# Prepare and explore the Data set, (Fruits Classes by kagle)

this note book is made to prepare and explore the data set

```python
import numpy as np
import matplotlib.pyplot as plt
```

## Install The data set

install the data set from kaggle hub into the local host  move it from the default locatoion into the current directory

### Library:

Animals-10: **https://www.kaggle.com/datasets/alessiocorrado99/animals10**

took from it the spiders photos

Animal Image Dataset: **https://www.kaggle.com/datasets/ashishsaxena2209/animal-image-datasetdog-cat-and-panda**

took from it the cats and pandas photos

```python
import kagglehub

# Download latest version
path = kagglehub.dataset_download("alessiocorrado99/animals10")
print("Path to dataset files:", path)
```

```
Downloading from
https://www.kaggle.com/api/v1/datasets/download/alessiocorrado99/animals10?
dataset_version_number=2...

100%|██████████| 586M/586M [00:28<00:00, 21.6MB/s]

Extracting files...



Path to dataset files: C:\Users\HP\.cache\kagglehub\datasets\alessiocorrado99\
animals10\versions\2
```

```python
import shutil
target_path = r"."

# Move the whole dataset folder
shutil.move(path, target_path)

print("Moved dataset to:", target_path)
```

```
Moved dataset to: .
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("ashishsaxena2209/animal-image-datasetdog-cat-and-panda")

print("Path to dataset files:", path)
```

```
Resuming download from 123731968 bytes (270134770 bytes left)...
Resuming download from
https://www.kaggle.com/api/v1/datasets/download/ashishsaxena2209/animal-image-
datasetdog-cat-and-panda?dataset_version_number=1 (123731968/393866738) bytes
left.
```

```
100%|██████████| 376M/376M [00:12<00:00, 21.3MB/s]
```

```
Extracting files...
```

```
Path to dataset files: C:\Users\HP\.cache\kagglehub\datasets\ashishsaxena2209\
animal-image-datasetdog-cat-and-panda\versions\1
```

```
import shutil
target_path = r"."

# Move the whole dataset folder
shutil.move(path, target_path)

print("Moved dataset to:", target_path)
```

```
Moved dataset to:  .
```

## Displaying Some Images

```
from PIL import Image

# Define image paths for each class
cat_img_path = fr"{target_path}/Animals/cats/cats_00038.jpg"

panda_img_path = fr"{target_path}/Animals/panda/panda_00034.jpg"

spider_img_path = fr"{target_path}/Animals/spiders/e830b2062dfc023ed1584d05fb1d4e9fe777ead218ac104497f5c97ca5ecb5b1_640.jpg"

# Open images
cat_img = Image.open(cat_img_path)
panda_img = Image.open(panda_img_path)
spider_img = Image.open(spider_img_path)

# Plot them side by side
plt.figure(figsize=(12, 4))

# Cat
plt.subplot(1, 3, 1)
plt.imshow(cat_img)
plt.title("Label: cat")
plt.axis('off')

# Dog
plt.subplot(1, 3, 2)
plt.imshow(panda_img)
plt.title("Label: panda")
plt.axis('off')

# Panda
plt.subplot(1, 3, 3)
plt.imshow(spider_img)
plt.title("Label: spider")
plt.axis('off')

plt.tight_layout()
plt.show()
```

## Load images features

here we load each image in each class, then prepare them and same into `.npy` files to load them quicly later on

```
from PIL import Image
from skimage.feature import hog

def extract_combined_features(image_path, image_size=(64, 64)):
    try:
        # Open the image from the given file path and resize it
        image_rgb = Image.open(image_path).resize(image_size)

        # Convert the image to grayscale for HOG feature extraction
        image_gray = image_rgb.convert('L')

        # Convert the grayscale image to a NumPy array
        gray_array = np.array(image_gray)

        # HOG (Histogram of Oriented Gradients) Features
        # Extract texture and edge information from the grayscale image
        hog_features = hog(
            gray_array,
            orientations=9,  # Number of orientation bins
            pixels_per_cell=(8, 8),  # Size of the cell in pixels
            cells_per_block=(2, 2),  # Number of cells per block
            block_norm='L2-Hys'  # Normalization method
        )

        # Convert the RGB image to a NumPy array
        rgb_array = np.array(image_rgb)

        # Color Histogram Features
        # Compute normalized histograms for R, G, B channels separately
        hist_features = []
        for i in range(3):  # Loop over Red, Green, Blue channels
            hist, _ = np.histogram(
                rgb_array[:, :, i],  # Select the color channel
                bins=32,  # Number of bins for the histogram
                range=(0, 256),  # Pixel intensity range
                density=True  # Normalize the histogram
            )
            hist_features.extend(hist)  # Add histogram data to the list

        # Combine HOG and Histogram Features
        combined = np.concatenate((hog_features, hist_features))

        return combined

    except Exception as e:
        # Print and skip any image that causes an error
        print(f"Skipping {image_path}: {e}")
        return None

import os

def load_images_with_combined_features(folder_path, image_size=(64, 64),
limit_per_class=1000):
    X, y = [], []
    label_map = {}
    current_label = 0

    for class_name in sorted(os.listdir(folder_path)): # List the three
classes
        class_path = os.path.join(folder_path, class_name) # For each folder +
class name
        if not os.path.isdir(class_path): # if the path doesn't exist, ignore
            continue

        label_map[current_label] = class_name
        count = 0

        # for all images in this class
        for filename in os.listdir(class_path):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                if count >= limit_per_class: # take the same range for all
classes
```

```
                    break
                image_path = os.path.join(class_path, filename)
                features = extract_combined_features(image_path, 
image_size=image_size)
                if features is not None:
                    X.append(features)
                    y.append(current_label)
                    count += 1

        current_label += 1

    return np.array(X), np.array(y), label_map
```

## Save the pre-processed data - Training

save the data that we processed to an appropriate extension to load them later

```
import pickle
X, y, label_map = load_images_with_combined_features(r"Animals")

np.save("X.npy", X)#save
np.save("y.npy", y)#save
with open("label_map.pkl", "wb") as f:
   pickle.dump(label_map, f)#write
```

```
Skipping Animals\panda\panda_00049.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
Skipping Animals\panda\panda_00347.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
Skipping Animals\panda\panda_00723.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
Skipping Animals\panda\panda_00781.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
Skipping Animals\panda\panda_00895.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
Skipping Animals\panda\panda_00914.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
Skipping Animals\panda\panda_00923.jpg: too many indices for array: array is 
2-dimensional, but 3 were indexed
```

## Data Set features summary

```
X = np.load("X.npy")
y = np.load("y.npy")

with open("label_map.pkl", "rb") as f:
    label_map = pickle.load(f)

print(f"Total images loaded: {len(X)}")
print(f"Shape of each image vector: {X[0].shape}")
print(f"Label distribution: {np.bincount(y)}")
print(f"Label map: {label_map}")
```

```
Total images loaded: 2993
Shape of each image vector: (1860,)
Label distribution: [1000  993 1000]
Label map: {0: 'cats', 1: 'panda', 2: 'spiders'}
```