EXPT NO: 6 **A python program to do face recognition using**

DATE: 27.09.2024 **SVM Classifier**

AIM:

To write a python program to implement face recognition using the SVM Classifier

PROCEDURE:

Implementing face recognition using the SVM Classifier using the cat and dog dataset involve the following steps:

Step 1: Import Necessary Libraries

First, import the libraries that are essential for data manipulation, visualization, and model building.

```
import pandas as pd
import imageio
import os
from skimage.transform import resize
from skimage.io import imread
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

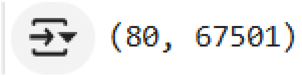
Step 2: Load theDog and cat Dataset

The dog and cat dataset can be loaded.

```
Categories=['cats','dogs']
```

```
flat_data_arr=[] #input array
target_arr=[] #output array
datadir='/content/images'
#path which contains all the categories of images
for i in Categories:

  print(f'loading... category : {i}')
  path=os.path.join(datadir,i)
  for img in os.listdir(path):
    img_array=imread(os.path.join(path,img))
    img_resized=resize(img_array,(150,150,3))
    flat_data_arr.append(img_resized.flatten())
    target_arr.append(Categories.index(i))
  print(f'loaded category:{i} successfully')
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)
#dataframe
df=pd.DataFrame(flat_data)
df['Target']=target
df.shape
```

OUTPUT :



```
(80, 67501)
```

Step 3: Separate input features and targets.

```
#input data
x=df.iloc[:,:-1]
#output data
y=df.iloc[:,-1]
```

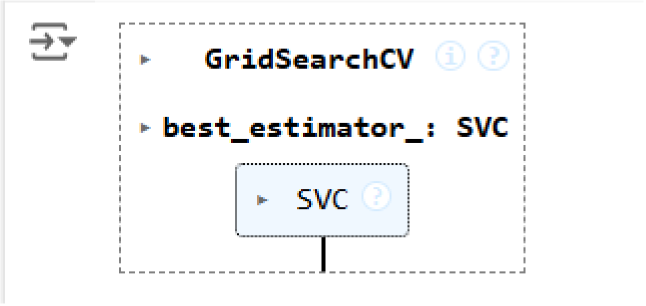
Step 4 : Separate the input features and target

```
# Splitting the data into training and testing sets
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,
random_state=77, stratify=y)
```

Step 5 : Build and train the model

```
# Defining the parameters grid for GridSearchCV
param_grid={'C':[0.1,1,10,100],
      'gamma':[0.0001,0.001,0.1,1],
      'kernel':['rbf','poly']}

# Creating a support vector classifier
svc=svm.SVC(probability=True)

# Creating a model using GridSearchCV with the parameters grid
model=GridSearchCV(svc,param_grid)

# Training the model using the training data
model.fit(x_train,y_train)
```

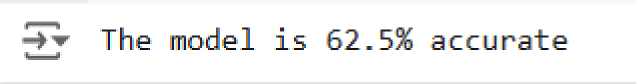
OUTPUT :



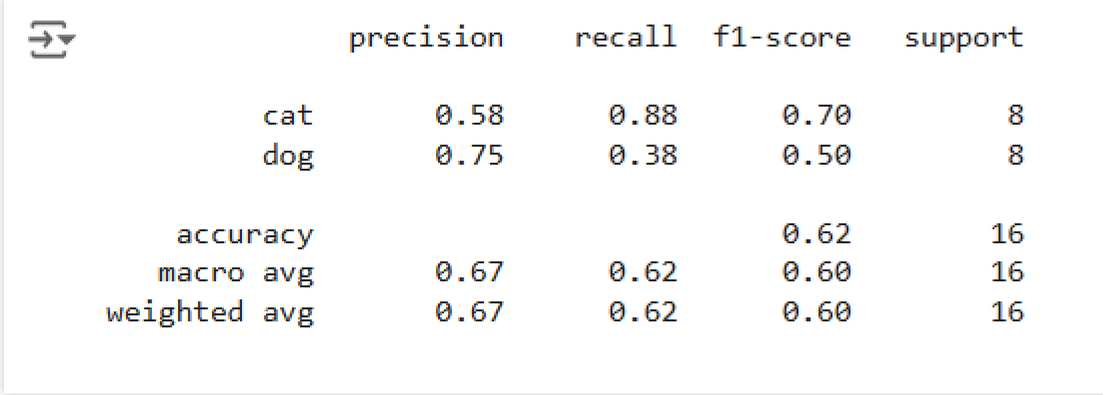
Step 6 : Model evaluation

```
# Testing the model using the testing data
y_pred = model.predict(x_test)

# Calculating the accuracy of the model
accuracy = accuracy_score(y_pred, y_test)

# Print the accuracy of the model
print(f"The model is {accuracy*100}% accurate")

print(classification_report(y_test, y_pred, target_names=['cat', 'dog']))
```

OUTPUT :



```
The model is 62.5% accurate
```

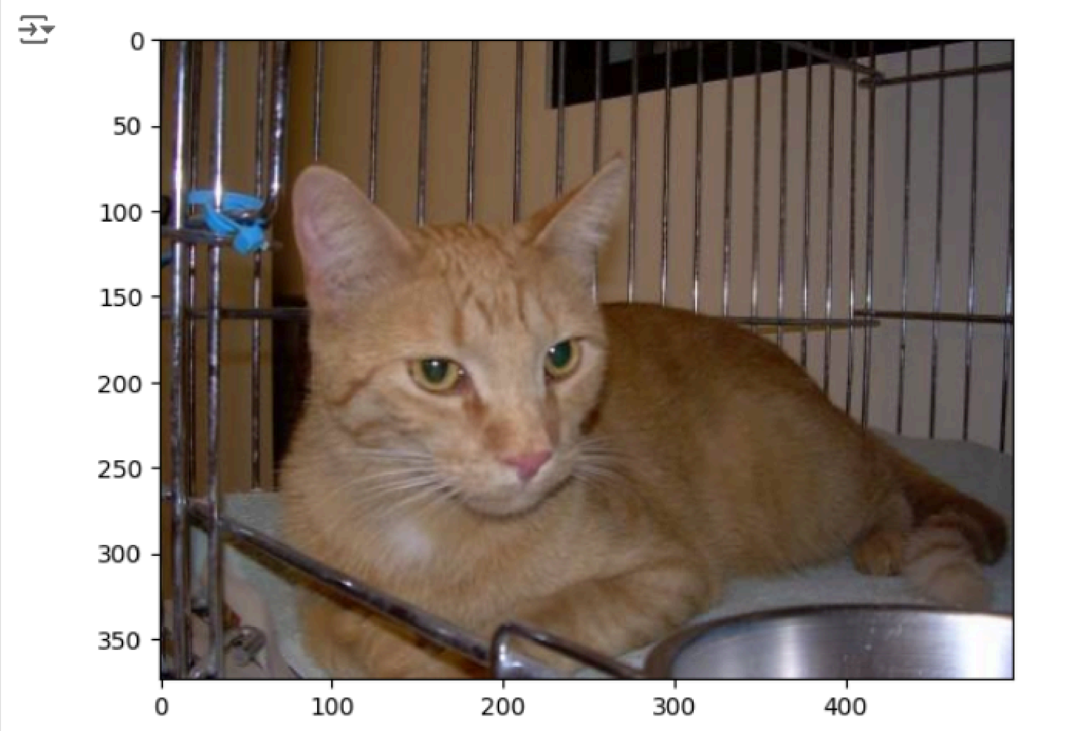


```
              precision    recall  f1-score   support

         cat       0.58      0.88      0.70         8
         dog       0.75      0.38      0.50         8

    accuracy                           0.62        16
   macro avg       0.67      0.62      0.60        16
weighted avg       0.67      0.62      0.60        16
```

Step 7 : Prediction

```
path='/content/cat.83.jpg'
img=imread(path)
plt.imshow(img)
plt.show()
img_resize=resize(img,(150,150,3))
l=[img_resize.flatten()]
probability=model.predict_proba(l)
for ind,val in enumerate(Categories):
  print(f'{val} = {probability[0][ind]*100}%')
print("The predicted image is : "+Categories[model.predict(l)[0]])
```

OUTPUT :



```
cats = 52.70216647851706%
dogs = 47.29783352148294%
The predicted image is : cat
```

RESULT :

Thus the process helps us to implement the face recognition using SVM Classifier using python program.