Artificial Intelligence and Machine Learning

(ใบงานที่ 2)

เสนอ

อาจารย์ รุจิพันธุ์ โกษารัตน์

จัดทำโดย

67543210031-0 นายธนภัทร นุกูล

คณะวิศวกรรมศาสตร์

สาขาวิชา วศ.บ.วิศวกรรมซอฟต์แวร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา เชียงใหม่

Ornamental Plant Images Dataset

ชื่อชุดข้อมูล: Ornamental Plant Images Dataset

แหล่งที่มา (Source): Kaggle (Contributor: Muhammad Irvan Arfirza)

ประเภทของปัญหา: การจำแนกประเภทรูปภาพ (Image Classification) แบบ Supervised Learning

วัตถุประสงค์: เพื่อใช้ฝึกฝนโมเดลปัญญาประดิษฐ์ให้สามารถแยกแยะสายพันธุ์ของไม้ประดับชนิดต่างๆ จากภาพถ่ายได้

Decision Tree

การทำงานของโค้ด

1. นำเข้าไลบรารี
   - **Random:** สุ่มภาพตัวอย่าง
   - **Kagglehub:** ดาวน์โหลด dataset จาก Kaggle
   - **Os:** จัดการไฟล์และโฟลเดอร์
   - **Numpy:** จัดการข้อมูลเชิงตัวเลข
   - **Matplotlib:** แสดงภาพและกราฟ
   - **PIL.Image:** เปิดและแก้ไขภาพ
   - **sklearn.model_selection:** แบ่งข้อมูล Train/Test
   - **DecisionTreeClassifier:** โมเดล Decision Tree
   - **plot_tree:** วาดโครงสร้างต้นไม้
   - **metrics:** วัดประสิทธิภาพโมเดล
2. โหลด Dataset จาก Kagglehub
   - dataset_path = kagglehub.dataset_download
     ("muhammadirvanarfirza/decorative-plant-image-dataset")
3. กำหนดขนาดทุกภาพให้เป็น 64×64 pixel
   - IMG_SIZE = (64, 64)
4. ค้นหาโฟลเดอร์ที่มีรูปภาพ
   - ใช้ os.walk() ไล่ทุกโฟลเดอร์
   - ถ้าเจอไฟล์รูป ⟶ คืน path นั้นทันที
   - def find_image_folder(start_path)
5. **def load_data(root_path):** โหลดและเตรียมข้อมูลภาพ
   - **images** : เก็บข้อมูลภาพหลังจากแปลงเป็นอาเรย์
   - **labels** : เก็บค่าป้ายกำกับ (Label) ของแต่ละภาพ
   - **class_names** : เก็บชื่อคลาส (ชนิดของต้นไม้)
6. แบ่งข้อมูลเป็น Training และ Testing Set
   - X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.3, random_state=42)

- ใช้ข้อมูล 70% สำหรับฝึกโมเดล
- ใช้ข้อมูล 30% สำหรับทดสอบโมเดล

7. **clf = DecisionTreeClassifier:** สร้างและฝึกโมเดล Decision Tree
8. ประเมินประสิทธิภาพของโมเดล
   - y_pred = clf.predict(X_test)
     accuracy_score(y_test, y_pred)
   - classification_report(y_test, y_pred, target_names=class_names)
9. แสดง Decision Tree
   - plt.figure กำหนดขนาดหน้าต่าง
   - plt.imshow แสดงภาพตัวอย่างขาวดำ
   - pred_cls ทำกราฟทำนายคลาสของภาพตัวอย่าง
   - true_cls ดึงค่าคลาสที่ถูกต้อง จาก y_test
   - plt.title กำหนดชื่อเรื่อง
   - plt.axis('off') ปิดการแสดงเส้นแกน X และ Y
   - plt.show() แสดงภาพและผลลัพธ์

Code Python:

```
import random
import kagglehub
import os
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, classification_report

print("--- 1. Downloading Dataset from Kaggle ---")
try:
    dataset_path = kagglehub.dataset_download(
        "muhammadirvanarfirza/decorative-plant-image-dataset")
    print(f"Original Download Path: {dataset_path}")
except Exception as e:
    print(f"Error downloading: {e}")
    exit()

IMG_SIZE = (64, 64)

def find_image_folder(start_path):
    for root, dirs, files in os.walk(start_path):
        if len(files) > 0:
            if any(f.lower().endswith(('.png', '.jpg', '.jpeg')) for f in files):
                return os.path.dirname(root)
    return start_path

def load_data(root_path):
    images = []
    labels = []
    class_names = []

    target_path = find_image_folder(root_path)
    print(f"Corrected Dataset Path: {target_path}")

    if os.path.exists(target_path):
        folder_list = sorted(os.listdir(target_path))

        for folder in folder_list:
            folder_path = os.path.join(target_path, folder)

            if os.path.isdir(folder_path) and not folder.startswith('.'):

                files_inside = os.listdir(folder_path)
                has_images = any(f.lower().endswith(('.jpg', '.png'))
                                 for f in files_inside)

                if has_images:
                    class_names.append(folder)
                    label_index = len(class_names) - 1

                    count = 0
                    for img_file in files_inside:
                        try:
                            if img_file.lower().endswith(('.jpg', '.png', '.jpeg')):
                                img_path = os.path.join(folder_path, img_file)

                                # Use PIL
                                with Image.open(img_path) as img:
                                    img = img.convert('L')       # Grayscale
                                    img = img.resize(IMG_SIZE)   # Resize
                                    img_array = np.array(img)    # Numpy

                                    images.append(img_array.flatten())
                                    labels.append(label_index)
                                    count += 1
```

```
                    except Exception as e:
                        pass
                print(f" - Loaded class '{folder}': {count} images")

    return np.array(images), np.array(labels), class_names

print("\n--- 2. Processing Images ---")
X, y, class_names = load_data(dataset_path)

if len(X) == 0:
    print("Error: Still no images found.")
    print("Check if the folder structure contains .jpg or .png files.")
    exit()

print(f"Done: {len(X)} images loaded. Feature shape: {X.shape}")

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42)

print("\n--- 3. Training Decision Tree Model ---")
clf = DecisionTreeClassifier(
    criterion='entropy', max_depth=10, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
acc = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {acc * 100:.2f}%")
print("\nClassification Report:")
print(classification_report(y_test, y_pred, target_names=class_names))

# Plot Tree
print("Plotting Decision Tree...")
plt.figure(figsize=(15, 10))
plot_tree(clf,
          feature_names=[f"px_{i}" for i in range(X.shape[1])],
          class_names=class_names,
          filled=True,
          rounded=True,
          max_depth=3,)

plt.title("Decision Tree ( Ornamental Plant )")
plt.show()

# Show Sample Prediction
try:
    idx = random.randint(0, len(X_test)-1)
    sample_img = X_test[idx].reshape(IMG_SIZE)

    plt.figure(figsize=(5, 5))
    plt.imshow(sample_img, cmap='gray')
    pred_cls = class_names[clf.predict([X_test[idx]])[0]]
    true_cls = class_names[y_test[idx]]
    plt.title(f"Actual: {true_cls} | Predicted: {pred_cls}")
    plt.axis('off')
    plt.show()
except Exception as e:
    print(f"Error showing sample image: {e}")
```

Result:

```
Done: 11581 images loaded. Feature shape: (11581, 4096)

--- 3. Training Decision Tree Model ---

Accuracy: 53.32%
```

Actual: Pakis-Leather Leaf Fern | Predicted: Pakis-Leather Leaf Fern

Decision Tree (Ornamental Plant)