่าการ Train โดยใช้จำนวน Epochs = 600 เหมาะสมแล้ว และจากกราฟก็จะเห็นได้ว่า Model ไม่เกิด Over fitting

3.4) การใช้งาน Model เพื่อจำแนกคลาส:

- การใช้งาน Model เพื่อจำแนกคลาส ต้องนำค่าข้อมูล input ที่โจทย์กำหนด (New Data) เข้าปรับค่าสเกลให้เป็น Standard Scale ก่อน

```
X_new = [[1.52, 12.8, 1.6, 2.17, 72.2, 0.76, 9.7, 0.24, 0.5]]
X_new_sc = sc.transform(X_new)
X_new_sc
```

- ใช้ Model Predict จะได้ค่าความเชื่อมั่น จากนั้นใช้ .argmax() หาว่าอยู่ในคลาสใด สรุปว่า Model ได้จำแนกเป็น class 5

เนื่องจากเราใช้ Activation function แบบ softmax ที่ output layer จะให้ผลลัพธ์เป็นค่าความเชื่อมั่นของแต่ละ class

```
y_pred_pr = model.predict(X_new_sc)
y_pred_pr #probability value from the SoftMax
```

```
1/1 [==============================] - 0s 34ms/step
array([[7.49472965e-05, 1.13689415e-02, 1.06438318e-04, 9.88436759e-01,
        1.55922175e-09, 1.29122345e-05]], dtype=float32)
```

model มีความเชื่อมั่นสูงสุด อ่านได้ว่า class ที่ 3 → class 5

ค่าความเชื่อมั่นที่ได้จาก soft max

```
y_pred = np.argmax(y_pred_pr, axis=1)
```

#ใช้ในการตรวจสอบว่าค่าความเชื่อมั่นคลาสใดมีค่าสูงสุด

```
y_pred #ได้ผลลัพธ์เป็น 3 นั่นหมายถึงคลาส 5 (0 1 2 3 4 5 --> 1 2 3 5 6 7)
```

```
array([3])
```

```
print(class_names[y_pred]) #ได้ผลลัพธ์คือคลาส 5
```

```
Int64Index([5], dtype='int64')
```