



BEE OR WASP BINARY IMAGE CLASSIFICATION USING LOGISTIC REGRESSION

6310401165 วิศรุต ห้อมแก่นจันทร์

ที่มาและปัญหา

โปรเจคนี้เกี่ยวกับการแยกแยะสัตว์ที่มีลักษณะคล้ายกันโดยใช้เทคนิคการ *LOGISTIC REGRESSION* เพื่อช่วยให้ผู้ใช้ได้แยกแยะระหว่างผึ้งกับตัวต่อได้อย่างถูกต้อง โดยต้องอาศัยข้อมูลตัวเกรนในจำนวนที่มากเพื่อเพิ่มความแม่นยำในการบ่งบอกชนิดของสัตว์ได้มากขึ้น

โดยภาพรวมการทำงานของโปรแกรมนี้จะทำงานได้โดยให้ผู้ใช้ป้อนข้อมูลเป็นรูปภาพผึ้งหรือตัวต่อ หลังจากนั้นโมเดลก็จะทำการคำนวณอุ่นมา ว่าภาพที่ผู้ใช้ป้อนเข้ามาเป็นผึ้งหรือตัวต่อ



วัตถุประสงค์



1. เพื่อคัดแยกระหว่างผึ้งและตัวต่อโดยใช้เทคโนโลยี *MACHINE LEARNING* ได้อย่างถูกต้องที่สุดเท่าที่จะเป็นไปได้
2. เพื่อให้ผู้ใช้งานโปรแกรมของโครงงานได้พัฒนาความรู้ของตัวผู้ใช้งานเอง โดยให้ผู้ใช้งานทำการค้นหารูปของผึ้งหรือตัวต่อเพื่อที่จะให้โปรแกรมคำนายนิดของสัตว์ได้อย่างถูกต้อง

ประโยชน์ที่ได้รับ

1. ผู้ใช้งานโปรแกรมของโครงงานสามารถแยกแยะระหว่างผึ้งและตัวต่อที่เกิดจากผู้ใช้งานป้อนรูปภาพผิวเสื่อเข้าไปให้โ้มเดลทำงาน
2. ผู้ใช้งานโปรแกรมของโครงงานจะสามารถแยกแยะระหว่างผึ้งและตัวต่อได้โดยอาศัยการสังเกตลักษณะต่างๆของสัตว์
3. ผู้ใช้งานโปรแกรมของโครงงานสามารถนำความรู้เรื่องลักษณะของผึ้งหรือตัวต่อไปใช้ประโยชน์ในด้านอื่นได้ เช่น การกำจัดที่เกี่ยวข้องกับแมลง



หลักการและเทคนิคที่นำมาใช้

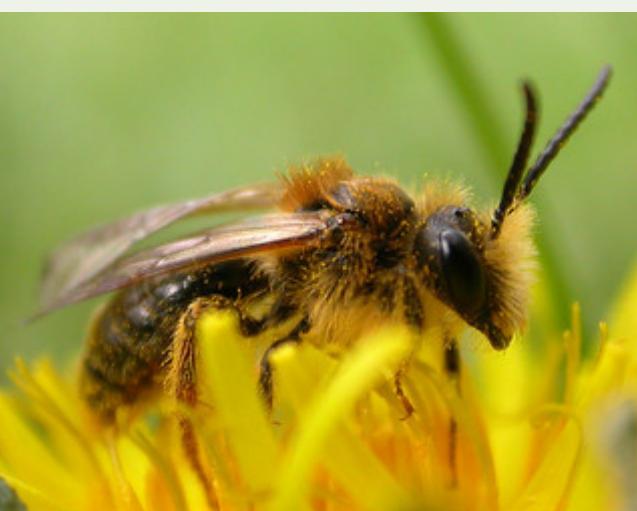
ลักษณะของข้อมูล และแหล่งข้อมูล

รูปภาพผึ้ง จำนวน 3183 ภาพ
และตัวต่อ จำนวน 4943 ภาพ
จากเว็บไซต์ Kaggle.com

BEE



WASP



การจัดการข้อมูลที่ไม่จำเป็น

Clean Data

```
path = 'datasets'
label = os.path.join(path, 'labels.csv')
data = pd.read_csv(label)

data.label.unique()

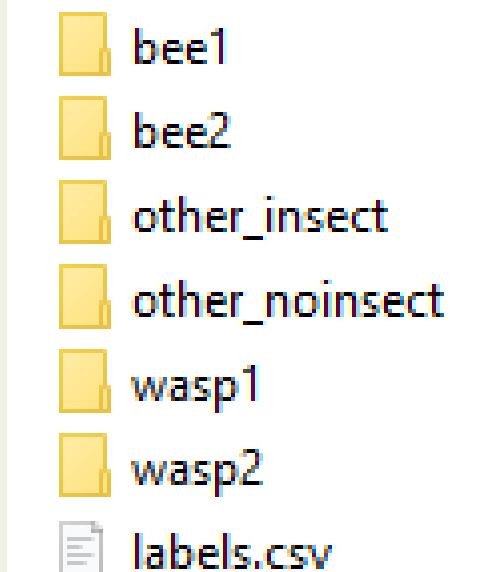
array(['bee', 'wasp', 'insect', 'other'], dtype=object)

data.label.value_counts()

label
wasp      4943
bee       3183
insect    2439
other     856
Name: count, dtype: int64

data = data[((data['label']=='bee') | (data['label']=='wasp')) & (data['photo_quality']==1)]
data.label.value_counts()

label
bee      2469
wasp    2127
Name: count, dtype: int64
```



การแบ่งข้อมูล

```

dataset = "datasets"

train_wasp = sorted(os.listdir(dataset +'/wasp1'))
train_bee = sorted(os.listdir(dataset +'/bee1'))

test_wasp = sorted(os.listdir(dataset +'/wasp2'))
test_bee = sorted(os.listdir(dataset +'/bee2'))

list=["train_wasp","train_bee","test_wasp","test_bee"]

wasp1=0
wasp2=0
bee1=0
bee2=0

for i in train_wasp:
    wasp1=wasp1+1

for i in train_bee:
    bee1=bee1+1

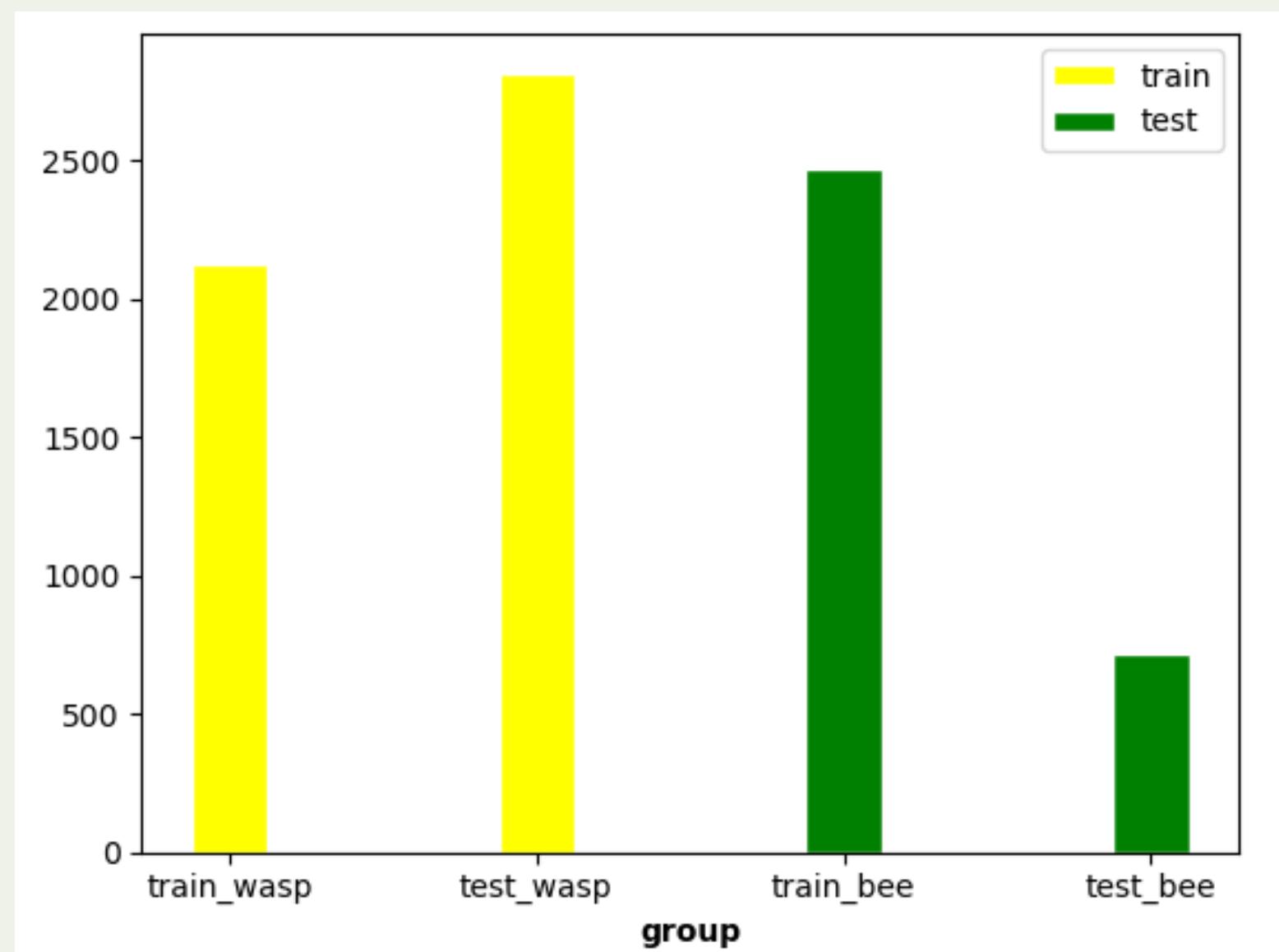
for i in test_wasp:
    wasp2=wasp2+1

for i in test_bee:
    bee2=bee2+1

print("train_wasp :",wasp1)
print("test_wasp :",wasp2)
print("train_bee :",bee1)
print("test_bee :",bee2)

train_wasp : 2127
test_wasp : 2816
train_bee : 2469
test_bee : 714

```



การเตรียมข้อมูลสำหรับการเรียนรู้

```

img_size = 50
wasp_insect = []
bee_insect = []
label = []

for i in train_wasp:
    if os.path.isfile(dataset +'/wasp1/'+ i):
        insect = Image.open(dataset +'/wasp1/'+ i).convert('L') #converting grey scale
        insect = insect.resize((img_size,img_size), PIL.Image.Resampling.LANCZOS) #resizing to 50,50
        insect = np.asarray(insect)/255.0 #normalizing images
        wasp_insect.append(insect)
        label.append(1)

for i in train_bee:
    if os.path.isfile(dataset +'/bee1/'+ i):
        insect = Image.open(dataset +'/bee1/'+ i).convert('L')
        insect = insect.resize((img_size,img_size), PIL.Image.Resampling.LANCZOS)
        insect = np.asarray(insect)/255.0 #normalizing images
        bee_insect.append(insect)
        label.append(0)

x_train = np.concatenate((wasp_insect,bee_insect),axis=0) # training dataset
x_train_label = np.asarray(label) # label array containing 0 and 1
x_train_label = x_train_label.reshape(x_train_label.shape[0],1)

print("wasp_insect:",np.shape(wasp_insect), "bee_insect:",np.shape(bee_insect))
print("train_dataset:",np.shape(x_train), "train_values:",np.shape(x_train_label))

wasp_insect: (2127, 50, 50) bee_insect: (2469, 50, 50)
train_dataset: (4596, 50, 50) train_values: (4596, 1)

```

```

img_size = 50
wasp_insect = []
bee_insect = []
label = []

for i in test_wasp:
    if os.path.isfile(dataset +'/wasp2/'+ i):
        insect = Image.open(dataset +'/wasp2/'+ i).convert('L')
        insect = insect.resize((img_size,img_size), PIL.Image.Resampling.LANCZOS)
        insect = np.asarray(insect)/255.0
        wasp_insect.append(insect)
        label.append(1)

for i in test_bee:
    if os.path.isfile(dataset +'/bee2/'+ i):
        faces = Image.open(dataset +'/bee2/'+ i).convert('L')
        faces = faces.resize((img_size,img_size), PIL.Image.Resampling.LANCZOS)
        faces = np.asarray(faces)/255.0
        bee_insect.append(faces)
        label.append(0)

x_test = np.concatenate((wasp_insect,bee_insect),axis=0) # test dataset
x_test_label = np.asarray(label) # corresponding labels
x_test_label = x_test_label.reshape(x_test_label.shape[0],1)

print("wasp_insect:",np.shape(wasp_insect), "bee_insect:",np.shape(bee_insect))
print("test_dataset:",np.shape(x_test), "test_values:",np.shape(x_test_label))

wasp_insect: (2816, 50, 50) bee_insect: (714, 50, 50)
test_dataset: (3530, 50, 50) test_values: (3530, 1)

```



การเตรียมข้อมูลสำหรับการใช้งาน

```
x = np.concatenate((x_train,x_test),axis=0) #train_data
y = np.concatenate((x_train_label,x_test_label),axis=0) #test data
x = x.reshape(x.shape[0],x.shape[1]*x.shape[2]) #flatten 3D image array to 2D
print("images:",np.shape(x), "labels:",np.shape(y))

images: (8126, 2500) labels: (8126, 1)

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x, y, test_size=0.2, random_state=42)
number_of_train = X_train.shape[0]
number_of_test = X_test.shape[0]

print("train number:",number_of_train, "test number:",number_of_test)

train number: 6500 test number: 1626
```



IMPLEMENTING LOGISTIC REGRESSION

```
def initialize_weights_and_bias(dimension):  
    w = np.full((dimension,1),0.01)  
    b = 0.0  
    return w, b
```

```
def sigmoid(z):  
    return 1 / (1 + np.exp(-z))
```

```
def forward_backward_propagation(w,b,x_train,y_train):  
    # forward propagation  
    z = np.dot(w.T,x_train) + b  
    y_head = sigmoid(z)  
    loss = -(1-y_train)*np.log(1-y_head)-y_train*np.log(y_head)  
    cost = (np.sum(loss))/x_train.shape[1]  
  
    # backward propagation  
    derivative_weight = (np.dot(x_train,((y_head-y_train).T)))/x_train.shape[1]  
    derivative_bias = np.sum(y_head-y_train)/x_train.shape[1]  
    gradients = {"derivative_weight": derivative_weight,"derivative_bias": derivative_bias}  
    return cost,gradients
```

IMPLEMENTING LOGISTIC REGRESSION

```

def update(w, b, x_train, y_train, learning_rate, number_of_iterarion):
    cost_list = []
    cost_list2 = []
    index = []
    # updating(learning) parameters is number_of_iterarion times
    for i in range(number_of_iterarion):
        # make forward and backward propagation and find cost and gradients
        cost, gradients = forward_backward_propagation(w,b,x_train,y_train)
        cost_list.append(cost)
        # Lets update
        w = w - learning_rate * gradients["derivative_weight"]
        b = b - learning_rate * gradients["derivative_bias"]
        if i % 50 == 0:
            cost_list2.append(cost)
            index.append(i)
            print ("Cost after iteration %i: %f" %(i, cost))
    # we update(learn) parameters weights and bias
    parameters = {"weight": w, "bias": b}
    plt.plot(index,cost_list2)
    plt.xticks(index,rotation='vertical')
    plt.xlabel("Number of Iterarion")
    plt.ylabel("Cost")
    plt.show()
    return parameters, gradients, cost_list

```

```

def predict(w, b, x_test):
    z = sigmoid(np.dot(w.T, x_test) + b)
    y_prediction = np.where(z <= 0.5, 0, 1)
    return y_prediction

```

```

def logistic_regression(x_train, y_train, x_test, y_test, learning_rate, num_iterations):
    # initialize
    dimension = x_train.shape[0] # 2500
    w, b = initialize_weights_and_bias(dimension)
    parameters, gradients, cost_list = update(w, b, x_train, y_train, learning_rate, num_iterations)

    y_prediction_test = predict(parameters["weight"], parameters["bias"], x_test)
    y_prediction_train = predict(parameters["weight"], parameters["bias"], x_train)

    train_acc_lr = round((100 - np.mean(np.abs(y_prediction_train - y_train)) * 100), 2)
    test_acc_lr = round((100 - np.mean(np.abs(y_prediction_test - y_test)) * 100), 2)
    # Print train/test Errors
    print("train accuracy: %", train_acc_lr)
    print("test accuracy: %", test_acc_lr)
    return train_acc_lr, test_acc_lr, w, b

```

```

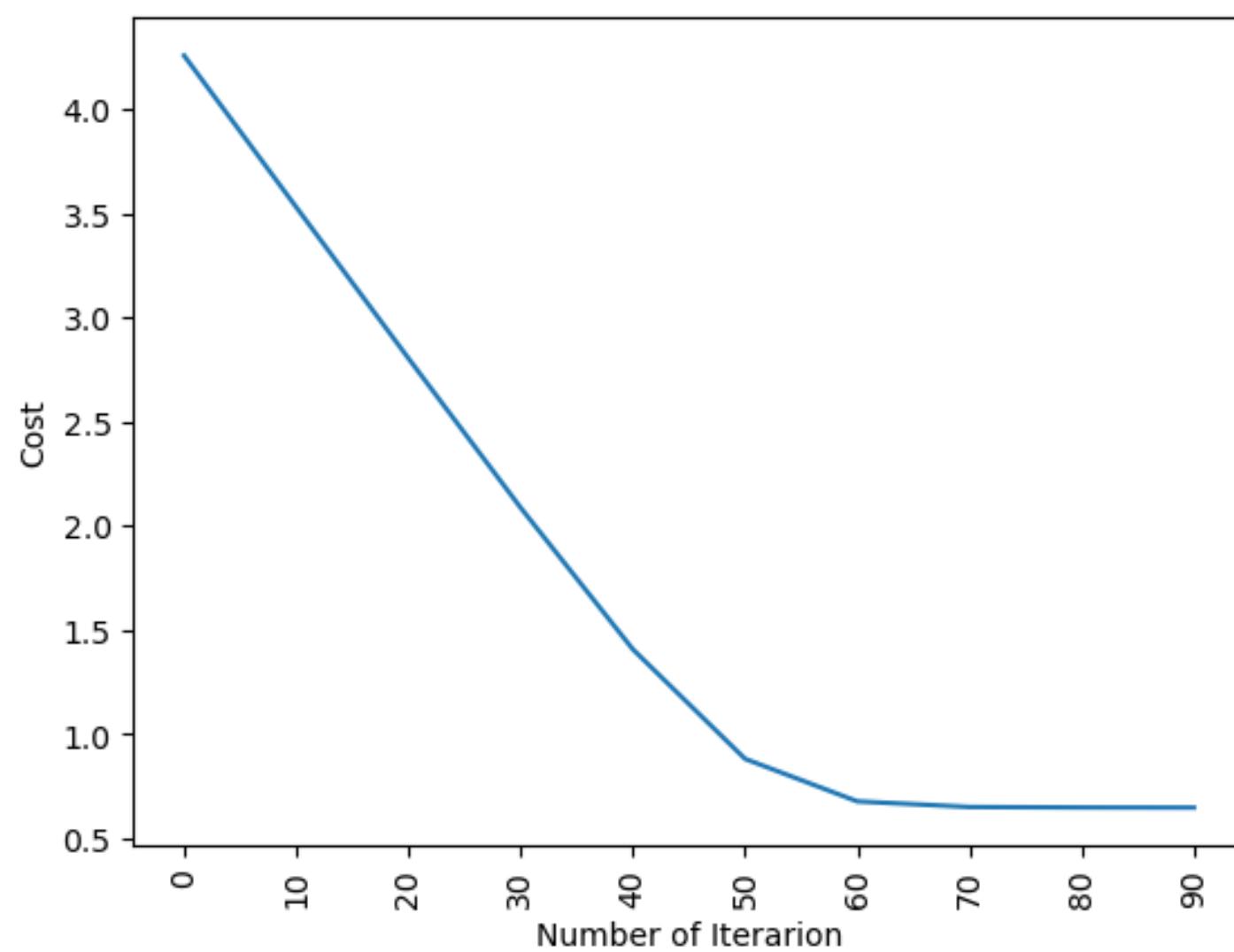
train_acc_lr, test_acc_lr, w, b = logistic_regression(x_train, y_train, x_test, y_test, learning_rate = 0.001, num_iterations = 100)

model = {"weight": w, "bias": b}

```

OUTPUT

```
Cost after iteration 0: 4.257751
Cost after iteration 10: 3.530008
Cost after iteration 20: 2.804560
Cost after iteration 30: 2.087735
Cost after iteration 40: 1.408219
Cost after iteration 50: 0.882543
Cost after iteration 60: 0.678557
Cost after iteration 70: 0.651580
Cost after iteration 80: 0.649284
Cost after iteration 90: 0.648656
```



train accuracy: % 60.83
test accuracy: % 60.82

OUTPUT

Number of Iteration

train accuracy: % 60.83
test accuracy: % 60.82

User Input

```
[*]: filetypes = (  
    ('Image files', '*.jpg'),  
    ('Image files', '*.jpeg'),  
    ('Image files', '*.png'),  
)  
  
root = tk.Tk()  
root.attributes('-topmost',True)  
root.iconify()  
  
img_path = askopenfilename(title='Choose your shark picture',  
                           filetypes=filetypes,  
                           parent=root)  
  
root.destroy()  
  
[ ]: display(Image(img_path))  
  
[ ]: def preprocess_image(image_path):  
    img = PIL.Image.open(image_path).convert("L")  
    img = img.resize((50, 50),PIL.Image.Resampling.LANCZOS)  
    img_array = np.array(img).flatten()  
    img_array = img_array / 255.0  
    return img_array.reshape(1, -1)
```

Choose your shark picture

Local Disk (D:) > test_image

Organize New folder

example_notebook
kaggle_bee_vs_wasp1
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
Local Disk (D:)

File name: 40277996_34bc230ad7_n.jpg

Image files (*.jpg;*.jpeg;*.png)

Open Cancel



OUTPUT

```
[26]: display(Image(img_path))
```



```
[27]: def preprocess_image(image_path):
    img = PIL.Image.open(image_path).convert("L")
    img = img.resize((50, 50), PIL.Image.Resampling.LANCZOS)
    img_array = np.array(img).flatten()
    img_array = img_array / 255.0
    return img_array.reshape(1, -1)
```

```
[28]: def predict_image_class(image_path, model):
    # Load model parameters
    w = model["weight"]
    b = model["bias"]
    # Preprocess the input image
    preprocessed_image = preprocess_image(image_path)
    # Make predictions using the logistic regression model
    predicted_class = predict(w, b, preprocessed_image.T)
    return predicted_class
```

```
[29]: predicted_class = np.argmax(predict_image_class(img_path, model))
print('This is {}'.format(output_classes[predicted_class]))
```

This is bee.

THANK YOU
FOR YOUR
ATTENTION

