image-classification-svm

May 31, 2024

[46]:
```
#Drive Mount
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[47]:
```
#Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from skimage.io import imread
from sklearn.model_selection import train_test_split
from skimage.transform import resize
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
```

[48]:
```
os.listdir('/content/drive/MyDrive/Bloodcells')
```

[48]:
```
['NEUTROPHIL', 'LYMPHOCYTE', 'EOSINOPHIL', 'MONOCYTE']
```

[49]:
```
len(os.listdir('/content/drive/MyDrive/Bloodcells/EOSINOPHIL'))
```

[49]:
```
623
```

[50]:
```
len(os.listdir('/content/drive/MyDrive/Bloodcells/LYMPHOCYTE'))
```

[50]:
```
620
```

[51]:
```
len(os.listdir('/content/drive/MyDrive/Bloodcells/MONOCYTE'))
```

[51]:
```
620
```

[52]:
```
len(os.listdir('/content/drive/MyDrive/Bloodcells/NEUTROPHIL'))
```

[52]:
```
624
```

[ ]:
```python
Eosinophil_path='/content/drive/MyDrive/Bloodcells/EOSINOPHIL'
for img in os.listdir(Eosinophil_path):
  print(img)
```

[ ]:
```python
Lymphocyte_path='/content/drive/MyDrive/Bloodcells/LYMPHOCYTE'
for img in os.listdir(Lymphocyte_path):
  print(img)
```

[ ]:
```python
Monocyte_path='/content/drive/MyDrive/Bloodcells/MONOCYTE'
for img in os.listdir(Monocyte_path):
  print(img)
```

[ ]:
```python
Neutrophil_path='/content/drive/MyDrive/Bloodcells/NEUTROPHIL'
for img in os.listdir(Neutrophil_path):
  print(img)
```

[57]:
```python
input=[]
target=[]
categories=['NEUTROPHIL', 'LYMPHOCYTE', 'EOSINOPHIL', 'MONOCYTE']
path1='/content/drive/MyDrive/Bloodcells'
```

[58]:
```python
for i in categories:
  path=os.path.join(path1,i)
  for img in os.listdir(path):
    img_arr=imread(os.path.join(path,img))
    img_resize=resize(img_arr,(150,150,3))
    input.append(img_resize.flatten())
    target.append(categories.index(i))
  print("Loaded completely",i)
```

```
Loaded completely NEUTROPHIL
Loaded completely LYMPHOCYTE
Loaded completely EOSINOPHIL
Loaded completely MONOCYTE
```

[59]:
```python
print(target)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
```

```
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
```

```
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,
3, 3, 3, 3]
```

[60]:
```python
input_data=np.array(input)
target_data=np.array(target)
df=pd.DataFrame(input_data)
df.shape
```

[60]: (2487, 67500)

[62]:
```python
df['Target']=target_data
df.shape
```

[62]: (2487, 67501)

[63]:
```
x=df.iloc[:,:-1]
x
```

[63]:

	0	1	2	3	4	5	6	\
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
4	0.008805	0.008805	0.008805	0.011868	0.011868	0.011868	0.016698	
…	…	…	…	…	…	…	…	
2482	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
2483	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
2484	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
2485	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
2486	0.266655	0.225064	0.225045	0.264318	0.221166	0.219620	0.267857	

	7	8	9	…	67490	67491	67492	\
0	0.000000	0.000000	0.000000	…	0.000002	0.000000	0.003433	
1	0.000000	0.000000	0.000000	…	0.000000	0.000000	0.000000	
2	0.000000	0.000000	0.000000	…	0.000000	0.000000	0.000000	
3	0.000000	0.000000	0.000000	…	0.717306	0.016006	0.016006	
4	0.016698	0.016688	0.661372	…	0.000000	0.000000	0.000000	
…	…	…	…	…	…	…		
2482	0.000000	0.000000	0.000016	…	0.756600	0.765451	0.743484	
2483	0.000000	0.000000	0.000000	…	0.819198	0.789337	0.747283	
2484	0.000000	0.000000	0.000000	…	0.636063	0.777747	0.664895	
2485	0.000000	0.000000	0.000000	…	0.000000	0.000000	0.000000	
2486	0.215475	0.212498	0.252179	…	0.000000	0.000000	0.000000	

	67493	67494	67495	67496	67497	67498	67499
0	0.000000	0.000000	0.003922	0.000000	0.000000	0.003922	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
3	0.015996	0.003744	0.003744	0.003744	0.014652	0.014652	0.014652
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
…	…	…	…	…	…	…	…
2482	0.760579	0.096240	0.086886	0.093805	0.009272	0.024069	0.007588
2483	0.820706	0.778812	0.770430	0.801632	0.780152	0.788410	0.799763
2484	0.641582	0.758928	0.641512	0.640300	0.739475	0.618243	0.638352
2485	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2486	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

[2487 rows x 67500 columns]

[64]:
```
y=df.iloc[:,-1]
y
```

[64]:
```
0       0
1       0
2       0
3       0
4       0
       ..
2482    3
2483    3
2484    3
2485    3
2486    3
Name: Target, Length: 2487, dtype: int64
```

[65]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪30,random_state=42)
```

[66]:
```
from sklearn.svm import SVC
model=SVC()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
y_pred
```

[66]:
```
array([0, 0, 2, 2, 1, 0, 2, 3, 1, 3, 1, 1, 1, 0, 2, 2, 0, 1, 3, 2, 1, 2,
       0, 3, 2, 1, 3, 1, 1, 2, 1, 3, 0, 2, 2, 2, 3, 2, 3, 0, 1, 3, 1, 1,
       0, 1, 3, 0, 2, 0, 2, 2, 3, 2, 3, 0, 0, 2, 1, 3, 2, 2, 0, 0, 3, 0,
       1, 0, 2, 3, 1, 3, 0, 1, 3, 1, 1, 3, 3, 1, 1, 2, 1, 3, 0, 1, 3, 2,
       1, 0, 1, 3, 1, 1, 3, 2, 0, 0, 3, 3, 2, 2, 1, 3, 1, 1, 3, 1, 1, 2,
       3, 1, 3, 3, 2, 1, 1, 3, 1, 3, 3, 3, 3, 0, 3, 3, 3, 0, 1, 3, 3, 2,
       3, 2, 0, 1, 3, 1, 1, 3, 1, 2, 3, 1, 1, 3, 2, 1, 2, 1, 2, 3, 2, 3,
       2, 0, 3, 1, 3, 3, 0, 0, 0, 0, 2, 1, 2, 0, 3, 0, 0, 0, 2, 1, 1, 0,
       3, 1, 1, 2, 3, 3, 1, 3, 3, 3, 3, 1, 3, 2, 3, 3, 2, 3, 3, 0, 3, 2,
       3, 3, 1, 1, 0, 3, 1, 1, 1, 1, 3, 1, 2, 0, 3, 3, 2, 0, 1, 3, 2, 3,
       3, 3, 0, 1, 0, 2, 1, 1, 3, 3, 2, 1, 1, 1, 2, 1, 2, 2, 0, 0, 0, 1,
       3, 1, 3, 2, 1, 3, 1, 3, 0, 1, 3, 1, 2, 3, 3, 3, 1, 0, 3, 2, 1, 2,
       3, 2, 0, 0, 2, 1, 3, 3, 1, 3, 2, 2, 3, 2, 3, 1, 2, 0, 2, 3, 3, 3,
       3, 3, 2, 1, 1, 3, 0, 0, 3, 3, 3, 1, 1, 0, 2, 0, 3, 3, 3, 3, 0, 3,
       3, 3, 0, 2, 2, 1, 2, 2, 1, 0, 0, 3, 0, 1, 3, 0, 2, 0, 0, 0, 2, 1,
       0, 3, 1, 0, 2, 3, 0, 3, 3, 0, 3, 3, 3, 3, 1, 2, 0, 3, 2, 3, 1, 3,
       0, 3, 0, 1, 1, 0, 1, 1, 2, 3, 2, 3, 2, 3, 3, 2, 0, 1, 1, 3, 3, 1,
       1, 3, 2, 3, 3, 2, 3, 1, 2, 0, 0, 1, 1, 3, 2, 2, 2, 0, 1, 3, 3, 0,
       3, 3, 1, 3, 2, 0, 0, 2, 3, 3, 0, 1, 2, 2, 2, 2, 3, 2, 3, 1, 3, 0,
       2, 0, 2, 3, 3, 1, 2, 3, 2, 3, 1, 3, 3, 0, 1, 1, 3, 2, 3, 0, 0, 2,
       0, 2, 2, 3, 0, 3, 2, 1, 3, 2, 1, 3, 3, 1, 2, 1, 3, 2, 2, 3, 2, 2,
       3, 3, 2, 3, 3, 0, 1, 0, 2, 1, 3, 2, 3, 2, 3, 2, 2, 3, 3, 3, 0, 0,
       0, 3, 2, 2, 2, 1, 3, 0, 2, 3, 0, 3, 2, 3, 3, 0, 3, 2, 1, 3, 1, 1,
       3, 3, 3, 0, 3, 3, 0, 0, 0, 3, 1, 2, 3, 3, 2, 3, 3, 3, 3, 3, 0, 0,
       3, 0, 2, 3, 0, 3, 0, 0, 3, 2, 3, 1, 0, 0, 1, 1, 3, 2, 3, 3, 0, 3,
```

```
0, 0, 2, 3, 0, 2, 3, 0, 3, 3, 0, 0, 1, 2, 1, 3, 2, 3, 3, 1, 1, 0,
1, 1, 1, 0, 2, 3, 2, 1, 3, 3, 0, 3, 0, 3, 1, 2, 1, 0, 2, 1, 0, 2,
0, 2, 2, 2, 2, 0, 1, 1, 0, 0, 1, 1, 1, 2, 1, 1, 1, 1, 0, 1, 1, 0,
1, 0, 3, 0, 3, 0, 1, 3, 1, 3, 2, 0, 2, 1, 3, 3, 3, 3, 2, 2, 0, 0,
0, 3, 1, 1, 2, 1, 2, 3, 0, 1, 1, 2, 1, 3, 2, 3, 1, 0, 3, 2, 0, 2,
2, 2, 1, 0, 0, 3, 2, 1, 1, 1, 3, 0, 2, 1, 1, 0, 1, 3, 0, 0, 1, 2,
0, 3, 3, 0, 1, 3, 1, 1, 1, 2, 2, 1, 1, 3, 0, 3, 3, 3, 2, 0, 0, 3,
1, 0, 1, 1, 3, 2, 1, 0, 3, 1, 1, 0, 0, 3, 1, 3, 0, 1, 3, 0, 0, 2,
3, 2, 1, 0, 1, 0, 3, 1, 2, 3, 1, 2, 3, 3, 1, 1, 1, 2, 2, 1, 0])
```

[ ]:
```
from sklearn.metrics import accuracy_score
print("Accuracy score is",accuracy_score(y_test,y_pred))
```

[68]:
```
#To make predictions for a new image of Eosinophil
path1='/content/drive/MyDrive/eosinophils.jpg'#assign path
img=imread(path1)#read image
img=resize(img,(150,150,3)).flatten().reshape(1,-1)#resize the image
model.predict(img)
```

[68]: array([3])

[69]:
```
#To make predictions for a new image of Lymphocyte
path2='/content/drive/MyDrive/lymphocyte.jpg'#assign path
img=imread(path2)#read image
img=resize(img,(150,150,3)).flatten().reshape(1,-1)#resize the image
model.predict(img)
```

[69]: array([0])

[70]:
```
#To make predictions for a new image of Monocyte
path3='/content/drive/MyDrive/monocyte.jpg'#assign path
img=imread(path3)#read image
img=resize(img,(150,150,3)).flatten().reshape(1,-1)#resize the image
model.predict(img)
```

[70]: array([3])

[71]:
```
#To make predictions for a new image of Neutrophil
path4='/content/drive/MyDrive/neutrophil.jpg'#assign path
img=imread(path4)#read image
img=resize(img,(150,150,3)).flatten().reshape(1,-1)#resize the image
model.predict(img)
```

[71]: array([3])