



รายงานประจำวิชา

หลักพื้นฐานของปัญญาประดิษฐ์ (Fundamentals of Artificial Intelligence)

รหัสวิชา 01418261 หมู่เรียน 870

เรื่อง

Face Mask Detection

กลุ่ม Chill Guys

จัดทำโดย

นาย	ปิ่นณวัฒน์ นิ่งเจริญ	รหัสบัณฑิต 6630250231
นาย	พันธุ์รัช สุวรรณวัฒนะ	รหัสบัณฑิต 6630250281
นาย	วรินทร์ สายปัญญา	รหัสบัณฑิต 6630250435
นางสาว	อัมพูชนิ บุญรักษ์	รหัสบัณฑิต 6630250532
นาย	ปณณภพ มีฤทธิ์	รหัสบัณฑิต 6630250591

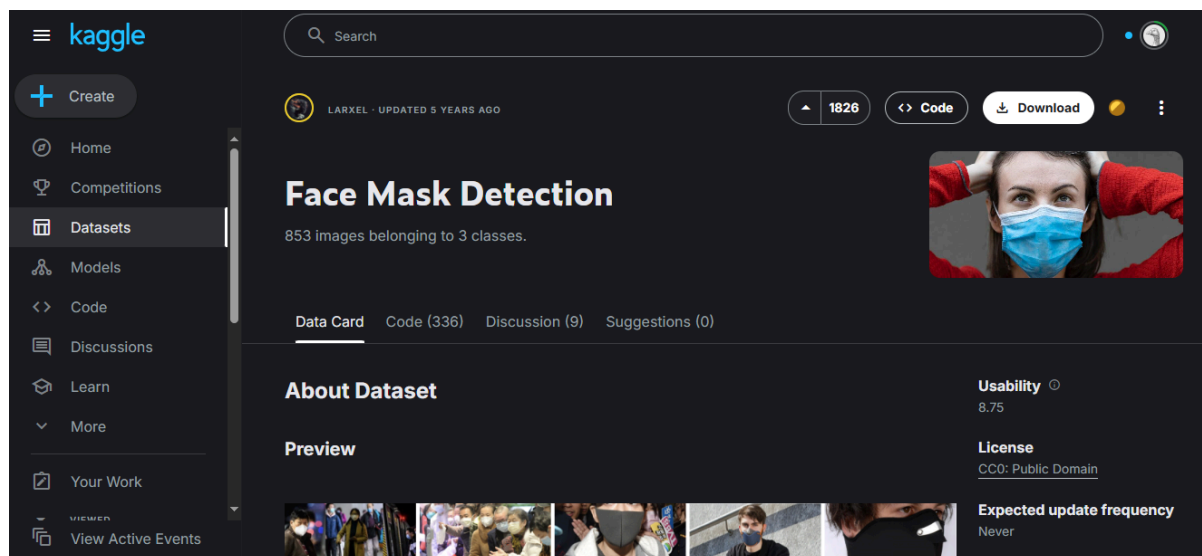
เสนอ

อาจารย์ ดร.ชโลธร ชูทอง

อาจารย์ประจำสาขาวิชาวิทยาการคอมพิวเตอร์

สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ วิทยาเขตศรีราชา

อธิบายข้อมูล Dataset



ชื่อข้อมูล dataset คือ **Face Mask Detection**

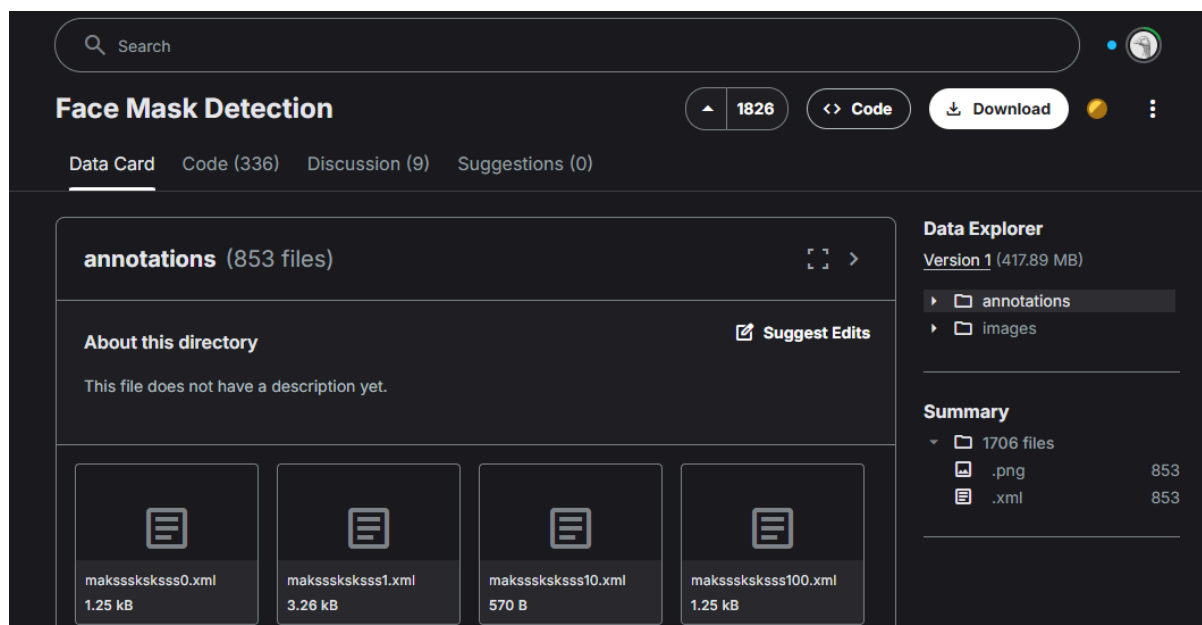
ชุดข้อมูลนี้เป็นชุดข้อมูลสำหรับการตรวจจับการสวมหน้ากากที่ใช้ในการป้องกันโรคระบาดทางเดินหายใจ เช่น COVID-19 หรือโรคอื่นๆ โดยมีรูปภาพทั้งหมด 853 ภาพ

ภาพแบ่งออกเป็น 3 ชนิด (3 classes) ได้แก่

- คนที่สวมหน้ากาก (With mask)
- คนที่ไม่สวมหน้ากาก (Without mask)
- คนที่สวมหน้ากากไม่ถูกต้อง (Mask worn incorrectly)

แต่ละภาพจะมีข้อมูล bounding boxes ที่บ่งชี้ถึงตำแหน่งของใบหน้าของคนในภาพ โดยใช้รูปแบบของ PASCAL VOC dataset ซึ่งเป็นรูปแบบที่ใช้กันอย่างแพร่หลายในการให้ข้อมูลการตรวจจับวัตถุในภาพ โดยบ่งชี้ถึงพิกเซลของภาพที่มีการตรวจจับและประเภทของวัตถุในพิกเซลนั้น ๆ ตามลำดับ

โครงสร้าง Dataset



มี 2 folders ประกอบด้วย folder

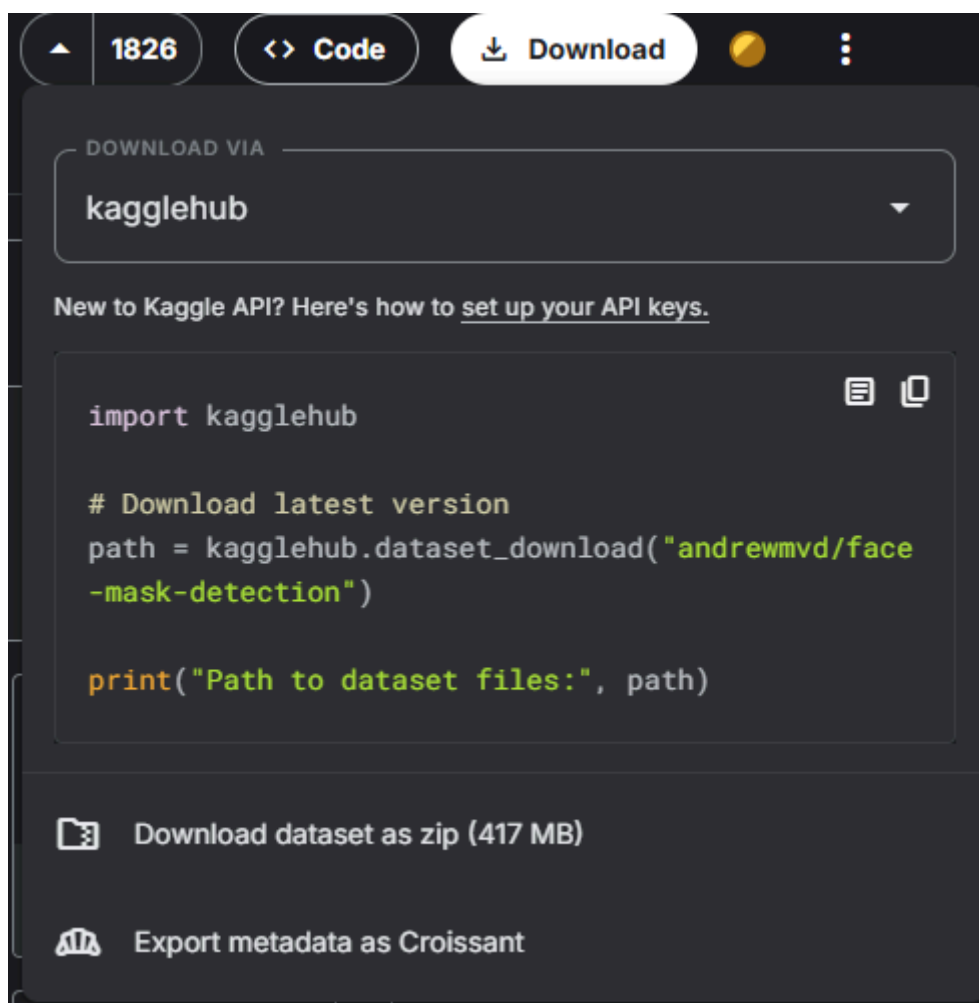
- images ที่เก็บรูปภาพข้อมูลคนใส่ mask และ ไม่ใส่ mask
- annotations ที่เก็บข้อมูลคำอธิบายของรูปภาพ

โดย folder images เก็บข้อมูลรูปภาพอยู่ที่ 853 รูปเป็นไฟล์นามสกุล .png

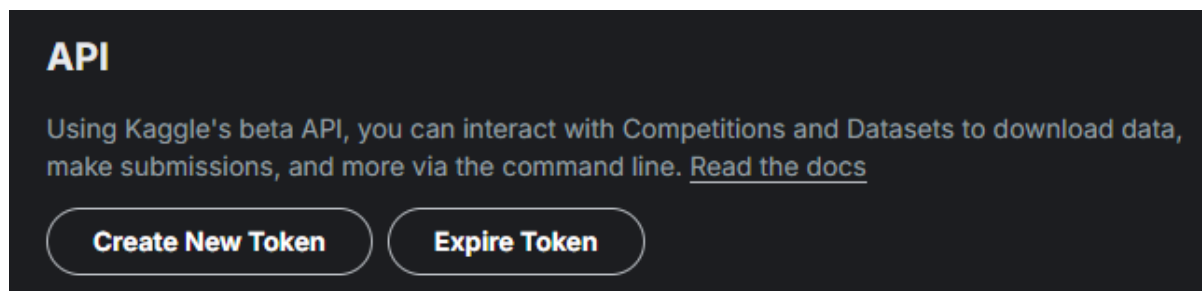
และ folder annotations เก็บข้อมูลคำอธิบายของรูปภาพที่ 853 ไฟล์เป็นไฟล์นามสกุล .xml

ขั้นตอนการเตรียม Dataset

ขั้นตอนการเตรียมการให้ทำการติดตั้ง library ของ kagglehub ใน Google Colab ให้เรียบร้อยแล้ว ทำตามขั้นตอนของเว็บไซต์ kaggle



จากนั้นให้ไปที่หน้า settings บัญชี kaggle ของเราทำการเปิดใช้งาน Kaggle API เพื่อจะได้ใช้ฟังก์ชันในการ download ข้อมูล datasets หลังจากเปิดใช้งานเสร็จจะได้ download ไฟล์ kaggle.json ซึ่งเป็นไฟล์ที่เก็บชื่อ username กับ api key เพื่อเอาไปกรอกในหน้า login ของ kaggle



AI Models ที่สร้าง

โมเดล AI ที่เราสร้างเป็นโมเดลที่ใช้สำหรับการจำแนก คนใส่แมส และ คนไม่ใส่แมส ตัวโมเดลที่เราสร้างมีด้วยกัน 4 โมเดล ประกอบด้วย Neural Network 1 โมเดล และ Machine Learning 3 โมเดล ได้แก่

- Neural Network Model
- KNN Model
- Decision Model
- Random Forest Model

อธิบายโค้ด

1. ติดตั้ง libraries ที่จำเป็น

```
[1] %pip install kagglehub
    %pip install pyyaml h5py
```

2. นำเข้า modules ของ libraries

```
# นำเข้า modules ของ libraries
import os
import re
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import cv2
from matplotlib.pyplot import figure
from PIL import Image
import xml.etree.ElementTree as ET
from sklearn.preprocessing import LabelEncoder, LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_score, recall_score, log_loss, accuracy_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
import tensorflow as tf
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Flatten, Dense, Dropout, Conv2D, MaxPooling2D, BatchNormalization, GlobalAveragePooling2D
from kagglehub import login, dataset_download
from pandas import DataFrame
from google.colab.patches import cv2_imshow
import multiprocessing
from functools import partial
from multiprocessing import Manager
from tqdm import tqdm
import pandas as pd
import random
from tensorflow.python.eager import profiler
```

3. แสดง Version ปัจจุบันที่กำลังใช้งานใน tensorflow

บังคับใช้ GPU

```
[3] # แสดง version ปัจจุบันที่กำลังใช้งาน tensorflow
    print(tf.__version__)
    tf.config.experimental.set_memory_growth(tf.config.list_physical_devices('GPU')[0], True)
    #บังคับใช้ GPU
    print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))
```

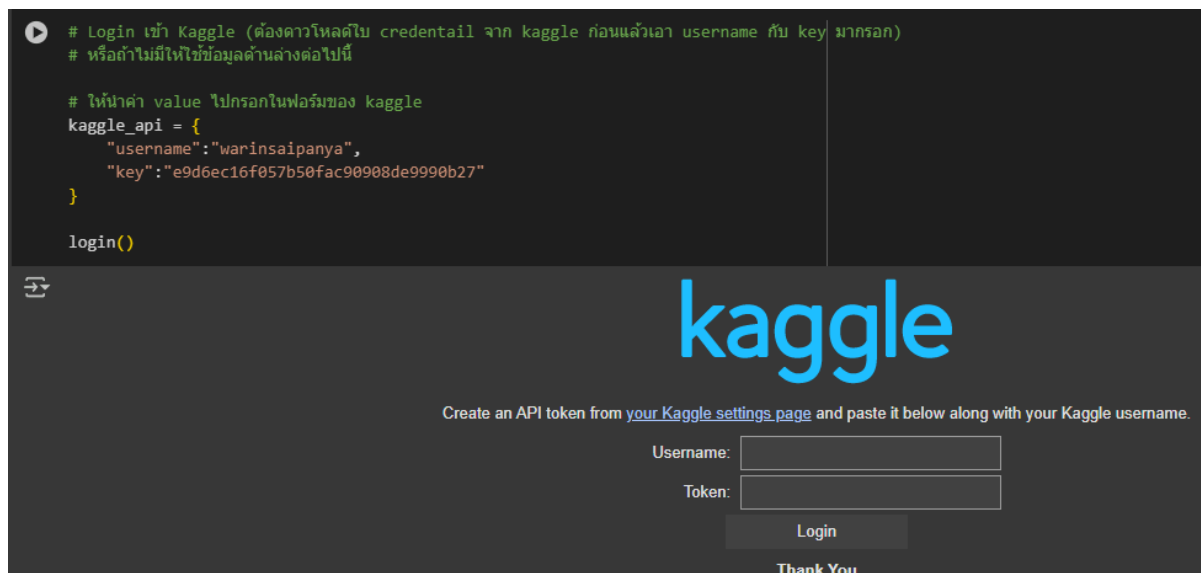


2.18.0

Num GPUs Available: 1

4. Login เข้า Kaggle (ต้องดาวโหลดใบ credentail จาก kaggle ก่อนแล้วเอา username กับ key มากรอก)
หรือถ้าไม่มีให้ใช้ข้อมูลด้านล่างต่อไปนี้

ให้นำค่า value ไปกรอกในฟอร์มของ kaggle



```
# Login เข้า Kaggle (ต้องดาวโหลดใบ credentail จาก kaggle ก่อนแล้วเอา username กับ key มากรอก)
# หรือถ้าไม่มีให้ใช้ข้อมูลด้านล่างต่อไปนี้

# ให้นำค่า value ไปกรอกในฟอร์มของ kaggle
kaggle_api = {
    "username": "warinsaipanya",
    "key": "e9d6ec16f057b50fac90908de9990b27"
}

login()
```

Username:

Token:

Login

Thank You

5. ดาวโหลดตัว dataset จาก kaggle เก็บไว้ใน path ที่ระบุ และแสดง path ที่เก็บ dataset

```
[5] # ดาวโหลดตัว dataset จาก kaggle เก็บไว้ใน path ที่ระบุ
path = dataset_download("andrewmvd/face-mask-detection")
# แสดง path ที่เก็บ dataset
print(f"path ของไฟล์ dataset อยู่ที่: {path}")
```

Downloading from https://www.kaggle.com/api/v1/datasets/download/andrewmvd/face-mask-detection?dataset_version_number=1...
100%|██████████| 398M/398M [00:09<00:00, 45.3MB/s]Extracting files...

path ของไฟล์ dataset อยู่ที่: /root/.cache/kagglehub/datasets/andrewmvd/face-mask-detection/versions/1

6. ตัวแปร classes เก็บเป็น tuple ไว้เก็บคำตอบ

```
[6] # ตัวแปร classes เก็บเป็น tuple ไว้เก็บคำตอบ
classes = ("mask_wearred_incorrect", "with_mask", "without_mask")
```

7. ชื่อ models ทั้งหมด

สร้างตัวแปรเก็บเป็น list ไว้เก็บค่าตัวเลขผลลัพธ์ของ model เพื่อนำไป plot graph และ ทำตาราง

function สำหรับเพิ่มค่า element ให้ตัวแปร list ด้านบน

```
# ชื่อ models ทั้งหมด
model_names = ("Neural Network", "KNN", "Desicion Tree", "Random Forest")

# สร้างตัวแปรเก็บเป็น list ไว้เก็บค่าตัวเลขผลลัพธ์ของ model เพื่อนำไป plot graph และ ทำตาราง
accuracy_values = []
precision_values = []
recall_values = []

# function สำหรับเพิ่มค่า element ให้ตัวแปร list ด้านบน
def add_elements(accuracy, precision, recall):
    accuracy_values.append(accuracy)
    precision_values.append(precision)
    recall_values.append(recall)
```

8. path ที่เก็บข้อมูลรูปภาพไว้ train

```
[8] # path ที่เก็บข้อมูลรูปภาพไว้ train
    folder_path = f"{path}/images"
    # path ที่เก็บข้อมูลไฟล์ xml
    xml_folder = f"{path}/annotations"
```


9. path ที่เก็บข้อมูลไฟล์ xml

```
# function สำหรับการอ่านข้อมูลในไฟล์ xml
def parse_xml(xml_path):

    tree = ET.parse(xml_path)
    root = tree.getroot()
    # เก็บข้อมูลใน tag ของ object
    objects = []

    # หา tag filename แล้วอ่านข้อมูลด้านใน content
    filename = root.find("filename").text
    # หา tag size
    size = root.find("size")
    # อ่านค่า width และ height ใน element size
    width = size.find("width").text
    height = size.find("height").text

    # วง loop หา element ที่มีชื่อว่า object และได้กลับคืนมาเป็น list แล้ววน loop
    for obj in root.findall("object"):
        # หา tag ชื่อ name แล้วอ่าน content ด้านในเก็บ class คำตอบของรูป
        label = obj.find("name").text
        # หา tag bndbox
        bndbox = obj.find("bndbox")
        # อ่านข้อมูลด้านใน element bndbox แล้วแปลงค่าเป็น int
        xmin = int(bndbox.find("xmin").text)
        ymin = int(bndbox.find("ymin").text)
        xmax = int(bndbox.find("xmax").text)
        ymax = int(bndbox.find("ymax").text)
        # เพื่อ dictionary เข้าไปใน list
        objects.append({"filename": filename, "size": { "width": width, "height": height }, "label": label, "bndbox": [xmin, ymin, xmax, ymax]})

    return objects
```

10. function สำหรับการอ่านข้อมูลในไฟล์ xml

```
[10] # ฟังก์ชัน extract_number จะพยายามค้นหาตัวเลขภายในชื่อไฟล์ หากพบตัวเลข จะส่งคืนตัวเลขนั้นเป็นจำนวนเต็ม หากไม่พบตัวเลขใดๆ จะส่งคืน -1
def extract_number(filename):
    match = re.search(r"\d+", filename)
    return int(match.group()) if match else -1
```

11. เก็บข้อมูลใน tag ของ object

```
[11] # list เก็บชื่อไฟล์ xml แบบเรียงลำดับชื่อไฟล์เป็นไฟล์ข้อมูลสำหรับแต่ละรูปภาพ
xml_files = sorted(os.listdir(xml_folder), key=extract_number)
# สร้าง object ของ label encoder ไว้ normalize ตัว label ของรูป
encoder = LabelEncoder()

# แสดงผล list ที่เก็บไฟล์ xml ไว้
print(xml_files)
```

['maksssksksss0.xml', 'maksssksksss1.xml', 'maksssksksss2.xml', 'maksssksksss3.xml',

12. อ่านไฟล์ .xml ดึงข้อมูลที่ต้องการ แล้วเก็บไว้ใน list เพื่อนำไปสร้าง DataFrame โดยตรวจสอบไฟล์, แปลง XML เป็น dictionary, และเพิ่มค่าลงใน list ตามหมวดหมู่ เช่น ชื่อไฟล์, ป้ายกำกับ, ขนาดภาพ และ พิกัดกรอบ

```
[12] # สร้างตัวแปร list ไว้เก็บข้อมูลที่ได้จากการอ่านค่าข้อมูลจากไฟล์ .xml เพื่อนำข้อมูลไปทำเป็นตาราง dataframe
filenames = []
labels = []
widths = []
heights = []
xmins = []
ymins = []
xmaxes = []
ymaxes = []

# วง loop แต่ละชื่อ xml file
for file in xml_files:

    # เอา path ของที่เก็บไฟล์ xml มา join กับชื่อไฟล์ จะได้ตำแหน่ง path ที่แท้จริงของไฟล์ xml
    file_path = os.path.join(xml_folder, file)

    # เช็คตรวจสอบไฟล์ว่ามี path ไฟล์นั้นอยู่จริงไหม หรือไฟล์นั้นไม่ใช่ไฟล์นามสกุล .xml ให้ข้าม loop นั้น
    if not os.path.isfile(file_path) or not file.lower().endswith('.xml'):
        continue

    # อ่านค่าข้อมูลไฟล์ .xml ได้เป็น list ของ dictionary หลายๆอัน
    objects = parse_xml(file_path)

    # loop ผ่าน objects ได้ค่า element ออกมาทีละรอบ
    for object in objects:
        # เพิ่มค่า elements เข้าไปใน list
        filenames.append(object["filename"])
        labels.append(object["label"])
        widths.append(object["size"]["width"])
        heights.append(object["size"]["height"])
        xmin = object["bndbox"][0]
        ymin = object["bndbox"][1]
        xmax = object["bndbox"][2]
        ymax = object["bndbox"][3]
```

13. สร้าง DataFrame จาก dictionary ของข้อมูล แล้วแสดงตัวอย่าง head(), โครงสร้าง info(), และค่าสถิติ describe()

```
# เตรียมข้อมูลเป็น dictionary แล้วส่งค่าเข้าไปใน dataframe
data = {
    "filename": filenames,
    "label": labels,
    "width": widths,
    "height": heights,
    "xmin": xmin,
    "ymin": ymin,
    "xmax": xmax,
    "ymax": ymax
}

# สร้าง object ของ dataframe แล้วส่งค่า argument ในการกำหนด index ให้แต่ละแถว
df = DataFrame(data, index=[i for i in range(len(filenames))])

# แสดงข้อมูลและรายละเอียดตาราง
print(df.head())
print(df.info())
df.describe()
```

```
filename label width height xmin ymin xmax ymax
0 maksssksksss0.png without_mask 512 366 79 105 109 142
1 maksssksksss0.png with_mask 512 366 185 100 226 144
2 maksssksksss0.png without_mask 512 366 325 90 360 141
3 maksssksksss1.png with_mask 400 156 321 34 354 69
4 maksssksksss1.png with_mask 400 156 224 38 261 73
```

<class 'pandas.core.frame.DataFrame'>

Index: 4072 entries, 0 to 4071

Data columns (total 8 columns):

#	Column	Non-Null	Count	Dtype
0	filename	4072 non-null	object	
1	label	4072 non-null	object	
2	width	4072 non-null	object	
3	height	4072 non-null	object	
4	xmin	4072 non-null	int64	
5	ymin	4072 non-null	int64	
6	xmax	4072 non-null	int64	
7	ymax	4072 non-null	int64	

dtypes: int64(4), object(4)

memory usage: 286.3+ KB

None

	xmin	ymin	xmax	ymax
count	4072.000000	4072.000000	4072.000000	4072.000000
mean	182.207024	85.780697	213.356090	120.785609
std	104.471254	52.571821	102.712267	70.355098
min	1.000000	1.000000	8.000000	6.000000
25%	96.000000	49.000000	134.000000	73.000000
50%	177.000000	75.000000	212.000000	103.000000
75%	266.000000	113.000000	292.000000	148.000000
max	569.000000	330.000000	592.000000	495.000000

14. แสดงจำนวนคำตอบของ class ทั้งหมด

```
[14] # แสดงจำนวนคำตอบของ class ทั้งหมด
# ข้อมูล dataset ทั้งหมด 4072 จำนวน
# คนใส่แมสแบ่งเป็น 3232 คน
# คนไม่ใส่แมสแบ่งเป็น 717 คน
# คนใส่แมสไม่ถูกต้องแบ่งเป็น 123 คน
for label in classes:
    print(f'{label} = {len(df[df["label"] == label])}')

mask_wearred_incorrect = 123
with_mask = 3232
without_mask = 717
```

15. โค้ดนี้ใช้ Matplotlib สร้างตารางจาก DataFrame โดยดึง 30 แถวแรก แล้วใช้ plt.table() สร้างตาราง, ปรับขนาด scale(), ปิดแกน axis('off'), และแสดงผล show()

```
# กำหนดจำนวนแถวที่จะนำไปแสดงในตาราง
n = 30
# สร้าง list เพื่อเก็บ cells ในแต่ละแถว
cell_texts = []

# วง loop ตามจำนวนรอบที่กำหนด
for i in range(n):
    # เพิ่มค่า cell ของแต่ละแถว
    cell_texts.append(df.iloc[i])

# plot สร้าง table เป็นรูปภาพ
table = plt.table(cellText=cell_texts,
                  colLabels=df.columns,
                  colWidths=[0.4 for i in range(n)],
                  loc='center',
                  colColours=['0.8' for j in range(n)],
                  cellLoc='center'
                  )

# ปรับขนาดของตารางในแนวแกน x และ y
table.scale(0.8, 2.5)

# ปิดใช้แกนของกราฟ
plt.axis('off')
# แสดงกราฟ
plt.show()
```

16. แสดงตารางรูปภาพโดยใช้ matplotlib.table ในการสร้างตาราง โดยแสดงข้อมูล 30 แถวได้แก่

ชื่อไฟล์รูปภาพ, คำตอบ (Label), ความกว้าง, ความสูง, xmin, ymin, xmax และ ymax

filename	label	width	height	xmin	ymin	xmax	ymax
makssskss0.png	without_mask	512	366	79	105	109	142
makssskss0.png	with_mask	512	366	185	100	226	144
makssskss0.png	without_mask	512	366	325	90	360	141
makssskss1.png	with_mask	400	156	321	34	354	69
makssskss1.png	with_mask	400	156	224	38	261	73
makssskss1.png	with_mask	400	156	299	58	315	81
makssskss1.png	with_mask	400	156	143	74	174	115
makssskss1.png	with_mask	400	156	74	69	95	99
makssskss1.png	with_mask	400	156	191	67	221	93
makssskss1.png	with_mask	400	156	21	73	44	93
makssskss1.png	with_mask	400	156	369	70	398	99
makssskss1.png	without_mask	400	156	83	56	111	89
makssskss2.png	with_mask	400	290	68	42	105	69
makssskss2.png	with_mask	400	290	154	47	178	74
makssskss2.png	with_mask	400	290	238	34	262	69
makssskss2.png	mask_wearred_incorrect	400	290	333	31	366	65
makssskss3.png	with_mask	400	271	52	53	73	76
makssskss3.png	with_mask	400	271	72	53	92	75
makssskss3.png	with_mask	400	271	112	51	120	68
makssskss3.png	with_mask	400	271	155	60	177	83
makssskss3.png	with_mask	400	271	189	59	210	80
makssskss3.png	with_mask	400	271	235	57	257	78
makssskss3.png	with_mask	400	271	289	60	309	83
makssskss3.png	with_mask	400	271	313	68	333	90
makssskss3.png	with_mask	400	271	351	35	364	59
makssskss4.png	with_mask	301	400	70	185	176	321
makssskss5.png	with_mask	400	266	118	54	161	96
makssskss5.png	without_mask	400	266	364	30	401	66
makssskss5.png	without_mask	400	266	192	106	229	144
makssskss5.png	with_mask	400	266	249	95	291	139

17. สร้างตัวแปรเก็บชื่อ path ของรูปภาพ และ path ของไฟล์ xml จากนั้นวน loop แต่ละชื่อไฟล์ ทำการแปลงไฟล์นามสกุล .xml เป็น .png (ตัวอย่าง example.xml เป็น example.png) เพิ่ม path รูปภาพ และ เพิ่ม path ไฟล์ xml จากนั้นทำการแสดงรูปภาพอันแรกออกมาเป็นตัวอย่าง

```
# list สำหรับเก็บ path ของไฟล์ xml และ รูปภาพ
image_paths = []
xml_paths = []

# วน loop รับ element แต่ละไฟล์
for xml_file in xml_files:
    # แทนชื่อไฟล์นามสกุล .xml เป็น .png
    image_file = xml_file.replace(".xml", ".png")
    # เอา path folder ที่เก็บรูปภาพ มา join กับชื่อไฟล์รูปภาพ
    image_path = os.path.join(folder_path, image_file)
    # เพิ่ม element ของ path รูปภาพ
    image_paths.append(image_path)

    # path ไฟล์ xml มา join กับไฟล์ xml
    xml_path = os.path.join(xml_folder, xml_file)
    # เพิ่ม element ของ path xml
    xml_paths.append(xml_path)

# แสดงรายการรูปภาพใน list อันแรก
print(image_paths[0])

# อ่านรูปภาพไฟล์อันแรก
img = cv2.imread(image_paths[0], cv2.IMREAD_UNCHANGED)
# แสดงรูปภาพ
cv2_imshow(img)
```



18. กำหนดตัวแปรเก็บค่าเลขสูงสุดสำหรับคนใส่ mask และคนไม่ใส่ mask 700 จำนวน และสร้างตัวแปร list สำหรับเก็บรูปภาพข้อมูลที่แบ่งเป็นข้อมูลแต่ละ class

```
# กำหนดค่า max ของ classes
# max_mask_wearred_incorrect = 120
max_with_mask = 700
max_without_mask = 700

# สร้าง list รวมข้อมูลของภาพแต่ละ class
x_with_masks = []
x_without_masks = []
x_mask_wearred_incorrects = []
```

19. แยกชื่อไฟล์ของภาพตามประเภท with_mask / without_mask จาก DataFrame, สุ่มลำดับ shuffle(), รวมรายการโดยลบค่าซ้ำ set(), และแสดงจำนวนไฟล์แต่ละประเภท len()

```
# สร้าง list รวมชื่อไฟล์ของคนใส่ mask
with_mask = df.loc[df['label'] == 'with_mask', 'filename'].to_list()
# สุ่มข้อมูล elements ทั้งหมดใน list
random.shuffle(with_mask)
# แสดงผล
print(f"with_mask data: {with_mask}")

# สร้าง list รวมชื่อไฟล์ของคนไม่ใส่ mask
without_mask = df.loc[df['label'] == 'without_mask', 'filename'].to_list()
# สุ่มข้อมูล elements ทั้งหมดใน list
random.shuffle(without_mask)
# สุ่มข้อมูล elements ทั้งหมดใน list
print(f"without_mask data: {without_mask}")

# wrong_mask = df.loc[df['label'] == 'mask_wearred_incorrect', 'filename'].to_list()
# random.shuffle(with_mask)
# print(f"wrong_mask data: {wrong_mask}")

# datasets_index_with_cutting = with_mask[:max_with_mask] + without_mask[:max_without_mask]
# datasets_index_with_cutting = with_mask[:] + without_mask[:] + wrong_mask[:]

# รวมไฟล์ทั้งหมดของคนใส่ mask และ ไม่ใส่ mask
datasets_index_with_cutting = with_mask[:] + without_mask[:]

# ใช้ set เพื่อไม่เอาชื่อไฟล์ที่ซ้ำกันและแปลงกับ list เหมือนเดิม
datasets_index_with_cutting = list(set(datasets_index_with_cutting))
print(f"datasets_index_with_cutting: {datasets_index_with_cutting}")

# ปริ้นจำนวนใน list ออกมา
print(f"\nmask index : {len(with_mask)}")
print(f"withoutmask index : {len(without_mask)}")
print(f"cutting index : {len(datasets_index_with_cutting)}")
```

```
with_mask data: ['makssssksksss805.png', 'makssssksksss65.png', 'makssssksksss253.png',
without_mask data: ['makssssksksss114.png', 'makssssksksss64.png', 'makssssksksss131.png',
datasets_index_with_cutting: ['makssssksksss713.png', 'makssssksksss341.png', 'makssssksksss3232
withoutmask index : 717
cutting index : 832
```


20. สร้างรูปภาพโดยการเปิดไฟล์ png เป็น xml แล้วเก็บไว้ใน ตัวแปร file สร้าง file_path มาเก็บ ตำแหน่ง path ของไฟล์ xml โดยใช้ os.path.join แล้วจะทำการตรวจสอบว่า ไฟล์นั้นมีจริงหรือไม่ แล้วทำการแปลง xml เป็น png .ใหม่อีกรอบ แล้วเอา path ที่เก็บไว้มา Join อีกครั้ง เพื่อให้ได้ path รูปภาพที่แท้จริง ถ้า path นั้น ไม่มีอยู่ จะ return ค่ากลับมา

ใช้ตัวแปร objects ในการอ่านไฟล์ xml ทำการ ดูค่า label แล้วเช็ค เงื่อนไขว่าคนนี้ใส่แมสไหม หรือ ไม่ใส่ ถ้าเป็นจริงเก็บไว้ใน List

```
In [22]: # กำหนดตำแหน่งของไฟล์สุดท้าย
file_index = -1
# กำหนดจำนวนเปอร์เซ็นต์ที่ train
train_percent = 0.8
# กำหนดขนาด batch
batch_size = 256
# กำหนดจำนวนรอบ
epochs = 50

def fetch_dataset(file_name, shared_x, shared_y):
    # รับ index จาก index_file
    # i = index_file
    # หาไฟล์จาก index
    # file = xml_files[i]

    # แปลงชื่อไฟล์นามสกุล .png เป็น .xml
    file_name = file_name.replace(".png", ".xml")
    # เก็บค่าลงตัวแปร file
    file = file_name
    # เอา path ของที่เก็บไฟล์ xml มา join กับชื่อไฟล์ จะได้ตำแหน่ง path ที่แท้จริงของไฟล์ xml
    file_path = os.path.join(xml_folder, file)

    # เช็คตรวจสอบไฟล์ว่ามี path ไฟล์นั้นอยู่จริงไหม หรือไฟล์นั้นไม่ใช่ไฟล์ นามสกุล .xml ให้ข้าม Loop นั้น
    if not os.path.isfile(file_path) or not file.lower().endswith('.xml'):
        return

    try:
        # แปลงไฟล์ นามสกุลลงท้าย .xml เป็น .png ชื่อไฟล์ xml กับ ชื่อรูปภาพเหมือนกันแต่คนหลังนามสกุลไฟล์
        image_name = file.replace(".xml", ".png")
        # เอา path ที่เก็บรูปภาพมา join กับ ชื่อไฟล์รูปภาพที่ทั้งนามสกุลไฟล์ไป ได้เป็น path ของรูปภาพที่แท้จริง
        image_path = os.path.join(folder_path, image_name)

        # เช็คเงื่อนไขถ้า path ของรูปภาพไม่มีอยู่ ให้return
        if not os.path.exists(image_path):
            print(f"Image not found: {image_path}")
            return

        # อ่านรายละเอียดไฟล์ xml
        objects = parse_xml(file_path)

        # ดูค่า label
        for _object in objects:
            # รับค่า label
            label = _object["label"]

            # tqdm.write(f"Processing {label}")
            # เช็คเงื่อนไขคำตอบว่าเป็นคนใส่ mask ไหม ถ้าเป็นจริงให้เพิ่มค่าเข้าไปใน list
            if label == "with_mask":
                x_with_masks.append(_object)

            # เช็คเงื่อนไขคำตอบว่าเป็นคนไม่ใส่ mask ไหม ถ้าเป็นจริงให้เพิ่มค่าเข้าไปใน list
            elif label == "without_mask":
                x_without_masks.append(_object)
```

กำหนดการใช้งาน multiprocessing เพื่อการสร้าง data ที่เร็วขึ้น กำหนด manager ไว้เก็บตัวแปรที่ใช้กันในหลาย process

shared_x, shared_y และ เรียกใช้ฟังก์ชัน fetch_dataset แบบ parallel

```
# position จาก xml
xmins = _object['bndbox'][0]
ymins = _object['bndbox'][1]
xmaxs = _object['bndbox'][2]
ymaxs = _object['bndbox'][3]

# ตัดภาพเฉพาะส่วนหน้า
img = img[ymins:ymaxs, xmin:xmaxs]

# ปรับขนาดของรูปภาพที่ size หลายขนาดให้เป็นขนาด 128 x 128 (height = width)
img = cv2.resize(img, (128,128))

# แสดงรูปภาพออกมา
# cv2.imshow(img)

# เพิ่ม element เข้าไปใน list โดยให้รูปภาพปรับขนาดรูปร่างและความสูงเป็น 128 และมีช่องสีแค่ 3
reshaped_array = np.array(img).reshape(128,128,3)

# แปลง array สำหรับแสดงภาพ
# converted_array = Image.fromarray(reshaped_array)
# converted_array.show()

# ให้ตัวแปร shared_y เป็น class ของค่าคอปแอนด์รูป
shared_y.append(objects[0]["label"])
# ให้ตัวแปร shared_x เป็น array รูป
shared_x.append(reshaped_array)

except Exception as e:
    print(f"Error processing {file}: {e}")

else:
    # print(f"Warning: No objects found in {file_path}")
    return

# ทำ multiprocessing
with Manager() as manager:
    # เก็บค่า shared_x เป็น list
    shared_x = manager.list()
    # เก็บค่า shared_y เป็น list
    shared_y = manager.list()

    # จำนวนรอบรูป จากไฟล์แรกไปไฟล์สุดท้าย
    tasks = range(len(xml_files[:file_index]))

    # จำนวนรอบรูปตาม index ของ ไฟล์ ที่คัดกรอง
    fileless_index = datasets_index_with_cutting

    # สร้าง object ของ multiprocessing ชื่อ pool
    with multiprocessing.Pool(processes=min(multiprocessing.cpu_count(), len(xml_files))) as pool:
        # ทำ load_bar ชื่อ pbar
        with tqdm(total=len(fileless_index), mininterval=0.1) as pbar:
            # เรียก function fetch_dataset รับค่า shared_x, shared_y, tasks
            for _ in pool.imap_unordered(partial(fetch_dataset, shared_x=shared_x, shared_y=shared_y, fileless_index):
                # update load_bar
                pbar.update(1)

    # แปลง x,y เป็น numpy array
    x = np.array(shared_x).astype("float32")
```

ตัวแปร x,y เก็บค่า shared_x,shared_y เป็น numpy array ทำการ fit_transform y ให้อยู่ในช่วง 0-2 และสร้าง Tensorflow Dataset Pipeline จาก x,y , shuffle Dataset และแยก train_dataset,test_dataset ที่กำหนดจากbatch size ,จากข้อมูล train_dataset แยก dataset ออกมา เป็น x_train , y_train และ test_dataset แยก dataset ออกมา เป็น x_test, y_test

```

y = np.array(shared_y)

# encoded ตัวคำตอบ Labels แปลงจาก string -> int ค่าอยู่ในช่วง 0 - 2
y = encoder.fit_transform(y)

# เช็คถ้าข้อมูลนั้นไม่มีอยู่ให้โยน exception นี้ออกไป
if len(x) == 0 or len(y) == 0:
    raise ValueError("No data found! Check your dataset paths and XML annotations.")

# ทำ x,y ใน tensorDataset
datasets = tf.data.Dataset.from_tensor_slices((x, y))

# ไซส์ ของ datasets
datasets_size = len(datasets)

# กำหนดจำนวนรูปที่ใช้ train
data_train_size = np.floor(train_percent*datasets_size)

# show sample
# for imgarray,label in datasets.take(3):
#     cv2.imshow(np.array(imgarray))
#     print(classes[label])

# สลับข้อมูล
datasets = datasets.shuffle(datasets_size, reshuffle_each_iteration=False)
# train ได้ batch ตามจำนวน batch_size ตาม data_train_size
train_dataset = datasets.take(data_train_size).batch(batch_size).prefetch(tf.data.AUTOTUNE)
# test ได้ batch ตามจำนวน batch_size ตาม data_train_size ที่เหลืออยู่
test_dataset = datasets.skip(data_train_size).batch(batch_size).prefetch(tf.data.AUTOTUNE)

print(f"batched train size:{len(train_dataset)}")
print(f"batched test size:{len(test_dataset)}")

# แปลงข้อมูล train dataset เป็น numpy iterator และแยกข้อมูล features (x_train) และ Labels (y_train) ด้วย zip
x_train, y_train = zip(*train_dataset.unbatch())
# แปลงข้อมูล test dataset เป็น numpy iterator และแยกข้อมูล features (x_test) และ Labels (y_test) ด้วย zip
x_test, y_test = zip(*test_dataset.unbatch().as_numpy_iterator())

# แปลงข้อมูล train x และ y เป็น array ของ numpy
x_train = np.array(x_train)
y_train = np.array(y_train)

# แปลงข้อมูล test x และ y เป็น array ของ numpy
x_test = np.array(x_test)
y_test = np.array(y_test)

# ปรับแสดงข้อมูลรูปร่าง train และ test โดย tuple มี 4 elements เก็บค่า จำนวนรูปคนที่ crop ออกมา, กว้าง, สูง และ ช่องสี 3 สี
# ส่วน y เก็บ tuple 1 element คือคำตอบของรูปภาพ
print(f"train shape (x,y):{x_train.shape},{y_train.shape}")
print(f"test shape (x,y):{x_test.shape},{y_test.shape}")

```

รูปแสดงการโหลดข้อมูล สร้าง dataset

```

100%|██████████| 832/832 [00:58<00:00, 14.12it/s]
batched train size:13
batched test size:4
train shape (x,y):(3240, 128, 128, 3),(3240,)
test shape (x,y):(811, 128, 128, 3),(811,)

```

21. กำหนดชื่อ folder ชื่อ save สำหรับเก็บตัว model ที่ training เสร็จ แล้วเขียนเงื่อนไขเช็คถ้าไม่มี path folder save ให้ทำการสร้างโฟลเดอร์ตัวนั้น

```
Neural Network Model

[20] # เก็บชื่อ folder
save_folder = "save"
# เก็บ path ของ neural network model
save_path = "save/model.keras"

# เช็คถ้าไม่มี path ที่เก็บ model ให้สร้าง folder save ไว้ทำการเก็บ model
if not (os.path.exists(save_folder)):
    os.mkdir("save")
```

22. สร้าง Sequential Model ที่มี Convolution Layer 4 layers และ Dense 512 X 512 และ OutputLayer = 3

```
# สร้าง object ของ model
model = Sequential()

# เพิ่มแต่ละ layers ให้ model
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))


# Another convolutional block
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Another convolutional block
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Another convolutional block
model.add(Conv2D(256, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))

# Reduce dimensions
model.add(GlobalAveragePooling2D())

# model.add(Flatten(input_shape=(512, 512, 3)))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
```

 /usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107
super().__init__(activity_regularizer=activity_regularizer, **kwargs)

23. compile โมเดลที่สร้างมา กำหนดค่า arguments ใน method ตามรูปภาพ

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

24. แสดง folder ว่ามีไฟล์โมเดลใหม่ model.keras จากนั้นกำหนดค่า สถานะ training เป็น True เช็kJเงื่อนไข ว่าถ้า folder เป็น folder ว่าไม่เก็บตัว model.keras ที่สร้างมา หรือ ค่าสถานะนั้น training เป็น True ให้ทำตามเงื่อนไข statements ด้านในของ if คือให้ model ทำการ training ฝึกสอนตัว model ตามจำนวน epochs ที่ 50 รอบ หลังจาก train เสร็จให้ save ตัว model ลง folder save แล้วเปลี่ยนค่าสถานะ training เป็น False เพื่อหยุดการ train ถ้าเงื่อนไข if เป็นเท็จจะทำเงื่อนไข else คือให้โหลดตัว model.keras ที่เราทำการบันทึกไว้ใน folder save หลังจากทำเงื่อนไข if หรือ else เสร็จให้เรียกใช้ model.summary() แสดงผลสรุปตัว model Neural Network

```
# แสดงรายการ folder save ว่ามี model ที่ save อยู่ใหม่
print(os.listdir(save_folder))
print()

# set สถานะว่ากำลังรันอยู่
run_training = True

# เช็คถ้า save ยังเป็น list ว่าง (ยังไม่สร้างตัว model เสร็จ) หรือ อยู่ในสถานะรัน ให้ทำการ train model
if (os.listdir(save_folder) == [] or run_training):
    # train model 14 รอบ
    model.fit(train_dataset, epochs=epochs)
    # train เสร็จ save model ลง folder save
    model.save('save/model.keras')
    # กำหนดค่าให้เป็นเท็จ ปิดสถานะการ train
    run_training = False
else:
    # ได้ชื่อไฟล์ตัว model
    latest = os.listdir(save_folder)
    # แสดงชื่อไฟล์ model
    print(f"using: {latest}")
    # โหลด model ที่ save/model.keras
    model = tf.keras.models.load_model("save/model.keras")

# สรุปผลของตัว model
model.summary()
```

🔍 []

```
Epoch 1/50
13/13 ————— 29s 896ms/step - accuracy: 0.6383 - loss: 0.9779
Epoch 2/50
13/13 ————— 18s 133ms/step - accuracy: 0.8323 - loss: 0.4916
Epoch 3/50
13/13 ————— 2s 133ms/step - accuracy: 0.8325 - loss: 0.4625
```

25. แสดงค่าผลลัพธ์ของ accuracy และ ค่า loss ของตัว model

```
# คำนวณค่า accuracy และ ค่า loss ของ model
loss, accuracy = model.evaluate(test_dataset)

# แสดงผลลัพธ์ค่า accuracy กับ loss ออกมา
print(f"Accuracy: {accuracy}")
print(f"Loss: {loss}")
```

4/4 ————— 5s 375ms/step - accuracy: 0.7910 - loss: 0.9695
Accuracy: 0.7990135550498962
Loss: 0.9279290437698364

26. ให้ model ทำการทำนายผลลัพธ์คนที่ใส่ mask หรือ คนที่ไม่ใส่ mask แล้วเก็บลงตัวแปร y_pred นำ y_pred ส่งเข้าไปใน function precision และ recall เพื่อคำนวณค่าดังกล่าวแล้วแสดงผลออกมา

```
# ทำนายค่าผลลัพธ์
y_pred = np.argmax(model.predict(x_test), axis=-1)

# คำนวณค่า precision และ recall
precision = precision_score(y_test, y_pred, average="weighted")
recall = recall_score(y_test, y_pred, average="weighted")

# แสดงผลลัพธ์ precision กับ recall
print(f"Precision: {precision}")
print(f"Recall: {recall}")
```

26/26 ————— 2s 41ms/step
Precision: 0.7661563862122158
Recall: 0.7990135635018496

27. เพิ่มค่า accuracy, precision และ recall เก็บไว้ใน list

```
[26] # เรียกใช้ function เพิ่มค่าที่สำคัญเข้าไปใน list
add_elements(accuracy, precision, recall)
```

28. ทำการ predict test ตัว model Neural Network โดยเลือก index 52 ของรูปภาพ test ทำการอ่านรูปภาพ และคำตอบของภาพ จากนั้นทำการ predict รูปภาพ และ ค่าความมั่นใจของ model จากนั้นแสดงผลปรี้นออกมา

```
# ตำแหน่งรูปที่ใช้
img_index = 52

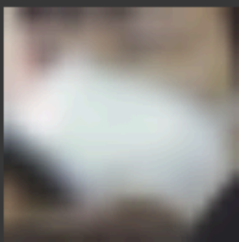
# รูปตัวอย่าง
img_array = x_test[img_index]
answer = y_test[img_index]
cv2_imshow(x_test[img_index])

# เพิ่ม dim ให้ model
img_array = np.expand_dims(img_array, axis=0)

# แสดงรูป
# img_array.show()

# ความแม่นยำ และ ความมั่นใจ
prediction = model.predict(img_array)[0]
predicted_class = np.argmax(prediction)
confidence = prediction[predicted_class]

# ปรี้นคำตอบจริงออกมา และเทียบกับค่า การทำนายของ model และค่าความมั่นใจของ model
print(f"Answer: {classes[answer]}")
print(f"Predict: {classes[predicted_class]}")
print(f"Confidence: {confidence}")
```



1/1 ————— 1s 623ms/step
 Answer: with_mask
 Predict: with_mask
 Confidence: 0.9991969466209412

29. แสดงจำนวนมิติของข้อมูล train และ test และ แสดงรูปร่างของข้อมูล train และ test

```
# แสดงจำนวนมิติของ train และ test
print(x_train.ndim)
print(x_test.ndim)
print(y_train.ndim)
print(y_test.ndim)

print()
# แสดงรูปร่างของข้อมูล train และ test
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
4
4
1
1

(3240, 128, 128, 3)
(811, 128, 128, 3)
(3240,)
(811,)
```

30. ทำการแปลงรูปร่างข้อมูลอันเก่าของ Neural Network จาก 4 มิติเป็น 2มิติ และคำตอบของรูปภาพเป็น 1 มิติ เพื่อใช้สำหรับการ train model Machine Learning จากนั้นแสดงรูปร่างของข้อมูล train และ test อันใหม่

```
[29] # แปลงจาก array 4 มิติให้เป็น array 2 มิติ
x_train2 = x_train.reshape(x_train.shape[0], -1)
x_test2 = x_test.reshape(x_test.shape[0], -1)

# แปลงเป็น array 1 มิติ
y_train2 = y_train.reshape(y_train.shape[0], -1)
y_test2 = y_test.reshape(y_test.shape[0], -1)

# แสดงรูปร่างของข้อมูล train และ test
print(f"train_flat shape (x,y): {x_train2.shape}, {y_train2.shape}")
print(f"test_flat shape (x,y): {x_test2.shape}, {y_test2.shape}")
```

```
train_flat shape (x,y): (3240, 49152), (3240, 1)
test_flat shape (x,y): (811, 49152), (811, 1)
```

31. แสดงรูปร่างข้อมูล train และ test อันใหม่

```
# แสดงรูปร่างของข้อมูล train และ test
print(x_train2.shape)
print(x_test2.shape)
print(y_train2.shape)
print(y_test2.shape)
```

```
(3240, 49152)
(811, 49152)
(3240, 1)
(811, 1)
```


33. ใช้ โมเดล KNN ทำนายคลาสของภาพตัวอย่าง จากชุดทดสอบ พร้อมแสดงผลเปรียบเทียบระหว่าง ค่าจริงกับค่าที่โมเดลทำนาย รวมถึงระดับความมั่นใจในการพยากรณ์

```

# ตำแหน่งรูปที่ใช้
img_index = 128

# รูปตัวอย่าง
img_array = x_test2[img_index]
answer = y_test[img_index]


# เพิ่ม dim ให้ model
img_array = np.expand_dims(img_array, axis=0)

# แสดงรูป
cv2_imshow(x_test[img_index])

# ความแม่นยำ และ ความมั่นใจ
predicted_class = knn.predict(img_array)[0]
y_pred_prob = knn.predict_proba(img_array)
confidence = y_pred_prob[0][predicted_class]

# ปรี้นค่าออกมาเทียบดูคำตอบจริง กับ คำตอบที่ model ทำนายออกมา
print(f"Answer: {classes[answer]}")
print(f"Predict: {classes[predicted_class]}")
print(f"Confidence: {confidence}")

```



```

Answer: without_mask
Predict: with_mask
Confidence: 0.9333333333333333

```

34. โค้ดนี้ใช้ Decision Tree Classifier เพื่อสร้างโมเดลสำหรับการจำแนกประเภทข้อมูล จากนั้นทำการ ฝึก โมเดล, ทำนายผลลัพธ์, และวัดประสิทธิภาพของโมเดล โดยใช้ตัวชี้วัดหลายค่า เช่น Accuracy, Precision, Recall และ Log Loss ซึ่งช่วยให้สามารถเปรียบเทียบโมเดลนี้กับโมเดลอื่นได้อย่างชัดเจน

▼ Desicion Tree Model

✓
4m

```
# สร้าง Decision Tree Model
decision_tree_model = DecisionTreeClassifier(random_state=42)

# ฝึก Decision Tree Model
decision_tree_model.fit(x_train2, y_train2)

# ทำนายค่าผลลัพธ์จากข้อมูลทดสอบ
dt_y_pred = decision_tree_model.predict(x_test2)

# ทำนายค่าความน่าจะเป็นจาก Decision Tree Model
dt_y_prob = decision_tree_model.predict_proba(x_test2)

# คำนวณ Log Loss
dt_loss = log_loss(y_test2, dt_y_prob, labels=[0, 1, 2])

# คำนวณ Precision, Recall สำหรับ Decision Tree
dt_precision = precision_score(y_test2, dt_y_pred, average='weighted')
dt_recall = recall_score(y_test2, dt_y_pred, average='weighted')

# คำนวณ Accuracy
dt_accuracy = decision_tree_model.score(x_test2, y_test2)

# แสดงผลลัพธ์ของ Decision Tree Model
print(f"Decision Tree Accuracy: {dt_accuracy}")
print(f"Decision Tree Precision: {dt_precision}")
print(f"Decision Tree Recall: {dt_recall}")
print(f"Decision Tree Loss: {dt_loss}")

# เพิ่มผลลัพธ์ของ Decision Tree ลงใน list
add_elements(dt_accuracy, dt_precision, dt_recall)
```

```
→ Decision Tree Accuracy: 0.7262638717632552
Decision Tree Precision: 0.7383805535872701
Decision Tree Recall: 0.7262638717632552
Decision Tree Loss: 9.866450126244153
```

35. โค้ดนี้ใช้ Decision Tree Model เพื่อทำนายคลาสของรูปภาพจากชุดข้อมูลทดสอบ (x_test2) โดยเลือกภาพที่มี ดัชนี (index) เท่ากับ 70 จากนั้นทำการแสดงภาพ ทำนายคลาส และคำนวณค่าความมั่นใจของโมเดล ก่อนพิมพ์ผลลัพธ์ออกมาเพื่อเปรียบเทียบกับค่าจริง

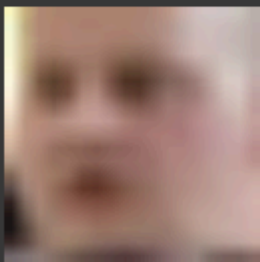
```
# ตำแหน่งรูปที่ใช้
img_index = 70
img_array = x_test2[img_index]
answer = y_test[img_index]

# เพิ่ม dim ให้ model
img_array = np.expand_dims(img_array, axis=0)

# แสดงรูป
cv2_imshow(x_test[img_index])

# ความแม่นยำ และ ความมั่นใจ
predicted_class = decision_tree_model.predict(img_array)[0]
y_pred_prob = decision_tree_model.predict_proba(img_array)
confidence = y_pred_prob[0][predicted_class]

# ปรี้นค่าออกมาเทียบดูคำตอบจริง กับ คำตอบที่ model ทำนายออกมา
print(f"Answer: {classes[answer]}")
print(f"Predict: {classes[predicted_class]}")
print(f"Confidence: {confidence}")
```



```
Answer: with_mask
Predict: with_mask
Confidence: 1.0
```

36. ทำการสร้าง model random forest คำนวณค่า precision และ ค่า recall คำนวณค่า accuracy เพิ่มค่าผลลัพธ์เข้าไปใน list และแสดงผลลัพธ์

Random Forest Model

15s

```

# สร้าง model random forest
rfc_model = RandomForestClassifier(n_estimators=100, random_state=42)

# train model
rfc_model.fit(x_test2,y_test2)

# คำนวณค่า precision และ ค่า recall
rfc_precision_score = precision_score(y_test2, rfc_model.predict(x_test2),average="weighted")
rfc_recall_score = recall_score(y_test2, rfc_model.predict(x_test2),average="weighted")

# คำนวณค่า accuracy
rfc_test_acc = accuracy_score(y_test2, rfc_model.predict(x_test2))

# loss
rfc_loss = log_loss(y_train2,rfc_model.predict_proba(x_train2))

# แสดงผลลัพธ์
print("RFC Test Accuracy:", rfc_test_acc)
print("RFC Loss:", rfc_loss)
print("RFC Precision Score:", rfc_precision_score)
print("RFC Recall Score:", rfc_recall_score)

# เพิ่มค่าผลลัพธ์เข้าไปใน list
add_elements(rfc_test_acc, rfc_precision_score, rfc_recall_score)

```

```

/usr/local/lib/python3.11/dist-packages/sklearn/base.py:1389: DataConversionWarning: A column-v
return fit_method(estimator, *args, **kwargs)
RFC Test Accuracy: 1.0
RFC Loss: 0.6788898623913587
RFC Precision Score: 1.0
RFC Recall Score: 1.0

```

37. โค้ดนี้ใช้ Random Forest Model เพื่อทำนายคลาสของรูปภาพจากชุดข้อมูลทดสอบ (x_test2) โดยเลือกภาพที่มี ดัชนี (index) เท่ากับ 104 จากนั้นทำการแสดงภาพ ทำนายคลาส และคำนวณค่าความมั่นใจของโมเดล ก่อนพิมพ์ผลลัพธ์ออกมาเพื่อเปรียบเทียบกับค่าจริง

```

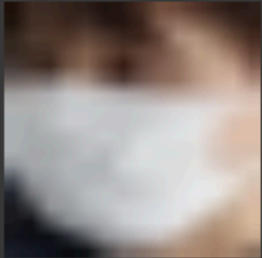
▶ #ตำแหน่งรูปที่ใช้
img_index = 104
img_array = x_test2[img_index]
answer = y_test[img_index]
cv2_imshow(x_test[img_index])

#เพิ่ม dim ให้ model
img_array = np.expand_dims(img_array, axis=0)

#ความแม่นยำ และ ความมั่นใจ
predicted_class = rfc_model.predict(img_array)[0]
y_pred_prob = rfc_model.predict_proba(img_array)
confidence = y_pred_prob[0][predicted_class]

# ปรี้นค่าออกมาเทียบดูคำตอบจริง กับ คำตอบที่ model ทำนายออกมา
print(f"Answer: {classes[answer]}")
print(f"Predict: {classes[predicted_class]}")
print(f"Confidence: {confidence}")

```



Answer: with_mask
Predict: with_mask
Confidence: 0.98

38. ปรี้นค่าผลลัพธ์ที่เก็บมาจากทั้ง 4 model

```
[37] # ปรี้นค่าผลลัพธ์ที่เก็บมา
      print(accuracy_values)
      print(precision_values)
      print(recall_values)

[0.7990135550498962, 0.8286066584463625, 0.7262638717632552, 1.0]
[0.7661563862122158, 0.7715669103960668, 0.7383805535872701, 1.0]
[0.7990135635018496, 0.8286066584463625, 0.7262638717632552, 1.0]
```

39. โค้ดนี้กำหนดค่าและเตรียมข้อมูลสำหรับการสร้างกราฟแท่ง (bar chart) ที่จะใช้แสดงผลลัพธ์ของ ค่าสถิติ ต่างๆ ที่ได้จากการทดสอบโมเดลหลายๆ ตัว (เช่น ความแม่นยำ (accuracy), Precision, Recall) โดยมี การใช้สีที่กำหนดไว้เพื่อแยกแต่ละโมเดลในกราฟให้ชัดเจน

Plot graphs

```
[38] # กำหนดค่าสีแต่ละ model ในกราฟแท่ง
      bar_colors = ['tab:blue', 'tab:red', 'tab:orange', 'tab:green']
      # สร้าง list ไว้กำหนดป้ายแต่ละป้ายที่จะแสดงในแกน y
      acc_labels = []
      pre_labels = []
      rec_labels = []
```

40. ฟังก์ชัน plot_graph สร้างกราฟแท่งโดยรับพารามิเตอร์ชื่อกราฟ, ป้ายแกน y, ค่าของแกน y, และป้ายแกน x (ค่าเริ่มต้นเป็น 'Models'). ใช้ plt.bar() เพื่อสร้างกราฟ, ปรับขนาดกราฟ, แสดงชื่อแกน, ชื่อกราฟ, และคำอธิบายของกราฟ ก่อนแสดงผลด้วย plt.show()

```
[39] # สร้าง function สำหรับ plot graph
      def plot_graph(title, y_label, y_labels, x_label = 'Models'):

          # ปรับขนาดของกราฟ
          figure(figsize=(10, 5.5))

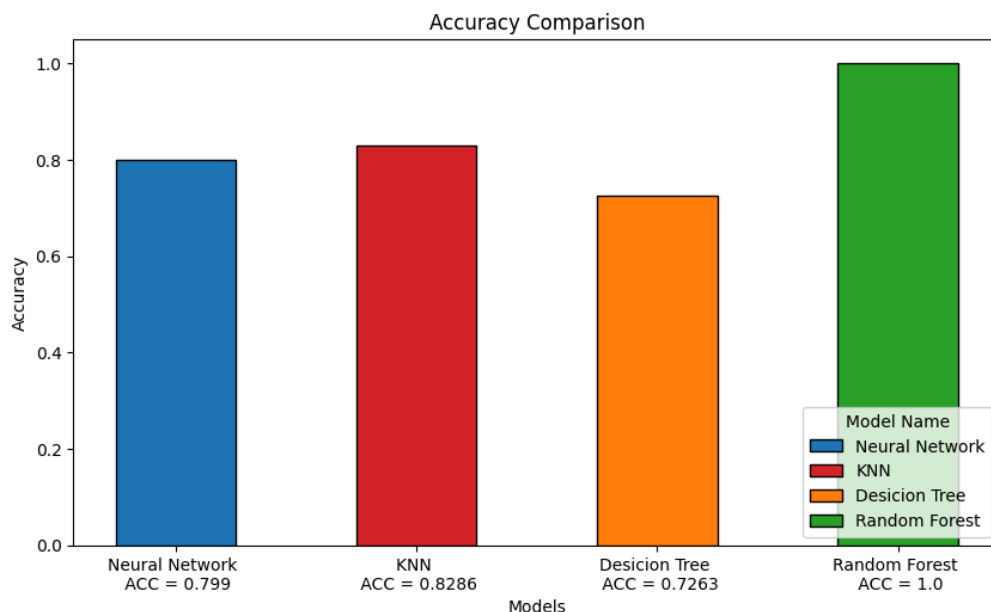
          # แสดงกราฟตามค่า arguments ที่ส่งมา
          plt.bar(y_labels, accuracy_values, width=.5, label=model_names, color=bar_colors, edgecolor='0')
          plt.xlabel(x_label)
          plt.ylabel(y_label)
          plt.title(title)
          plt.legend(title='Model Name', loc='lower right')
          plt.show()
```

41. โค้ดนี้จะเพิ่มชื่อโมเดลและค่าประสิทธิภาพ (accuracy, precision, recall) ลงใน acc_labels, pre_labels, และ rec_labels แล้วปัดค่าทศนิยม 4 ตำแหน่งสำหรับแต่ละค่า

```
# วง loop เพิ่มค่า y labels
for i, name in enumerate(model_names):
    # เพิ่ม element แล้วปรับทศนิยมของตัวเลขให้เป็น 4 ตำแหน่ง
    acc_labels.append(f'{name} \n ACC = {round(accuracy_values[i], 4)}')
    pre_labels.append(f'{name} \n P = {round(precision_values[i], 4)}')
    rec_labels.append(f'{name} \n R = {round(recall_values[i], 4)}')
```

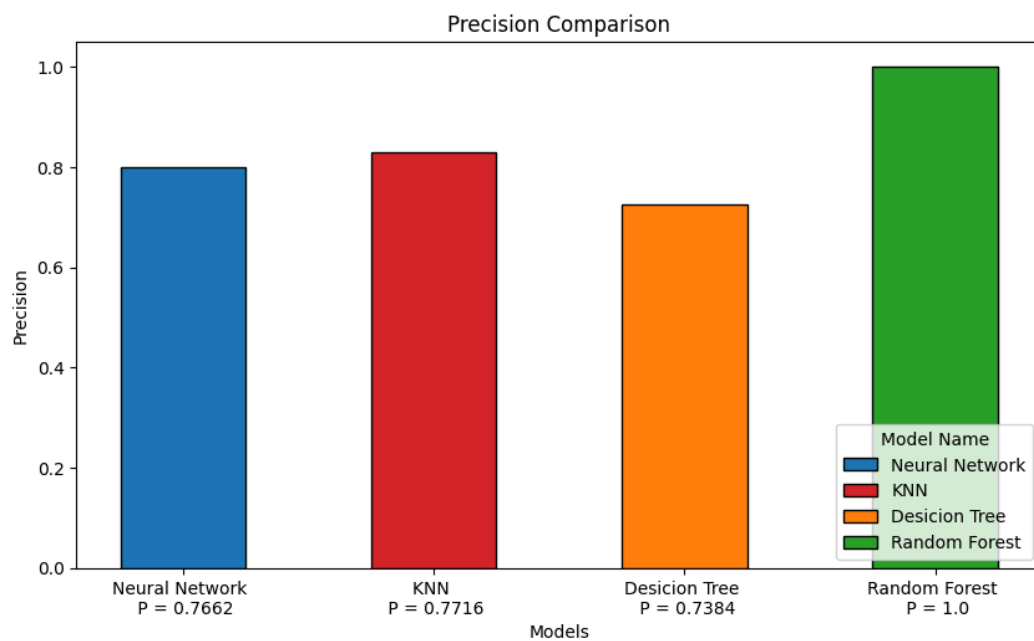
42. โค้ดนี้เป็นการสร้างกราฟเพื่อเปรียบเทียบค่า accuracy ของโมเดลต่าง ๆ โดยที่ชื่อกราฟคือ "Accuracy Comparison" และชื่อของแกน Y คือ "Accuracy" ซึ่งข้อมูลที่ใส่แสดงในกราฟจะมาจากตัวแปร acc_labels ที่ถูกส่งผ่านไปนฟังก์ชัน plot_graph()

```
# เรียกใช้ function plot graph
# สร้างกราฟเปรียบเทียบค่า accuracy
plot_graph('Accuracy Comparison', 'Accuracy', acc_labels)
```



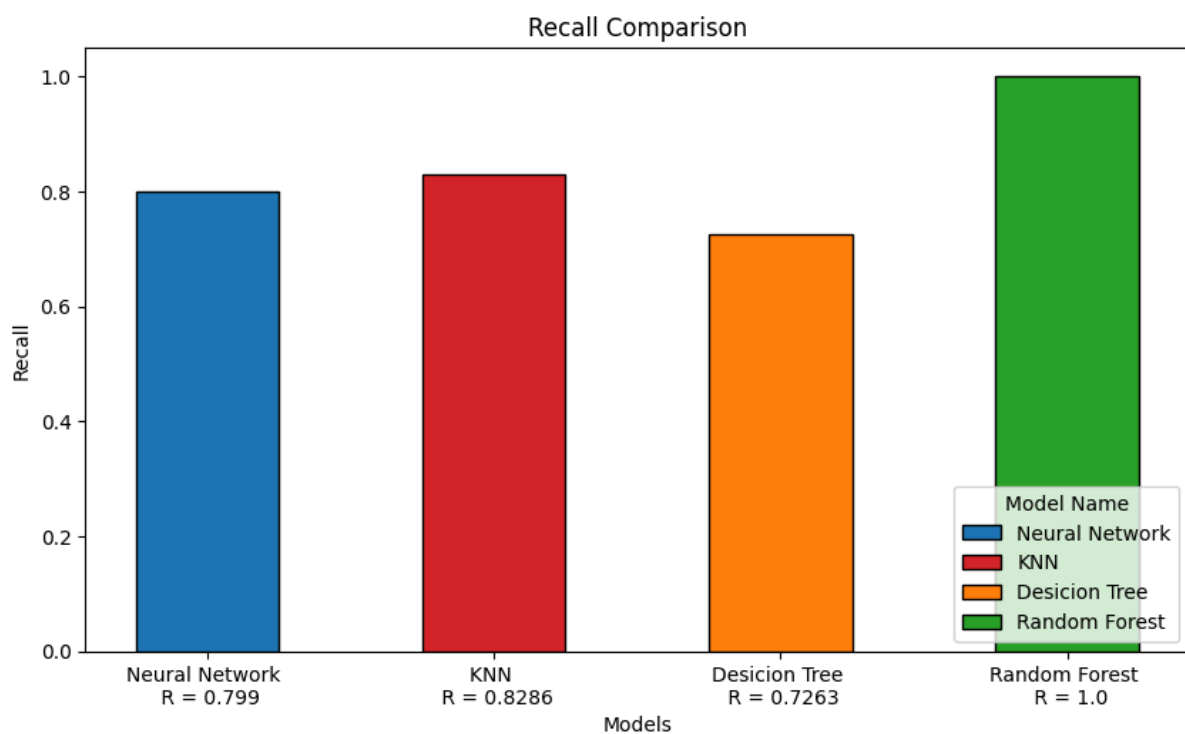
43. ใ้ค้ดนี้้จะสร้างกราฟเปรียบเทียบค่า precision ของโมเดลต่าง ๆ โดยที่ชื่อกราฟคือ Precision Comparison และชื่อของแกน Y คือ Precision โดยใช้ข้อมูลจากตัวแปร pre_labels ซึ่งเป็นค่าของ precision ที่ได้จากการประเมินผลของโมเดลต่าง ๆ

```
[42] # สร้างกราฟเปรียบเทียบค่า precision
      plot_graph('Precision Comparison', 'Precision', pre_labels)
```



44. โค้ดนี้จะสร้างกราฟเปรียบเทียบค่า recall ของโมเดลต่าง ๆ โดยที่ชื่อกราฟคือ Recall Comparison และชื่อของแกน Y คือ Recall โดยใช้ข้อมูลจากตัวแปร rec_labels ซึ่งเป็นค่าของ recall ที่ได้จากการประเมินผลของโมเดลต่าง ๆ

```
[43] # สร้างกราฟเปรียบเทียบค่า recall
      plot_graph('Recall Comparison', 'Recall', rec_labels)
```



หน้าที่ความรับผิดชอบของสมาชิกแต่ละคน

นาย ปณณวัฒน์ นิ่งเจริญ รหัสสถิติ 6630250231 (ทำ Neural Network ทำรายงาน)

นาย พันธุ์ช สุวรรณวัฒนะ รหัสสถิติ 6630250281 (ทำ Decision Tree Model และทำเอกสาร)

นาย วรินทร์ สายปัญญา รหัสสถิติ 6630250435 (เขียนตารางรูปภาพ, ทำ DataFrame, เขียนตาราง
วาดกราฟแท่งเปรียบเทียบแต่ละ models, เขียน comments โดยรวมของโค้ด, เตรียมข้อมูล datasets จาก
kagglehub, ช่วยจัดทำรายงานเอกสาร)

นางสาว อัมพูชนิ บุญรักษ์ รหัสสถิติ 6630250532 (เขียน KNN Model, Decision Tree Model และทำเอกสาร)

นาย ปุณณภพ มีฤทธิ์ รหัสสถิติ 6630250591 (แก้โค้ด Neural Network, สร้าง Dataset, random forest
classifier)

ลิงค์ Google Colab:

https://drive.google.com/file/d/1cKicMmXocJzn-DhdU2wU_2TCAJ2mQ3hj/view?usp=sharing