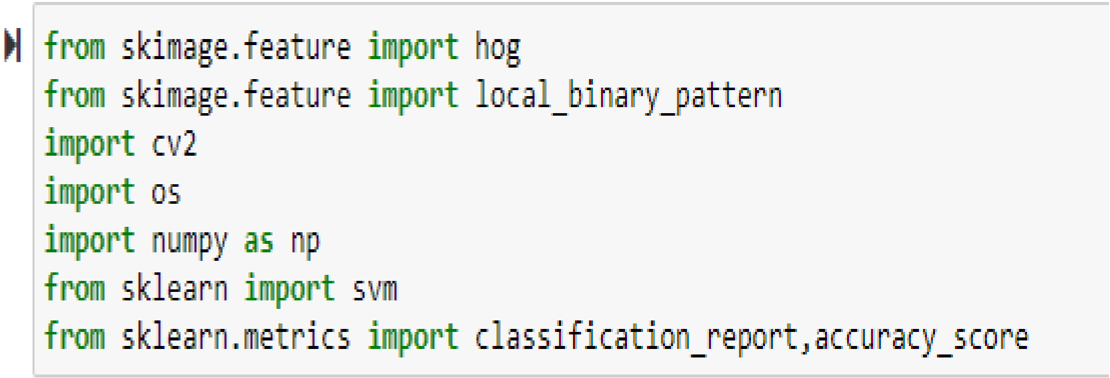
# Image Analysis and Computer Vision

Name: Harsha Siddapura Gnaneshwara
WsuID: W786P696.

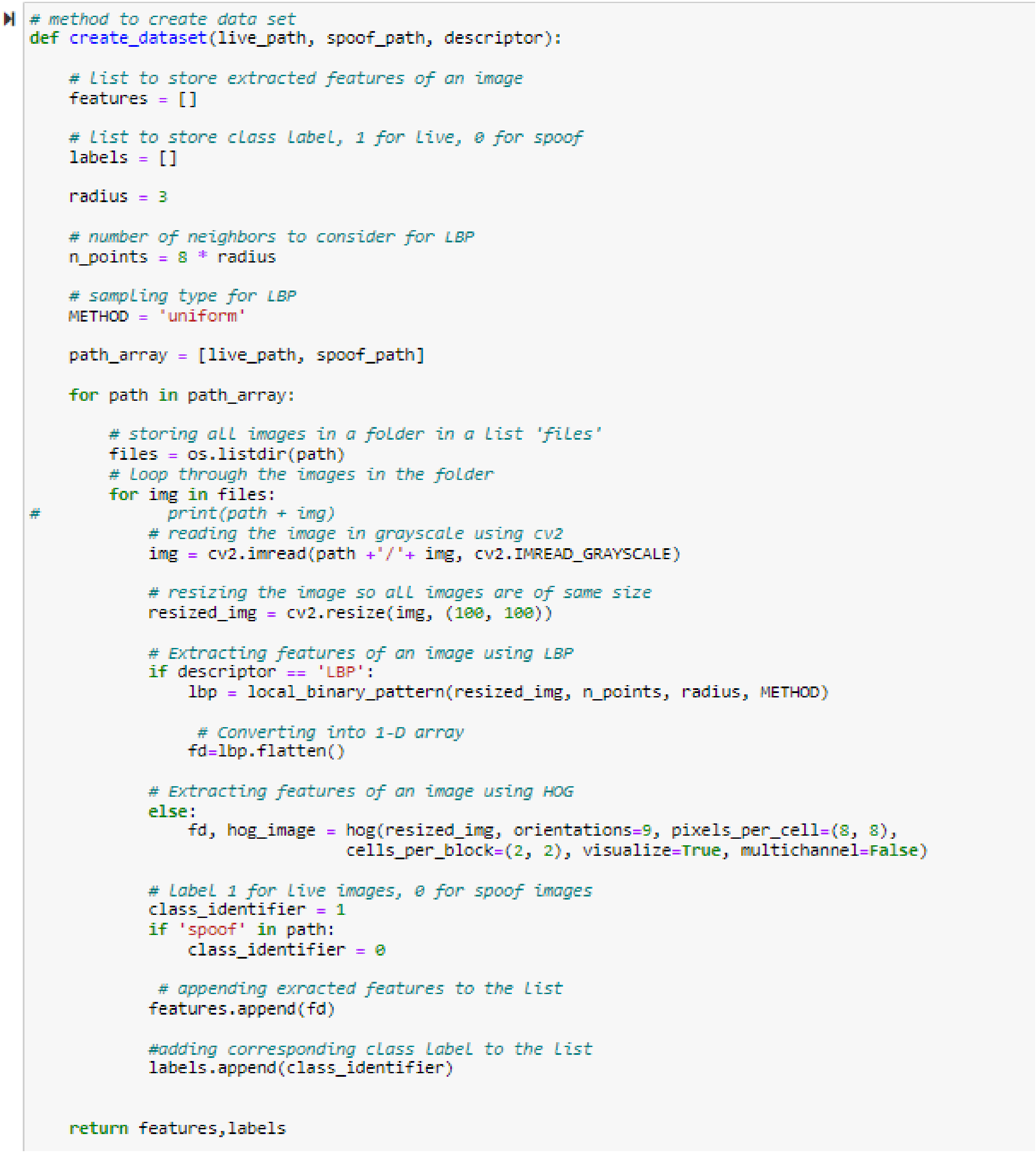
## Title: Fingerprint-based Spoof Detector based on LBP.



```
from skimage.feature import hog
from skimage.feature import local_binary_pattern
import cv2
import os
import numpy as np
from sklearn import svm
from sklearn.metrics import classification_report,accuracy_score
```

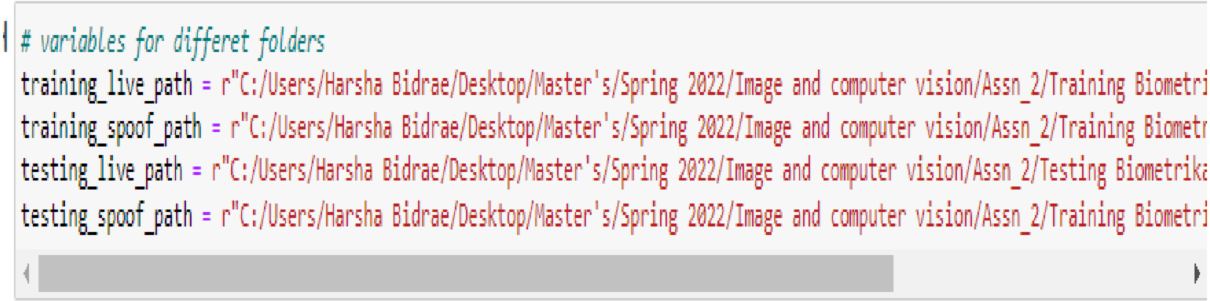
- Importing the required libraries to implement the project.

- In this step I have created a method create dataset and passing three parameters (live_path, spoof_path, description).
- I have used the list to store the extracted features of the images and also to store the class labels, 1 for live and 0 for spoof.



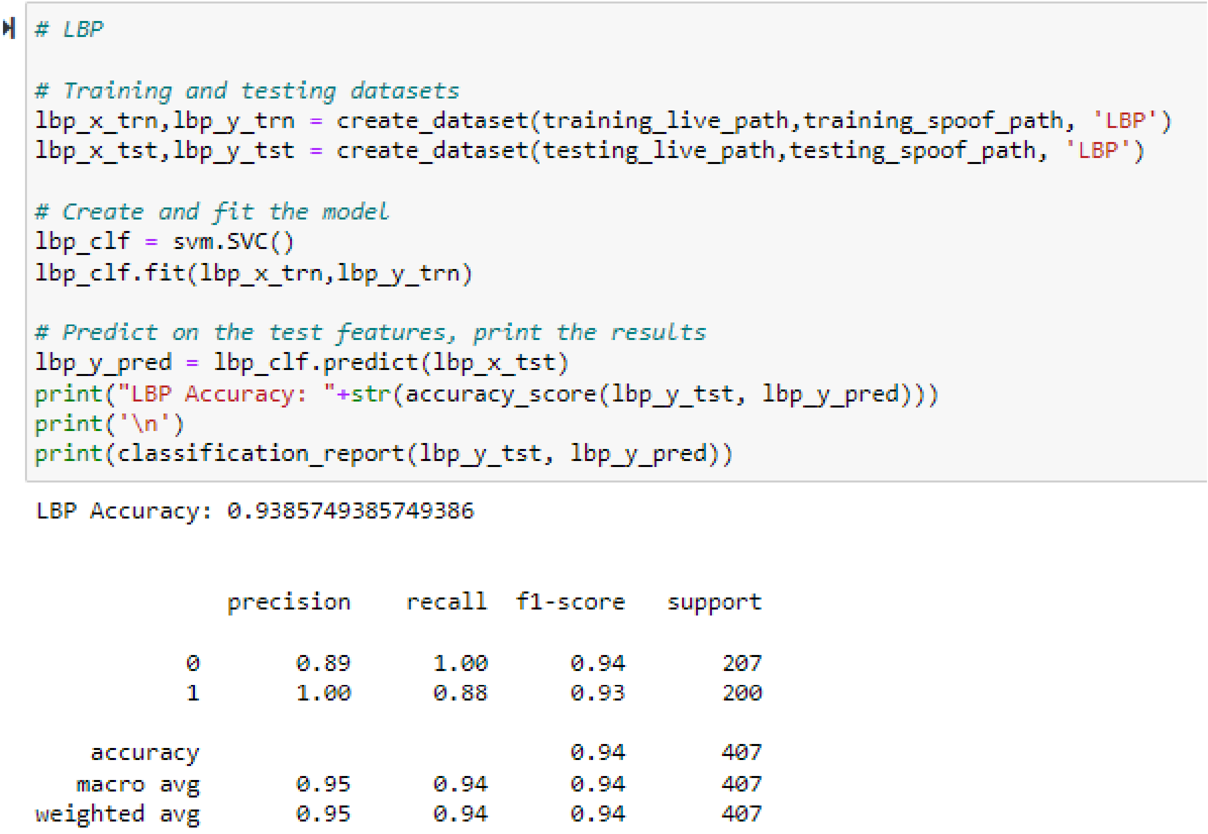
```
# method to create data set
def create_dataset(live_path, spoof_path, descriptor):

    # list to store extracted features of an image
    features = []

    # list to store class label, 1 for live, 0 for spoof
    labels = []

    radius = 3

    # number of neighbors to consider for LBP
    n_points = 8 * radius

    # sampling type for LBP
    METHOD = 'uniform'

    path_array = [live_path, spoof_path]

    for path in path_array:

        # storing all images in a folder in a list 'files'
        files = os.listdir(path)
        # loop through the images in the folder
        for img in files:
#             print(path + img)
            # reading the image in grayscale using cv2
            img = cv2.imread(path +'/'+ img, cv2.IMREAD_GRAYSCALE)

            # resizing the image so all images are of same size
            resized_img = cv2.resize(img, (100, 100))

            # Extracting features of an image using LBP
            if descriptor == 'LBP':
                lbp = local_binary_pattern(resized_img, n_points, radius, METHOD)

                 # Converting into 1-D array
                fd=lbp.flatten()

            # Extracting features of an image using HOG
            else:
                fd, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8),
                                cells_per_block=(2, 2), visualize=True, multichannel=False)

            # label 1 for live images, 0 for spoof images
            class_identifier = 1
            if 'spoof' in path:
                class_identifier = 0

             # appending exracted features to the list
            features.append(fd)

            #adding corresponding class label to the list
            labels.append(class_identifier)


    return features,labels
```

- After, importing the images I have converted all the images into grayscale and resized them using cv2.resize to get all the images in the same size.
- Then I have applied Linear binary pattern with four parameters (resized image, n points, radius and method), and then adding the features into 1-D array.
- Also implemented the Histogram of Oriented Gradients and adding the features into the list.



```
# variables for differet folders
training_live_path = r"C:/Users/Harsha Bidrae/Desktop/Master's/Spring 2022/Image and computer vision/Assn_2/Training Biometri
training_spoof_path = r"C:/Users/Harsha Bidrae/Desktop/Master's/Spring 2022/Image and computer vision/Assn_2/Training Biometr
testing_live_path = r"C:/Users/Harsha Bidrae/Desktop/Master's/Spring 2022/Image and computer vision/Assn_2/Testing Biometrika
testing_spoof_path = r"C:/Users/Harsha Bidrae/Desktop/Master's/Spring 2022/Image and computer vision/Assn_2/Training Biometri
```

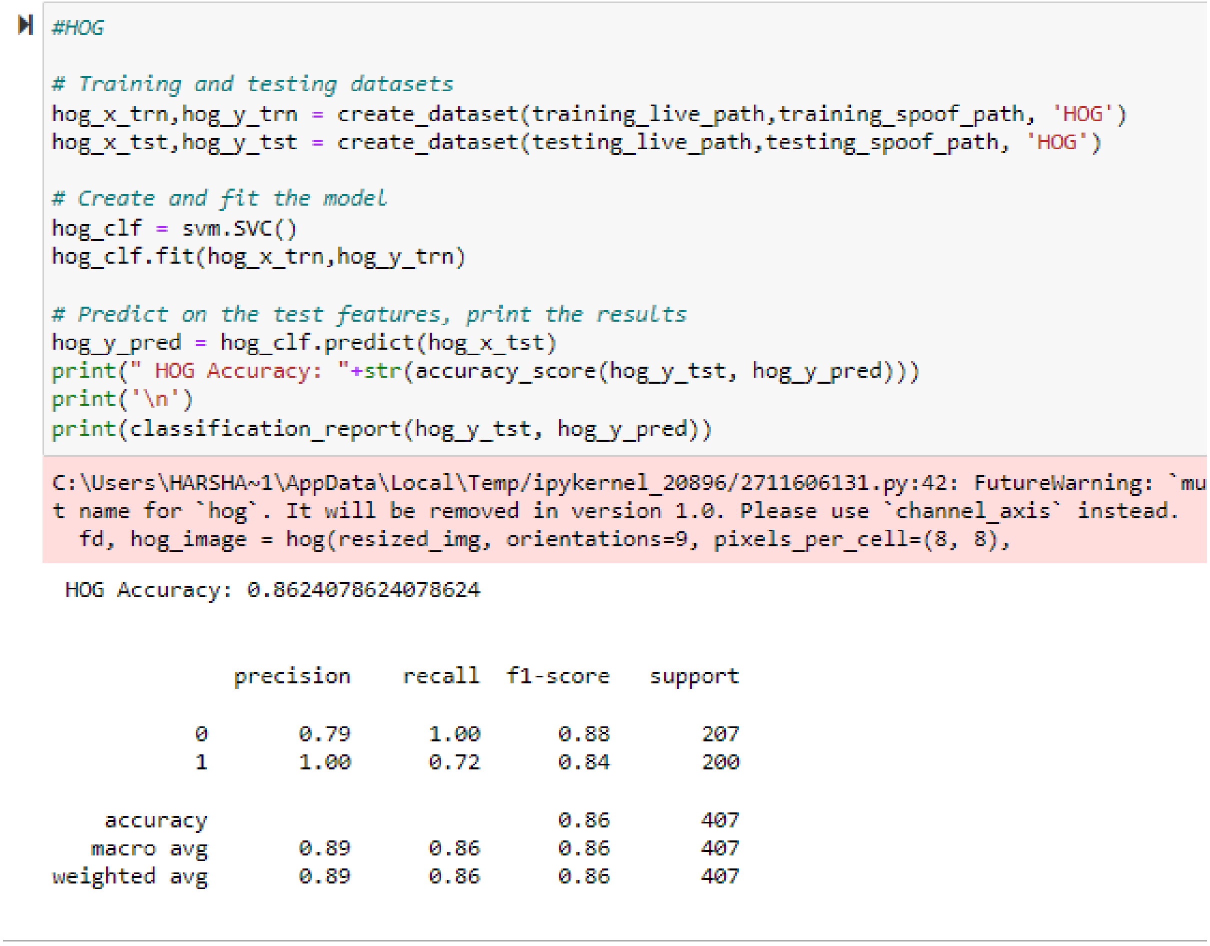
- Here I have created variables for different variables.



```
# LBP

# Training and testing datasets
lbp_x_trn,lbp_y_trn = create_dataset(training_live_path,training_spoof_path, 'LBP')
lbp_x_tst,lbp_y_tst = create_dataset(testing_live_path,testing_spoof_path, 'LBP')

# Create and fit the model
lbp_clf = svm.SVC()
lbp_clf.fit(lbp_x_trn,lbp_y_trn)

# Predict on the test features, print the results
lbp_y_pred = lbp_clf.predict(lbp_x_tst)
print("LBP Accuracy: "+str(accuracy_score(lbp_y_tst, lbp_y_pred)))
print('\n')
print(classification_report(lbp_y_tst, lbp_y_pred))
```

```
LBP Accuracy: 0.9385749385749386


              precision    recall  f1-score   support

           0       0.89      1.00      0.94       207
           1       1.00      0.88      0.93       200

    accuracy                           0.94       407
   macro avg       0.95      0.94      0.94       407
weighted avg       0.95      0.94      0.94       407
```

- Here I have trained and tested the dataset using the create data set method above.
- After that, I have fitted the svm model to predict the feature and print the results.
- The accuracy score of LBP is 93.2%, compare to HOG, LBP is giving good score.



```
#HOG

# Training and testing datasets
hog_x_trn,hog_y_trn = create_dataset(training_live_path,training_spoof_path, 'HOG')
hog_x_tst,hog_y_tst = create_dataset(testing_live_path,testing_spoof_path, 'HOG')

# Create and fit the model
hog_clf = svm.SVC()
hog_clf.fit(hog_x_trn,hog_y_trn)

# Predict on the test features, print the results
hog_y_pred = hog_clf.predict(hog_x_tst)
print(" HOG Accuracy: "+str(accuracy_score(hog_y_tst, hog_y_pred)))
print('\n')
print(classification_report(hog_y_tst, hog_y_pred))
```

```
C:\Users\HARSHA~1\AppData\Local\Temp/ipykernel_20896/2711606131.py:42: FutureWarning: `mu
t name for `hog`. It will be removed in version 1.0. Please use `channel_axis` instead.
  fd, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8),
```

```
 HOG Accuracy: 0.8624078624078624


              precision    recall  f1-score   support

           0       0.79      1.00      0.88       207
           1       1.00      0.72      0.84       200

    accuracy                           0.86       407
   macro avg       0.89      0.86      0.86       407
weighted avg       0.89      0.86      0.86       407
```

- Here I have trained and tested the dataset using the create data set method above.
- After that, I have fitted the svm model to predict the feature and print the results.
- The accuracy score of HOG is 86.2%, compare to LBP, HOG is giving less score.