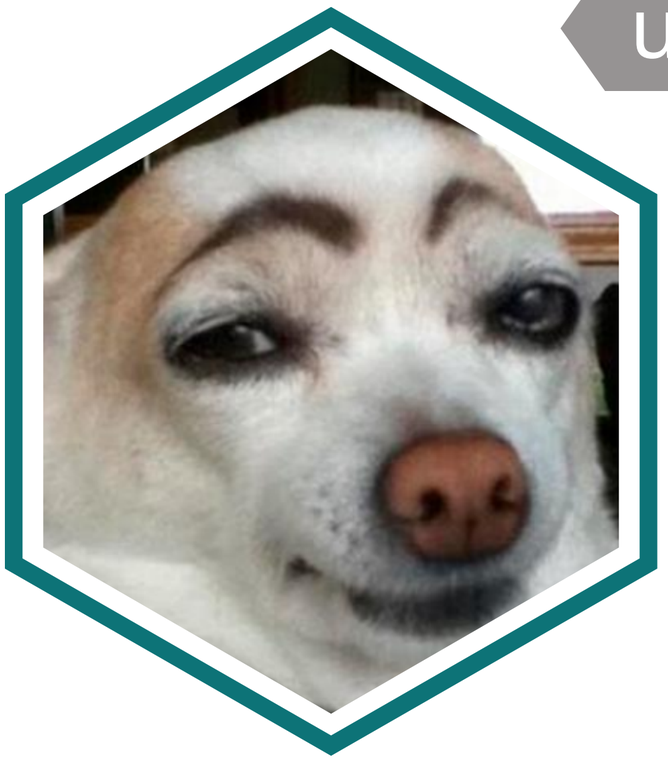
ANIMAL IMAGE CLASSIFIER

ผู้จัดทำ

นายพงศธร พันธ์ศรี เลขที่ 2

นายศรสิทธิ์ ฤทธิธา เลขที่ 6



นายกษิดิ์เดช เดชแดง เลขที่ 10

นางสาวจิรนุช บูรณ์เจริญ เลขที่ 15

นายเตชินท์ ซาไธสง เลขที่ 20

นายเกียรติศักด์ เสือใหล เลขที่ 21

บทนำ

การแยกแยะและจำแนกสัตว์ต่าง ๆ มีความสำคัญในหลายสาขาของวิทยาธรรมชาติและวิทยาศาสตร์ คอมพิวเตอร์ เช่น การศึกษาความหลากหลายของชีวิตในธรรมชาติ, การจัดการสิ่งแวดล้อม, และการเฝ้าระวัง สุขภาพสัตว์และมนุษย์ การระบุและจำแนกสัตว์ที่ถูกต้องและรวดเร็วมีความสำคัญในงานดังกล่าว แต่การทำ เช่นนี้ด้วยมือใช้เวลาและแรงงานมาก ดังนั้นเรามุ่งหวังที่จะใช้เทคโนโลยี Machine Learning เพื่อเร่ง กระบวนการนี้และทำให้ง่ายขึ้นโปรเจกต์นี้มุ่งเน้นการสร้างแบบจำแนกสัตว์โดยใช้ข้อมูลภาพ โดยการสร้างแบบ จำแนกสัตว์ด้วย Machine Learning จะช่วยลดเวลาและแรงงานที่ต้องใช้ในกระบวนการการจำแนกดังกล่าว ทั้งยังเสริมสร้างความแม่นยำในการจำแนกที่มีประสิทธิภาพในโปรเจกต์นี้เราจะใช้ชุดข้อมูลที่รวบรวมมาจาก ภาพสัตว์ต่าง ๆ ที่มีความหลากหลายเพื่อสร้างแบบจำแนกสัตว์ โดยเราจะใช้ซอฟต์แวร์และไลบรารี Machine Learning ที่มีอยู่และได้รับความนิยม เช่น TensorFlow หรือ scikit-learn เพื่อฝึกและทดสอบแบบจำแนก

1. เพื่อสร้างโปรแกรมจำแนกชนิดสัตว์ตามภาพ สามารถช่วยระบุสัตว์ได้
2. เพื่อนำไปศึกษาและประยุกต์เพื่อใช้ machine learning ในการจำแนกสิ่งอื่นๆได้

วัตถุประสงค์ของการวิจัย

เครื่องมือที่ใช้และเทคนิคที่ใช้

เครื่องมือที่ใช้(Visual Studio Code)

Visual Studio Code เป็นโปรแกรมแก้ไขซอร์สโค้ดที่พัฒนาโดยไมโครซอฟท์สำหรับ Windows, Linux และ macOS มีการสนับสนุนสำหรับการดีบัก การควบคุม Git ในตัวและ GitHub การเน้นไวยากรณ์ การเติมโค้ดอัจฉริยะ ตัวอย่าง และ code refactoring มันสามารถปรับแต่งได้ หลายอย่าง ให้ผู้ใช้สามารถเปลี่ยนธีม แป้นพิมพ์ลัด การตั้งค่า และติดตั้งส่วนขยายที่เพิ่มฟังก์ชันการ ทำงานเพิ่มเติม

Python เป็นภาษาการเขียนโปรแกรมที่ใช้อย่างแพร่หลายในเว็บแอปพลิเคชัน การพัฒนา ซอฟต์แวร์ วิทยาศาสตร์ข้อมูล และแมชชีนเลิร์นนิง (ML) นักพัฒนาใช้ Python เนื่องจากมี ประสิทธิภาพ เรียนรู้ง่าย และสามารถทำงานบนแพลตฟอร์มต่างๆ ได้มากมาย

เครื่องมือที่ใช้และเทคนิคที่ใช้

เทคนิคที่ใช้ (Python library)

Scikit-Learn เป็นไลบรารีฟรีในภาษาไพธอนสำหรับการพัฒนาโปรแกรมโดยใช้การเรียนรู้ ของเครื่อง จุดเด่นคือฟังก์ชันในการแบ่งประเภทข้อมูล การแบ่งกลุ่มข้อมูล การวิเคราะห์การถดถอย หลายอย่างไม่ว่าจะเป็นซัพพอร์ตเวกเตอร์แมชชีน การเรียนรู้ต้นไม้ตัดสินใจ และการแบ่งกลุ่มข้อมูลแบบเคมีน **scikit-learn** ถูกออกแบบสามารถใช้ร่วมกับไลบรารีนัมไพและไซไพของภาษาไพธอนได้ มี **5** อย่างหลักๆที่ใช้กันอย่างแพร่หลาย

1. Classification คือการแยกอีเมลว่าเป็นสแปมหรือไม่ **classification** ถือว่าเป็นหนึ่งใน แขนงของ **Supervised Learning** การเรียนรู้ของ อัลกอริทึมจากชุดข้อมูล**(Datasets)** ที่มีคำตอบที่ถูกต้อง

2. Regression เป็นเครื่องมือเพื่อเข้าใจความสัมพันธ์ระหว่างข้อมูล **Input** กับ **Output** ซึ่งก็ถือว่าเป็นอีกหนึ่งแขนงของ **Supervised Learning**

3. Clustering คือการจำแนกกลุ่มข้อมูลที่มีคุณสมบัติคล้ายกัน เครื่องมือนี้ถือว่าเป็นแขนงของ **Unsupervised Learning**

เครื่องมือที่ใช้และเทคนิคที่ใช้

4. Model selection อัลกอริทึมเพื่อใช้เปรียบเทียบตรวจสอบ และเลือกโมเดลและ Parameter ที่เหมาะสมกับชุดข้อมูลที่สุดใน โปรเจ็ค จะช่วยเพิ่มความแม่นยำของอัลกอริทึม Machine Learning ได้

5. Pre-processing ในขั้นตอนของ Data Analysis หรือการเข้าใจและวิเคราะห์ข้อมูล อาจต้องมีการแก้ไขให้ข้อมูลอยู่ในรูปแบบที่ เรานำไปใช้งานได้ เครื่องมือตัวนี้ของ Scikit-learn จะสามารถช่วยจัดการกับข้อมูลได้

Keras

เป็น open-source neural network(นิวรอล เน็ตเวิร์ค) ที่เขียนด้วยภาษา Python Keras ใช้โปรเจคแบบไหนได้บ้าง - ทำนาย คำตอบที่เป็นค่าต่อเนื่อง (จำนวนจริง) - ทำนาย คำตอบที่เป็นค่าไม่ต่อเนื่อง (Class) - ส่วนใหญ่ใช้ประมวลผลรูปภาพ

วิธีการดำเนินงาน

โปรเจค เรื่อง การจำแนกรูปสัตว์ตามกลุ่มโดยใช้การเรื่องรู้ของเครื่อง ด้วยภาษา Python เป็นการบูรณาการความรู้ ใน Machine learning จัดทำโดยโปรแกรม Visual Studio Code มีวิธีการดำเนินการ ดังนี้

- 3.1 ศึกษาและทำความเข้าใจข้อมูล
- 3.2 เครื่องมือที่ใช้ในการศึกษา
- 3.3 วิธีการเก็บรวบรวมข้อมูล
- 3.4 การวิเคราะห์ข้อมูล

import library ที่จำเป็นต่อการใช้ในการสร้าง animal image classifier

```
import pandas as pd
import numpy as np
import itertools
import keras
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from keras.models import Sequential
from keras import optimizers
from keras.preprocessing import image
from keras.layers import Dropout, Flatten, Dense
from keras import applications
from keras.utils import to_categorical
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
import math
import datetime
import time
```

การวิเคราะห์ข้อมูล

Loading up our image datasets

```
#Default dimensions we found online
img_width, img_height = 224, 224

#Create a bottleneck file
top_model_weights_path = 'bottleneck_fc_model.h5'

# loading up our datasets
train_data_dir = 'data/train'
validation_data_dir = 'data/validation'
test_data_dir = 'data/test'

# number of epochs to train top model
epochs = 7 #this has been changed after multiple model run
# batch size used by flow_from_directory and predict_generator
batch_size = 50
```

```
#Loading vgc16 model
vgg16 = applications.VGG16(include_top=False, weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 40s 1us/step
```

```
datagen = ImageDataGenerator(rescale=1. / 255)  #needed to create the bottleneck .npy files
```

การวิเคราะห์ข้อมูล

Creation of weights/features with VGG16

- การ train ข้อมูล

```
#__this can take an hour and half to run so only run it once.
#once the npy files have been created, no need to run again. Convert this cell to a code cell to run.__

start = datetime.datetime.now()

generator = datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)

nb_train_samples = len(generator.filenames)
num_classes = len(generator.class_indices)

predict_size_train = int(math.ceil(nb_train_samples / batch_size))

bottleneck_features_train = vgg16.predict_generator(generator, predict_size_train)

np.save('bottleneck_features_train.npy', bottleneck_features_train)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

การวิเคราะห์ข้อมูล

การวิเคราะห์ข้อมูล

- การทำ validation ข้อมูล

```
#__this can take half an hour to run so only run it once. once the npy files have been created, no need to run again. Convert this cell to a code cell to run.__

start = datetime.datetime.now()
generator = datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)

nb_validation_samples = len(generator.filenames)

predict_size_validation = int(math.ceil(nb_validation_samples / batch_size))

bottleneck_features_validation = vgg16.predict_generator(
    generator, predict_size_validation)

np.save('bottleneck_features_validation.npy', bottleneck_features_validation)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

- การ test ข้อมูล

```
#__this can take half an hour to run so only run it once. once the npy files have been created, no need to run again. Convert this cell to a code cell to run.__

start = datetime.datetime.now()
generator = datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode=None,
    shuffle=False)

nb_test_samples = len(generator.filenames)

predict_size_test = int(math.ceil(nb_test_samples / batch_size))

bottleneck_features_test = vgg16.predict_generator(
    generator, predict_size_test)

np.save('bottleneck_features_test.npy', bottleneck_features_test)
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

Loading training, validation and testing data - training data

```
#training data
generator_top = datagen.flow_from_directory(
        train_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False)

nb_train_samples = len(generator_top.filenames)
num_classes = len(generator_top.class_indices)

# load the bottleneck features saved earlier
train_data = np.load('bottleneck_features_train.npy')

# get the class lebels for the training data, in the original order
train_labels = generator_top.classes

# convert the training labels to categorical vectors
train_labels = to_categorical(train_labels, num_classes=num_classes)
```

การวิเคราะห์ข้อมูล

การวิเคราะห์ข้อมูล

- validation data

```
#validation data
generator_top = datagen.flow_from_directory(
        validation_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

nb_validation_samples = len(generator_top.filenames)

validation_data = np.load('bottleneck_features_validation.npy')


validation_labels = generator_top.classes
validation_labels = to_categorical(validation_labels, num_classes=num_classes)
```

- testing data

```
#testing data
generator_top = datagen.flow_from_directory(
        test_data_dir,
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode=None,
        shuffle=False)

nb_test_samples = len(generator_top.filenames)

test_data = np.load('bottleneck_features_test.npy')


test_labels = generator_top.classes
test_labels = to_categorical(test_labels, num_classes=num_classes)
```

Training of model

```
#This is the best model we found. For additional models, check out I_notebook.ipynb
start = datetime.datetime.now()
model = Sequential()
model.add(Flatten(input_shape=train_data.shape[1:]))
model.add(Dense(100, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.5))
model.add(Dense(50, activation=keras.layers.LeakyReLU(alpha=0.3)))
model.add(Dropout(0.3))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])

history = model.fit(train_data, train_labels,
      epochs=7,
      batch_size=batch_size,
      validation_data=(validation_data, validation_labels))

model.save_weights(top_model_weights_path)

(eval_loss, eval_accuracy) = model.evaluate(
 validation_data, validation_labels, batch_size=batch_size, verbose=1)

print("[INFO] accuracy: {:.2f}%".format(eval_accuracy * 100))
print("[INFO] Loss: {}".format(eval_loss))
end= datetime.datetime.now()
elapsed= end-start
print ('Time: ', elapsed)
```

```
#Model summary
model.summary()
```

การวิเคราะห์ข้อมูล

Model Evaluation on Testing Set

```
model.evaluate(test_data, test_labels)
```

Classification metrics and Confusion Matrix

- Classification Metrics

```
print('test data', test_data)
preds = np.round(model.predict(test_data),0)
#to fit them into classification metrics and confusion metrics, some additional modificaitions are required
print('rounded test_labels', preds)
```

```
animals = ['butterflies', 'chickens', 'elephants', 'horses', 'spiders', 'squirells']
classification_metrics = metrics.classification_report(test_labels, preds, target_names=animals )
print(classification_metrics)
```

Confusion Matrix

```
#Since our data is in dummy format we put the numpy array into a dataframe and call idxmax axis=1 to return the column
# label of the maximum value thus creating a categorical variable
#Basically, flipping a dummy variable back to it's categorical variable
categorical_test_labels = pd.DataFrame(test_labels).idxmax(axis=1)
categorical_preds = pd.DataFrame(preds).idxmax(axis=1)
```

```
confusion_matrix= confusion_matrix(categorical_test_labels, categorical_preds)
```

การวิเคราะห์ข้อมูล

Testing images on model

```python
def read_image(file_path):
    print("[INFO] loading and preprocessing image...")
    image = load_img(file_path, target_size=(224, 224))
    image = img_to_array(image)
    image = np.expand_dims(image, axis=0)
    image /= 255.
    return image
```

```python
def test_single_image(path):
    animals = ['butterflies', 'chickens', 'elephants', 'horses', 'spiders', 'squirells']
    images = read_image(path)
    time.sleep(.5)
    bt_prediction = vgg16.predict(images)
    preds = model.predict(bt_prediction)
    for idx, animal, x in zip(range(0,6), animals , preds[0]):
        print("ID: {}, Label: {} {}%".format(idx, animal, round(x*100,2) ))
    print('Final Decision:')
    time.sleep(.5)
    for x in range(3):
        print('.'*(x+1))
        time.sleep(.2)
    class_predicted = np.argmax(preds, axis=-1)
    # class_predicted = model.predict_classes(bt_prediction)
    class_dictionary = generator_top.class_indices
    inv_map = {v: k for k, v in class_dictionary.items()}
    print("ID: {}, Label: {}".format(class_predicted[0], inv_map[class_predicted[0]]))
    return load_img(path)
```

```python
path = 'data/test/wildlifeelephant.jpg'
```

```python
test_single_image(path)
```

การวิเคราะห์ข้อมูล

สรุปผลและอภิปรายผล

จากการศึกษาและประมวลผลการจำแนกชนิดของสัตว์โดยใช้เทคนิค CNN จะสรุปได้ว่า รูปภาพที่ นำไปใช้ ซึ่งเป็นรูปภาพของช้าง พบว่า Machine learning สามารถจำแนกผลลัพธ์ความคล้ายของสัตว์ ได้ดังนี้

ผีเสื้อ คิดเป็นร้อยละ 0.14

ไก่ คิดเป็นร้อยละ 1.92

ม้า คิดเป็นร้อยละ 1.52

แมงมุม คิดเป็นร้อยละ 0.63

กระรอก คิดเป็นร้อยละ 4.85 และ

ช้าง คิดเป็นร้อยละ 90.95 พบได้ว่า Machine learning สามารถจำแนก สัตว์ชนิดใดได้อย่างแม่นยำ

ทำการทดลองใช้รูปภาพนี้เพื่อให้ **Machine learning** ทำการจำแนก

```
[INFO] loading and preprocessing image...
1/1 [==============================] - 0s 124ms/step
1/1 [==============================] - 0s 25ms/step
ID: 0, Label: butterflies 0.14%
ID: 1, Label: chickens 1.92%
ID: 2, Label: elephants 90.95%
ID: 3, Label: horses 1.52%
ID: 4, Label: spiders 0.63%
ID: 5, Label: squirells 4.85%
Final Decision:
.
..
...
ID: 2, Label: elephants
```

ผลลัพธ์ของการจำแนกชนิดสัตว

THANK YOU

THANK YOU