Experiment 4 - Cognitive Computing Application on Image data using Python

Roll No.	37
Name	Mikil Lalwani
Class	D20 B
Subject	Data Science Lab
LO Mapped	L2: Explore use cases of Cognitive Computing

Aim: To implement Cognitive computing for Image data using python .

Python Library Function Used:

1. Pandas-
   Pandas is a popular Python library for data manipulation and analysis. It provides powerful data structures, like DataFrames and Series, that make it easy to work with structured data. Pandas simplifies tasks such as loading data from various sources, cleaning and transforming data, and performing operations like filtering, grouping, and aggregating. It also supports handling missing data and time series data efficiently. With its versatile functionality, Pandas is widely used in data science, enabling tasks like data exploration, visualization, and preparation for machine learning. Its concise syntax and extensive documentation make it a valuable tool for anyone working with tabular data in Python.

2. Matplotlib-
   Matplotlib is a popular Python library for creating static, animated, and interactive visualizations in various formats. It provides a wide range of functions and tools for creating plots, charts, and graphs, making it a go-to choice for data visualization and scientific plotting. Matplotlib's customizable and publication-quality graphics support various plot types, including line plots, scatter plots, bar charts, histograms, and more. It offers flexibility to control every aspect of the visualization, from axis labels to colors and styles. Matplotlib can be used both as a standalone library and in conjunction with other libraries, such as NumPy and Pandas, to visualize data effectively.

3. NumPy (Numerical Python):
   NumPy is a core library for numerical and scientific computing in Python. It provides essential data structures, such as arrays and matrices, along with a collection of mathematical functions to perform operations on these arrays efficiently. NumPy is the foundation for many other scientific libraries and data analysis tools in Python. It's commonly used for tasks like data manipulation, linear algebra, statistical analysis, and numerical simulations.

4. scikit-learn (sklearn):
   Scikit-learn is a machine learning library in Python that offers a wide range of tools and algorithms for various machine learning tasks, including classification, regression, clustering, dimensionality reduction, and model selection. It provides a consistent and user-friendly interface for building and evaluating machine learning models. Scikit-learn also includes utilities for data preprocessing, feature selection, and model evaluation. It's

a valuable resource for both beginners and experienced practitioners in the field of machine learning.

5. cv2-
   cv2, also known as OpenCV (Open Source Computer Vision Library), is a popular open-source computer vision and image processing library in Python. OpenCV provides a comprehensive set of tools, functions, and algorithms for various computer vision tasks, including:
   - Image and video manipulation: OpenCV allows you to read, write, and manipulate images and videos in various formats. You can perform tasks like resizing, cropping, filtering, and color space conversion.
   - Object detection and tracking: OpenCV offers pre-trained models and algorithms for object detection, face recognition, and tracking objects in video streams.
   - Feature extraction: It provides tools for extracting features from images, such as keypoints and descriptors, which are essential for tasks like image matching and recognition.
   - Image processing: OpenCV includes a wide range of image processing functions, including edge detection, image smoothing, morphological operations, and more.
   - Computer vision algorithms: It contains implementations of various computer vision algorithms like camera calibration, stereo vision, optical flow, and 3D reconstruction.
   - Machine learning integration: OpenCV can be integrated with machine learning libraries like scikit-learn to create complex computer vision and machine learning pipelines.

Code and Observation:

1. Import the required libraries.

Kidney Stone prediction using MRI

Dataset Link - https://www.kaggle.com/datasets/mohammedrizwanmalik/kidney-stones-mri-and-ct-scans

[1]:
```
import cv2
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

2. Import filters to images for different views.

Filters for image

[2]:
```
stone = cv2.imread('./Dataset/Train/Kidney_stone/1.3
stone = cv2.resize(stone, (500,500))
cv2.imshow('Stone', stone)
cv2.waitKey()
```

[2]: -1

[3]:
```
laplace_img = cv2.Laplacian(stone,-5)
cv2.imshow('Laplace', laplace_img)
cv2.waitKey()
```

[3]: -1

[4]:
```
from skimage.filters import sobel
sobel_img = sobel(stone)
cv2.imshow('Sobel', sobel_img)
cv2.waitKey()
```

[4]: -1

3. Find the minimum dimension from all the pictures for cropping all the images.

Reading the dataset

[5]:
```
stone_absent = os.listdir('./Dataset/Train/Normal/')
stone_present = os.listdir('./Dataset/Train/Kidney_stone/')

min_width = 10000
min_height = 10000

for i in stone_absent:
    s = cv2.imread('./Dataset/Train/Normal/'+i).shape
    min_width = min_width if min_width<s[0] else s[0]
    min_height = min_height if min_height<s[1] else s[1]
print(min_width,min_height)
```

```
702 614
```

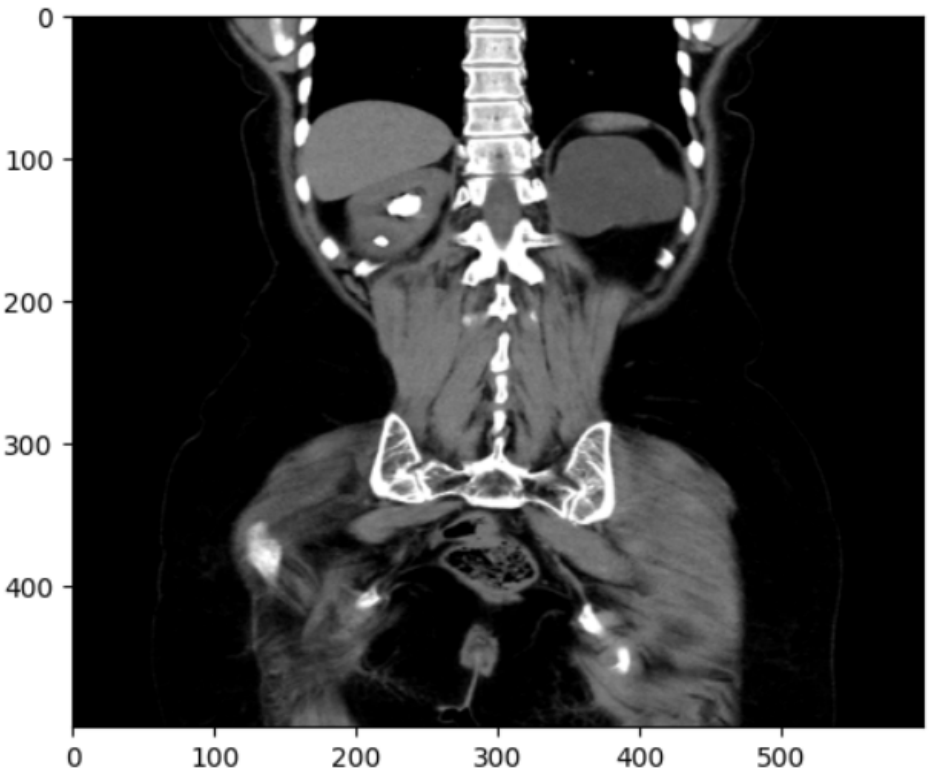
4. Read the dataset.

[6]:
```
stone_absent = os.listdir('./Dataset/Train/Normal/')
stone_present = os.listdir('./Dataset/Train/Kidney_stone/')

X = []
y = []

for i in stone_absent:
    gray = cv2.cvtColor(cv2.resize(cv2.imread('./Dataset/Train/Normal/'+i),(600,500)),cv2.COLOR_BGR2GRAY)
    X.append(gray)
    y.append(0)

for i in stone_present:
    gray = cv2.cvtColor(cv2.resize(cv2.imread('./Dataset/Train/Kidney_stone/'+i),(600,500)),cv2.COLOR_BGR2GRAY)
    X.append(gray)
    y.append(1)
```

[7]:
```
X = np.array(X)
y = np.array(y)
X.shape
```

[7]:
```
(1453, 500, 600)
```

[8]:
```
plt.imshow(X[1000], cmap='gray')
plt.show()
```



```
[9]: X = X.reshape(len(X),-1)
     X.shape

[9]: (1453, 300000)

[10]: pd.Series(y).value_counts()

[10]: 0    828
      1    625
      Name: count, dtype: int64
```

5. Train test split.

Train Test Split

```
[11]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
      print(X_train.shape, X_test.shape)

      (1162, 300000) (291, 300000)

[12]: print(X_train.max(), X_test.min())

      255 0

[13]: X_train = X_train/X_train.max()
      X_test = X_test/X_test.max()
      print(X_train.max(), X_test.min())

      1.0 0.0
```

6. Logistic regression(Number of iterations = 100)

Logistic Regression

```
[14]: log = LogisticRegression()
      log.fit(X_train,y_train)

      C:\Users\Mikil\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
      erge (status=1):
      STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

      Increase the number of iterations (max_iter) or scale the data as shown ir
          https://scikit-learn.org/stable/modules/preprocessing.html
      Please also refer to the documentation for alternative solver options:
          https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
        n_iter_i = _check_optimize_result(

[14]: ▾ LogisticRegression
      LogisticRegression()

[15]: print('Training score(Logistic Regression):',log.score(X_train,y_train))
      print('Testing score(Logistic Regression):',log.score(X_test,y_test))

      Training score(Logistic Regression): 1.0
      Testing score(Logistic Regression): 0.8865979381443299
```

7. Logistic regression(Number of iterations = 300)

Logistic Regression with higher numberr of iterations

[16]:
```
log = LogisticRegression(max_iter=300)
log.fit(X_train,y_train)
print('Training score(Logistic Regression):',log.score(X_train,y_train))
print('Testing score(Logistic Regression):',log.score(X_test,y_test))
```

```
C:\Users\Mikil\AppData\Local\Programs\Python\Python311\Lib\site-packages\s
erge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown ir
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-reg
  n_iter_i = _check_optimize_result(
Training score(Logistic Regression): 1.0
Testing score(Logistic Regression): 0.8762886597938144
```

8. SVM

SVM

[17]:
```
svc = SVC()
svc.fit(X_train,y_train)
```

[17]:
```
▾ SVC
SVC()
```

[18]:
```
test_score = svc.score(X_test,y_test)
```

[18]:
```
0.865979381443299
```

[19]:
```
print('Training score(SVM):',svc.score(X_train,y_train))
print('Testing score(SVM):',test_score)
```

```
Training score(SVM): 0.9500860585197934
Testing score(SVM): 0.865979381443299
```

9. Test Data

Test Data

[20]:
```
stone_absent = os.listdir('./Dataset/Test/Normal/')
stone_present = os.listdir('./Dataset/Test/Kidney_stone/')

test_X = []
test_y = []

for i in stone_absent:
    gray = cv2.cvtColor(cv2.resize(cv2.imread('./Dataset/Test/Normal/'+i),(600,500)),cv2.COLOR_BGR2GRAY)
    test_X.append(gray)
    test_y.append(0)

for i in stone_present:
    gray = cv2.cvtColor(cv2.resize(cv2.imread('./Dataset/Test/Kidney_stone/'+i),(600,500)),cv2.COLOR_BGR2GRAY)
    test_X.append(gray)
    test_y.append(1)
```

10. Accuracy of both the models.

[21]:
```
test_X = np.array(test_X)
test_y = np.array(test_y)
test_X.shape
```

[21]:
```
(346, 500, 600)
```

[22]:
```
test_X = test_X.reshape(len(test_X),-1)
test_X.shape
```

[22]:
```
(346, 300000)
```

[23]:
```
pd.Series(test_y).value_counts()
```

[23]:
```
0    181
1    165
Name: count, dtype: int64
```

[24]:
```
print('Testing score(Logistic Regression):',log.score(test_X,test_y))
```

```
Testing score(Logistic Regression): 0.5317919075144508
```

[25]:
```
print('Testing score(SVM):',svc.score(test_X,test_y))
```

```
Testing score(Logistic Regression): 0.523121387283237
```

Conclusion:

Thus we have implemented Cognitive computing for Image data using python.