# Pollen's Profiling: Automated Classification of Pollen Grains

## 1. Project Overview

This project aims to classify different types of pollen grains using machine learning. By processing microscopic images of pollen grains and extracting key features, we can build an automated model to recognize and categorize them.

## 2. Project Flow

1. Collect Dataset

2. Exploratory Data Analysis (EDA)

3. Image Preprocessing

4. Feature Extraction

5. Model Training

6. Model Evaluation

7. Prediction & Conclusion

## 3. Prior Knowledge Required

- Python Programming

- NumPy, Pandas, Matplotlib, Seaborn

- Scikit-learn (for ML)

- OpenCV (for image processing)

- Basic understanding of CNNs (for deep learning approach)

## 4. Project Objectives

- Understand pollen structure using image data

- Preprocess images and extract features

- Build an ML model to classify pollen grain types

- Evaluate model performance with metrics like accuracy and confusion matrix

## 5. Dataset Collection

You can use this public dataset:

Pollen Grain Dataset - Kaggle: https://www.kaggle.com/datasets/itachi9604/dataset-pollen

Contains images of different pollen types (e.g., Daisy, Dandelion, Rose, Sunflower, Tulip).

## 6. Exploratory Data Analysis (EDA)

Steps:

- Load dataset and labels

- Display sample images from each class

- Analyze class distribution

- Identify image resolutions, quality, noise levels

Example code:

```python

import matplotlib.pyplot as plt

import os

import cv2

path = "/path/to/dataset"

categories = os.listdir(path)

for category in categories:

folder = os.path.join(path, category)

sample\_img = os.listdir(folder)[0]

img = cv2.imread(os.path.join(folder, sample\_img))

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.title(category)

plt.axis('off')

plt.show()

```

## 7. Image Preprocessing

- Resize images to standard shape (e.g., 128x128)

- Normalize pixel values

- Convert to grayscale or keep RGB depending on model

## 8. Feature Extraction

Option 1: Manual

- Use OpenCV to extract texture, shape, and edge features (e.g., contours, Hu moments)

Option 2: Automatic

- Use Convolutional Neural Networks (CNNs) to automatically extract features

## 9. Model Training

- Split data into train/test sets

- Choose model:

- SVM / KNN / Random Forest (for classical ML)

- CNN (for deep learning)

Example: Using CNN with Keras

```python

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([

Conv2D(32, (3,3), activation='relu', input\_shape=(128,128,3)),

MaxPooling2D(2,2),

Flatten(),

Dense(64, activation='relu'),

Dense(5, activation='softmax') # Assuming 5 pollen types

])

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

```

## 10. Model Evaluation

- Accuracy

- Precision, Recall, F1 Score

- Confusion Matrix

- Visualize predictions with sample images

## 11. Conclusion

- Discuss performance results

- Challenges faced in image preprocessing or misclassifications

- Future improvements like data augmentation or using pre-trained models (e.g., VGG16, ResNet)

## 12. Data Augmentation (to improve model generalization)

Python code:

from keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

import cv2

import os

datagen = ImageDataGenerator(rotation\_range=30, zoom\_range=0.2,

width\_shift\_range=0.1, height\_shift\_range=0.1,

horizontal\_flip=True)

img\_path = "/path/to/dataset/Daisy/sample.jpg"

img = cv2.imread(img\_path)

img = cv2.resize(img, (128, 128))

img = img.reshape((1,) + img.shape)

# Generate and plot 5 augmented images

i = 0

for batch in datagen.flow(img, batch\_size=1):

plt.imshow(batch[0].astype('uint8'))

plt.axis('off')

plt.show()

i += 1

if i == 5:

break

Output: Displays 5 different augmented versions of the same pollen image.

## 13. Plot Confusion Matrix

Python code:

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

import numpy as np

true\_labels = [0, 1, 2, 2, 0, 1, 4, 3, 3, 4]

predicted\_labels = [0, 2, 2, 2, 0, 1, 4, 3, 3, 4]

cm = confusion\_matrix(true\_labels, predicted\_labels)

labels = ['Daisy', 'Dandelion', 'Rose', 'Sunflower', 'Tulip']

plt.figure(figsize=(6,5))

sns.heatmap(cm, annot=True, fmt='d', xticklabels=labels, yticklabels=labels, cmap="YlGnBu")

plt.xlabel("Predicted")

plt.ylabel("True")

plt.title("Confusion Matrix")

plt.show()

Output: A heatmap showing the confusion matrix between predicted and actual pollen categories.

## 14. Use Pre-trained VGG16 for Feature Extraction

Python code:

from keras.applications.vgg16 import VGG16, preprocess\_input

from keras.preprocessing.image import load\_img, img\_to\_array

import numpy as np

model = VGG16(include\_top=False, input\_shape=(128, 128, 3))

model.summary() # Optional: print architecture

img = load\_img('/path/to/image.jpg', target\_size=(128, 128))

img\_array = img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0)

img\_array = preprocess\_input(img\_array)

features = model.predict(img\_array)

print("Extracted features shape:", features.shape)

Output:  
Extracted features shape: (1, 4, 4, 512)