In [1]:

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier

```

In [2]:

```
dataset=[]
```

In [3]:

```
folder_paths=['C:\\Users\\usama\\Desktop\\archive\\fruits-360_dataset\\fruits-360\\Tr
             'C:\\Users\\usama\\Desktop\\archive\\fruits-360_dataset\\fruits-360\\Tra
             'C:\\Users\\usama\\Desktop\\archive\\fruits-360_dataset\\fruits-360\\Tra
             'C:\\Users\\usama\\Desktop\\archive\\fruits-360_dataset\\fruits-360\\Tra
             'C:\\Users\\usama\\Desktop\\archive\\fruits-360_dataset\\fruits-360\\Tra
             'C:\\Users\\usama\\Desktop\\archive\\fruits-360_dataset\\fruits-360\\Tra
             ]

```

In [4]:

```
# Iterate over the folder paths
for i in folder_paths:
    folder_name = os.path.basename(i)

    # Iterate over the images in the subdirectory
    for file_name in os.listdir(i):
        image_path = os.path.join(i, file_name)

        if os.path.isfile(image_path):  # Only consider files
            # Load the image using OpenCV
            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

            # If the image was successfully loaded
            if image is not None:
                # Resize the grayscale image to 250X250 pixels
                resized_image = cv2.resize(image, (250, 250))

                # Flatten the image and append each pixel as a separate feature alon
                flattened_image = resized_image.flatten().tolist()
                dataset.append(flattened_image + [folder_name])
```

In [5]:

```
"""Convert the dataset to a pandas DataFrame"""
df = pd.DataFrame(dataset, columns=[f'pixel_{i+1}' for i in range(250*250)] + ['labe

"""Print the DataFrame"""
df.head()
```

Out[5]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	...
0	255	255	255	255	255	255	255	255	255	255	...
1	255	255	255	255	255	254	254	254	254	254	...
2	254	254	254	255	255	255	255	254	254	254	...
3	255	255	255	255	255	254	254	254	254	254	...
4	255	255	254	254	254	254	254	253	253	253	...

5 rows × 62501 columns

In [6]:

```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix
```

In [7]:

```
X=df.drop('label',axis=1)
X=X/255
X
```

Out[7]:

	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_
0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
1	1.000000	1.000000	1.000000	1.000000	1.000000	0.996078	0.996078	0.996078	0.99607
2	0.996078	0.996078	0.996078	1.000000	1.000000	1.000000	1.000000	0.996078	0.99607
3	1.000000	1.000000	1.000000	1.000000	1.000000	0.996078	0.996078	0.996078	0.99607
4	1.000000	1.000000	0.996078	0.996078	0.996078	0.996078	0.996078	0.992157	0.99215
...	...	...	...	...	...	...	...	...	
2864	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
2865	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
2866	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
2867	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
2868	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000

2869 rows × 62500 columns

In [8]:

```
y=df.label
```

In [9]:

```
label_count= y.value_counts()
label_count
```

Out[9]:

```
Apple Golden 1        960
Apple Braeburn        492
Apple Granny Smith    492
Apple Golden 3        481
Apple Crimson Snow    444
Name: label, dtype: int64
```

In [10]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
Y_encoded = label_encoder.fit_transform(y)
y_series=pd.Series(Y_encoded)
y_series
```

Out[10]:

```
0       0
1       0
2       0
3       0
4       0
       ..
2864    4
2865    4
2866    4
2867    4
2868    4
Length: 2869, dtype: int32
```

In [11]:

```
clf=MLPClassifier(hidden_layer_sizes=(100,),
    activation='relu')
```

In [12]:

```
x_train,x_test,y_train,y_test=train_test_split(X,y_series,test_size=0.2,random_state

```

In [13]:

```
clf.fit(x_train,y_train);
```

In [14]:

```
clf.score(x_test,y_test)
```

Out[14]:

```
1.0
```

In [15]:

```
y_preds=clf.predict(x_test)
result=pd.DataFrame({'y_test':y_test,'y_preds':y_preds})
result.head()
```

Out[15]:

	y_test	y_preds
2182	3	3
2733	4	4
1716	2	2
588	1	1
2117	3	3

In [16]:

```
conf_matrix = confusion_matrix(y_test, y_preds)
print("Confusion Matrix:")
print(conf_matrix)
```

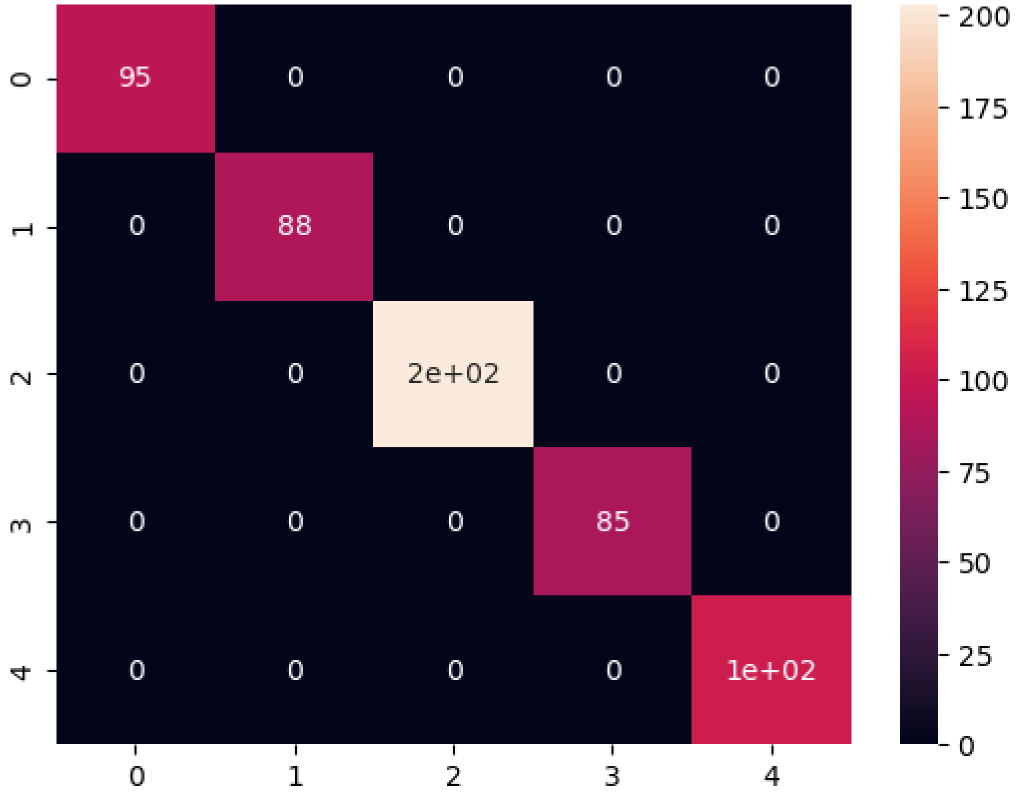
```
Confusion Matrix:
[[ 95   0   0   0   0]
 [  0  88   0   0   0]
 [  0   0 203   0   0]
 [  0   0   0  85   0]
 [  0   0   0   0 103]]
```

In [17]:

```
import seaborn as sns
sns.heatmap(conf_matrix,annot=True)
```

Out[17]:

```
<Axes: >
```



In [ ]:

```
 
```

In [ ]:

```
 
```