```
unzip:  cannot find or open /content/drive/MyDrive/Document from Sam.zip, /content/drive/MyDrive/Document from Sam
```

```
import os
len(os.listdir("/content/drive/MyDrive/fruits/fruits-360-original-size/fruits-360-original-size/Training"))
```

```
import cv2 as cv
import os
import pandas as pd
```

```
dataset=[]
folder_paths=["/content/drive/MyDrive/frnds/aoun",
            "/content/drive/MyDrive/frnds/billo",
             "/content/drive/MyDrive/frnds/usama"]
for i in folder_paths:
  folder_name=os.path.basename(i)
  for file_name in os.listdir(i):
    img_path=os.path.join(i,file_name)
    if os.path.isfile(img_path):
      img=cv.imread(img_path, cv.IMREAD_GRAYSCALE)
      if img is not None:
        resize_img=cv.resize(img,(250,250))
        flattened_img=resize_img.flatten().tolist()
        dataset.append(flattened_img+[folder_name])
```

```
df = pd.DataFrame(dataset)
```

```
df.head(10)
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 62491 | 62492 | 62493 | 6249 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 219 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | ... | 106 | 114 | 102 | 18 |
| **1** | 143 | 142 | 138 | 132 | 129 | 127 | 125 | 125 | 130 | 130 | ... | 121 | 129 | 131 | 13 |
| **2** | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | ... | 128 | 128 | 130 | 14 |
| **3** | 220 | 217 | 213 | 209 | 207 | 199 | 197 | 191 | 186 | 179 | ... | 144 | 144 | 144 | 15 |
| **4** | 227 | 222 | 223 | 223 | 219 | 217 | 215 | 210 | 205 | 200 | ... | 199 | 151 | 147 | 14 |
| **5** | 229 | 229 | 229 | 228 | 228 | 228 | 228 | 228 | 227 | 227 | ... | 180 | 135 | 145 | 14 |
| **6** | 220 | 220 | 220 | 219 | 219 | 219 | 219 | 219 | 220 | 217 | ... | 149 | 150 | 152 | 15 |
| **7** | 170 | 170 | 170 | 169 | 169 | 169 | 169 | 170 | 169 | 169 | ... | 96 | 92 | 142 | 16 |
| **8** | 150 | 150 | 150 | 150 | 151 | 151 | 151 | 152 | 150 | 143 | ... | 78 | 74 | 75 | 7 |
| **9** | 116 | 118 | 117 | 117 | 122 | 129 | 134 | 139 | 143 | 144 | ... | 72 | 100 | 126 | 14 |

10 rows × 62501 columns

```
df.rename(columns={df.iloc[:,-1].name:'label'},inplace=True)
df.head()
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | ... | 62491 | 62492 | 62493 | 6249 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | 219 | ... | 106 | 114 | 102 | 18 |
| 1 | 143 | 142 | 138 | 132 | 129 | 127 | 125 | 125 | 130 | 130 | ... | 121 | 129 | 131 | 13 |
| 2 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | 228 | ... | 128 | 128 | 130 | 14 |
| 3 | 220 | 217 | 213 | 209 | 207 | 199 | 197 | 191 | 186 | 179 | ... | 144 | 144 | 144 | 15 |
| 4 | 227 | 222 | 223 | 223 | 219 | 217 | 215 | 210 | 205 | 200 | ... | 199 | 151 | 147 | 14 |

5 rows × 62501 columns

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder
```

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

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.8 |
| 1 | 0.560784 | 0.556863 | 0.541176 | 0.517647 | 0.505882 | 0.498039 | 0.490196 | 0.490196 | 0.5 |
| 2 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.8 |
| 3 | 0.862745 | 0.850980 | 0.835294 | 0.819608 | 0.811765 | 0.780392 | 0.772549 | 0.749020 | 0.7 |
| 4 | 0.890196 | 0.870588 | 0.874510 | 0.874510 | 0.858824 | 0.850980 | 0.843137 | 0.823529 | 0.8 |

5 rows × 62500 columns

```
encoder=LabelEncoder()
y=df.label
y_encoded=encoder.fit_transform(y)
y_series=pd.Series(y_encoded,name='target')
```

```
df_encoded=pd.concat([x,y_series],axis=1)
df_encoded.head()
```

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.858824 | 0.8 |
| **2** | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.894118 | 0.8 |

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y_encoded,test_size=0.2,random_state=42)
```

```
from keras.models import Sequential
from keras.layers import Dense, Flatten
```

```
from keras.models.cloning import sequential
model=Sequential()
```

```
model.add(Dense(128,activation='relu',input_shape=(250*250,)))
model.add(Dense(128,activation='relu'))
#output layer
model.add(Dense(len(encoder.classes_),activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 128)               8000128

 dense_9 (Dense)             (None, 64)                8256

 dense_10 (Dense)            (None, 3)                 195

 dense_11 (Dense)            (None, 128)               512

 dense_12 (Dense)            (None, 64)                8256

 dense_13 (Dense)            (None, 3)                 195

 dense_14 (Dense)            (None, 128)               512

 dense_15 (Dense)            (None, 128)               16512

 dense_16 (Dense)            (None, 3)                 387

=================================================================
Total params: 8,034,953
Trainable params: 8,034,953
Non-trainable params: 0
_________________________________________________________________
```

```
from keras.utils import to_categorical
y_train_encoded=to_categorical(y_train)
y_test_encoded=to_categorical(y_test)
```

```
history = model.fit(x_train, y_train_encoded, batch_size=32, epochs=100, validation_split=0.2)
```

```
Epoch 74/100
2/2 [==============================] - 0s 44ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0736 - val_a
Epoch 75/100
2/2 [==============================] - 0s 42ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0735 - val_a
Epoch 76/100
2/2 [==============================] - 0s 47ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0726 - val_a
Epoch 77/100
2/2 [==============================] - 0s 42ms/step - loss: 1.0620 - accuracy: 0.4600 - val_loss: 1.0710 - val_a
Epoch 78/100
2/2 [==============================] - 0s 45ms/step - loss: 1.0616 - accuracy: 0.4600 - val_loss: 1.0691 - val_a
Epoch 79/100
2/2 [==============================] - 0s 43ms/step - loss: 1.0617 - accuracy: 0.4600 - val_loss: 1.0671 - val_a
Epoch 80/100
2/2 [==============================] - 0s 56ms/step - loss: 1.0621 - accuracy: 0.4600 - val_loss: 1.0660 - val_a
Epoch 81/100
2/2 [==============================] - 0s 41ms/step - loss: 1.0628 - accuracy: 0.4600 - val_loss: 1.0664 - val_a
Epoch 82/100
2/2 [==============================] - 0s 44ms/step - loss: 1.0616 - accuracy: 0.4600 - val_loss: 1.0671 - val_a
Epoch 83/100
2/2 [==============================] - 0s 43ms/step - loss: 1.0614 - accuracy: 0.4600 - val_loss: 1.0683 - val_a
Epoch 84/100
2/2 [==============================] - 0s 46ms/step - loss: 1.0617 - accuracy: 0.4600 - val_loss: 1.0701 - val_a
Epoch 85/100
2/2 [==============================] - 0s 50ms/step - loss: 1.0614 - accuracy: 0.4600 - val_loss: 1.0714 - val_a
Epoch 86/100
2/2 [==============================] - 0s 42ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0728 - val_a
Epoch 87/100
2/2 [==============================] - 0s 56ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0743 - val_a
Epoch 88/100
2/2 [==============================] - 0s 51ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0752 - val_a
Epoch 89/100
2/2 [==============================] - 0s 46ms/step - loss: 1.0611 - accuracy: 0.4600 - val_loss: 1.0757 - val_a
Epoch 90/100
2/2 [==============================] - 0s 42ms/step - loss: 1.0612 - accuracy: 0.4600 - val_loss: 1.0772 - val_a
Epoch 91/100
2/2 [==============================] - 0s 41ms/step - loss: 1.0611 - accuracy: 0.4600 - val_loss: 1.0775 - val_a
Epoch 92/100
2/2 [==============================] - 0s 46ms/step - loss: 1.0608 - accuracy: 0.4600 - val_loss: 1.0786 - val_a
Epoch 93/100
2/2 [==============================] - 0s 43ms/step - loss: 1.0621 - accuracy: 0.4600 - val_loss: 1.0797 - val_a
Epoch 94/100
2/2 [==============================] - 0s 43ms/step - loss: 1.0614 - accuracy: 0.4600 - val_loss: 1.0794 - val_a
Epoch 95/100
2/2 [==============================] - 0s 44ms/step - loss: 1.0612 - accuracy: 0.4600 - val_loss: 1.0791 - val_a
Epoch 96/100
2/2 [==============================] - 0s 42ms/step - loss: 1.0620 - accuracy: 0.4600 - val_loss: 1.0778 - val_a
Epoch 97/100
2/2 [==============================] - 0s 44ms/step - loss: 1.0610 - accuracy: 0.4600 - val_loss: 1.0777 - val_a
Epoch 98/100
2/2 [==============================] - 0s 43ms/step - loss: 1.0611 - accuracy: 0.4600 - val_loss: 1.0770 - val_a
Epoch 99/100
2/2 [==============================] - 0s 44ms/step - loss: 1.0613 - accuracy: 0.4600 - val_loss: 1.0771 - val_a
Epoch 100/100
2/2 [==============================] - 0s 48ms/step - loss: 1.0612 - accuracy: 0.4600 - val_loss: 1.0758 - val_a
```

## prediction

```
y_pred_encoded=model.predict(x_test)
```

```
1/1 [==============================] - 0s 96ms/step
```

```
import numpy as np
y_pred=np.argmax(y_pred_encoded,axis=1)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

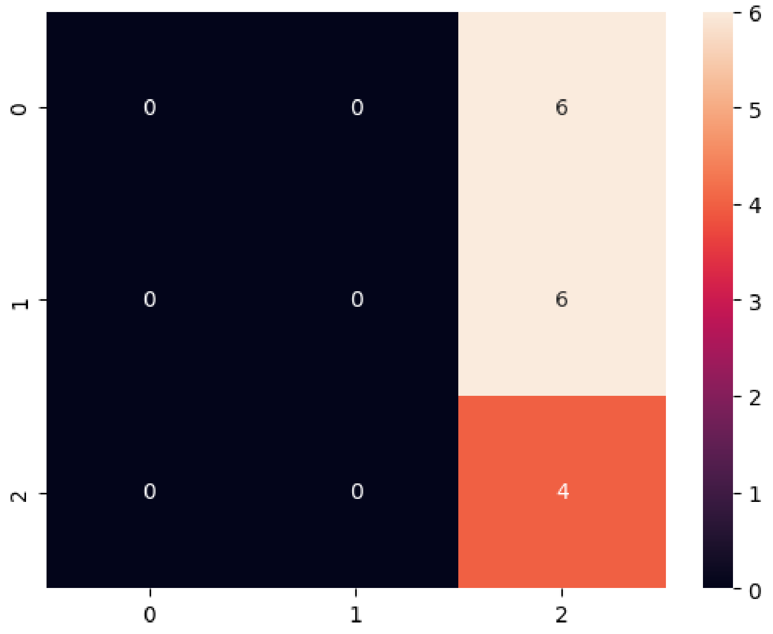
```
Accuracy: 0.25
```

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

```
Confusion Matrix:
[[0 0 6]
 [0 0 6]
 [0 0 4]]
```

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

```
<Axes: >
```



## DeployMent

```
model.save('frnds.h5')
```

```
import cv2 as cv
labels = {i: label for i, label in enumerate(encoder.classes_)}
img=cv.imread("/content/drive/MyDrive/frnds/usama/WhatsApp Image 2023-07-26 at 6.40.40 PM (1).jpeg",cv.IMREAD_GRAYSCALE)
resized_img=cv.resize(img,(250,250))
flattened_img1=resized_img.flatten().tolist()
y_preprocessed=pd.DataFrame(np.array([flattened_img1]))
op=model.predict(y_preprocessed)
arg_max=op.argmax()
predicted_value=labels[arg_max]
print('this is',predicted_value)
```

```
WARNING:tensorflow:5 out of the last 15 calls to <function Model.make_predict_function.<locals>.predict_function a
1/1 [==============================] - 0s 136ms/step
this is usama
```

✓ 0s completed at 9:38 PM