```python
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

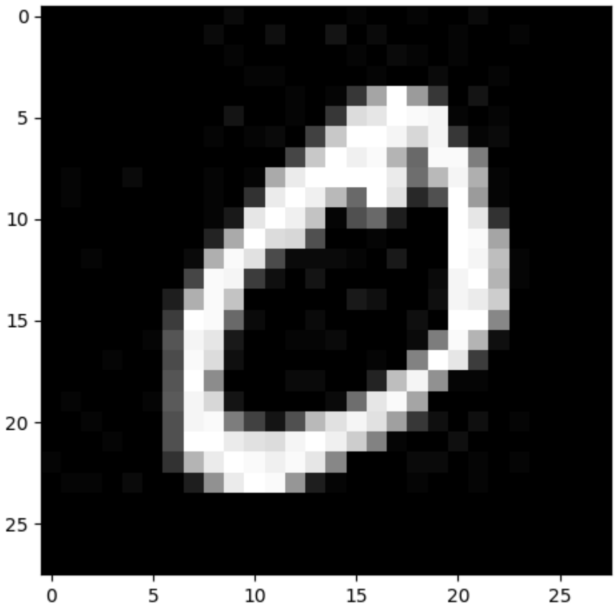
```python
from skimage.io import imread ##importing the required libraries

from skimage.color import rgb2gray
from skimage.transform import resize
from scipy.ndimage import sobel
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from tqdm import tqdm
```

```python
from skimage.io import imshow
```

```python
##reading one image and viewing
image = imread("/kaggle/input/mnist-digit/MNIST Dataset JPG format/MNIST - JPG - training/0/1.jpg")
imshow(image)
print(f"The shape of image is {image.shape}")
```

```
The shape of image is (28, 28)
<ipython-input-29-dc33dd558c6e>:3: FutureWarning: `imshow` is deprecated since version 0.25 and will be removed in version 0.27. Please use `matpl
  imshow(image)
```



```python
##Here, I am defining the function to load the images from folder and simulteniously computing hog features
## to create dataset for my ml model.
def load_dataset(dataset_path):
    features = []
    labels = []
    for label in os.listdir(dataset_path):
        label_path = os.path.join(dataset_path, label)
        if os.path.isdir(label_path):
            for image_file in tqdm(os.listdir(label_path), desc=f"Processing class {label}"):
                image_path = os.path.join(label_path, image_file)
                image = imread(image_path)
                feature = compute_hog_features(image)
                features.append(feature)
                labels.append(int(label))
    return np.array(features), np.array(labels)
```

```
##loading the train dataset from training folder
train_path="/kaggle/input/mnist-digit/MNIST Dataset JPG format/MNIST - JPG - training"
X_train, y_train = load_dataset(train_path)
```

```
Processing class 7: 100%|██████████| 6265/6265 [00:40<00:00, 153.30it/s]
Processing class 2: 100%|██████████| 5958/5958 [00:39<00:00, 152.02it/s]
Processing class 5: 100%|██████████| 5421/5421 [00:35<00:00, 152.81it/s]
Processing class 8: 100%|██████████| 5851/5851 [00:36<00:00, 159.72it/s]
Processing class 0: 100%|██████████| 5923/5923 [00:37<00:00, 157.35it/s]
Processing class 3: 100%|██████████| 6131/6131 [00:38<00:00, 157.95it/s]
Processing class 1: 100%|██████████| 6742/6742 [00:42<00:00, 157.96it/s]
Processing class 4: 100%|██████████| 5842/5842 [00:37<00:00, 157.54it/s]
Processing class 9: 100%|██████████| 5949/5949 [00:38<00:00, 156.01it/s]
Processing class 6: 100%|██████████| 5918/5918 [00:38<00:00, 155.22it/s]
```

```
##loading the dataset from test folder
test_path="/kaggle/input/mnist-digit/MNIST Dataset JPG format/MNIST - JPG - testing"
X_test,y_test = load_dataset(test_path)
```

```
Processing class 7: 100%|██████████| 1028/1028 [00:06<00:00, 158.43it/s]
Processing class 2: 100%|██████████| 1032/1032 [00:06<00:00, 150.37it/s]
Processing class 5: 100%|██████████| 892/892 [00:05<00:00, 157.53it/s]
Processing class 8: 100%|██████████| 974/974 [00:06<00:00, 149.23it/s]
Processing class 0: 100%|██████████| 980/980 [00:06<00:00, 153.40it/s]
Processing class 3: 100%|██████████| 1010/1010 [00:06<00:00, 159.59it/s]
Processing class 1: 100%|██████████| 1135/1135 [00:07<00:00, 156.09it/s]
Processing class 4: 100%|██████████| 982/982 [00:06<00:00, 155.85it/s]
Processing class 9: 100%|██████████| 1009/1009 [00:06<00:00, 153.70it/s]
Processing class 6: 100%|██████████| 958/958 [00:05<00:00, 160.17it/s]
```

```
## function to extract the hog(Histogram of Oriented Gradients) features

def compute_hog_features(image, pixels_per_cell=(8, 8), cells_per_block=(2, 2), orientations=9):
    # Convert to grayscale and resize for uniformity
    image = resize(image, (64, 64))  # Resize all images to 64x64 for consistency

    # Computing the gradients using sobel filter
    gx = sobel(image, axis=1)
    gy = sobel(image, axis=0)
    magnitude = np.sqrt(gx**2 + gy**2)      ## computing the magnitude of gradient
    orientation = (np.arctan2(gy, gx) * (180 / np.pi)) % 180   ## computing the angle(orientation) of gradient

    # Calculating the HOG features
    cell_rows, cell_cols = pixels_per_cell
    block_rows, block_cols = cells_per_block
    height, width = image.shape
    num_cells_x = width // cell_cols
    num_cells_y = height // cell_rows
    histograms = np.zeros((num_cells_y, num_cells_x, orientations))
    bin_edges = np.linspace(0, 180, orientations + 1)

    for i in range(num_cells_y):
        for j in range(num_cells_x):
            cell_magnitude = magnitude[i * cell_rows:(i + 1) * cell_rows, j * cell_cols:(j + 1) * cell_cols]
            cell_orientation = orientation[i * cell_rows:(i + 1) * cell_rows, j * cell_cols:(j + 1) * cell_cols]
            histogram, _ = np.histogram(cell_orientation, bins=bin_edges, weights=cell_magnitude)
            histograms[i, j, :] = histogram

    block_features = []
    for i in range(num_cells_y - block_rows + 1):
        for j in range(num_cells_x - block_cols + 1):
            block = histograms[i:i + block_rows, j:j + block_cols, :].ravel()
            block = block / np.sqrt(np.sum(block**2) + 1e-6)
            block_features.append(block)

    return np.concatenate(block_features)
```

```
##Loading the SVC model for classification
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

```
svm = SVC(kernel='rbf', C=1.0)
svm.fit(X_train, y_train)
```

Show hidden output

```
# Evaluating the model
y_pred = svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"The accuracy of model is: {accuracy * 100:.2f}%")
```

```
The accuracy of model is: 99.11%
```

```
from sklearn.metrics import classification_report
report=classification_report(y_test,y_pred)
```

```
print(report)
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       980
           1       0.99      1.00      0.99      1135
           2       0.99      0.99      0.99      1032
           3       0.99      0.99      0.99      1010
           4       0.99      0.99      0.99       982
           5       0.99      0.99      0.99       892
           6       0.99      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Start coding or generate with AI.