Fruit & Vegetable Freshness Recognition

Final Report

1ST SEMESTER SY 2023 - 2024
FINAL PERIOD

Emerging Technologies 1
CPE 018
CPE31S1

Submitted to:
Engr. Jonathan V. Taylar

Submitted on:
December 15, 2023

Submitted by:
Group 2
Capalungan, James Victor
Catorce, Mayah Mae
Cortez, Angelica
Cuevas, Christian Jay
De Leon, Bryan Paul

Technological Institute of the Philippines, Quezon City
CPE 018 - Emerging Technologies 1 in CpE
Dr. Jonathan Taylar

POWERPOINT

- **This is the powerpoint that explains the problem and the solution that we came up with. We explained the problem, the solution, and what are the possible applications and benefits of our solution.**

Problem

Manual assessment of fruit and vegetable freshness is labor-intensive.

slidesmania.com

Solution:

Our project aims to alleviate this by implementing an automated Freshness Recognition Program using OpenCV. This system reduces manual labor, efficiently assesses the condition of each item, and can be employed in warehouses for real-time monitoring of produce freshness.

slidesmania.com

Efficiently segregating fresh fruits and vegetables from rotten ones ensures quality preservation, minimizes waste, and promotes healthier consumption.

slidesmania.com

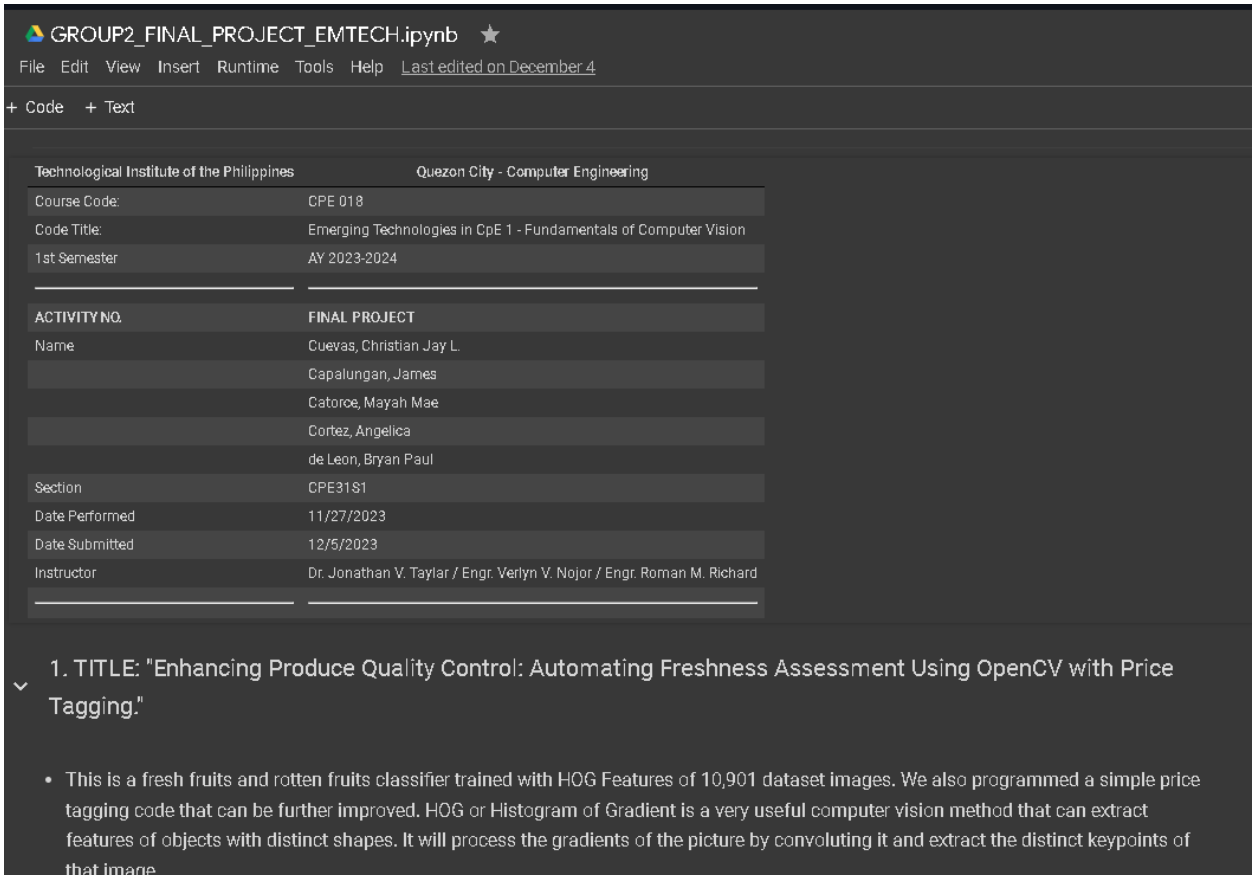
Thank you!

Do you have any questions?

slidesmania.com

GOOGLE COLAB (LINK)

- **This is the google colab link that we made to compile the codes and the result of our testing of the program.**



GROUP2_FINAL_PROJECT_EMTECH.ipynb
(https://colab.research.google.com/drive/1FRkObiE85U2FiJeT4NvqhkiHmR1HzTKf?usp=sharing)

CODE BREAKDOWN

This section elucidates every aspect of the code, providing a detailed explanation of the function of each block. This code is also in the Google Colab link that we submitted. We used 2 computer vision techniques that we learned in the class, the HOG and Color Histogram. We then trained a K-Nearest Neighbor classifier algorithm with the extracted features and keypoints from the dataset that we gathered.

```
–This contains all the imports of the program–
import cv2
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import pandas as pd
from skimage.color import rgb2gray
from skimage.transform import rescale, resize, downscale_local_mean
from sklearn.model_selection import train_test_split
from skimage import data, color, feature
from skimage.feature import hog
from skimage import exposure
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import glob

–This is used to extract HoG Features from the images–
def FtrExtractHOG(img):
    #Feature Extraction using HOG
    ftr,_=hog(img, orientations=8, pixels_per_cell=(16, 16),
            cells_per_block=(1, 1), visualize=True)
    return ftr

–This is used to extract color histogram from the images–
def FtrExtractColorHist(img):
    chans = cv2.split(img)
    colors = ("h", "s", "v")
    features = []

    for (chan, color) in zip(chans, colors):
        hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
        features.extend(hist)
    return np.array(features).flatten()

–This is used to preprocess images for HOG extraction–
def preprocessing1(arr):
```

```
    arr_prep=[]
    for i in range(np.shape(arr)[0]):
        img = cv2.cvtColor(arr[i], cv2.COLOR_BGR2GRAY)
        img=resize(img, (72, 72),anti_aliasing=True)
        arr_prep.append(img)
    return arr_prep
```

–Function that loops every element in array to extract the HOG–

```
def featureExtraction1(arr):
    arr_feature=[]
    for i in range(np.shape(arr)[0]):
        arr_feature.append(FtrExtractHOG(arr[i]))
    return arr_feature
```

–This is used to preprocess images for Color Histogram extraction–

```
def preprocessing2(arr):
    arr_prep=[]
    for i in range(np.shape(arr)[0]):
        img=matplotlib.colors.rgb_to_hsv(arr[i])
        arr_prep.append(img)
    return arr_prep
```

–Function that loops every element in array to extract the Color Histogram–

```
def featureExtraction2(arr):
    arr_feature=[]
    for i in range(np.shape(arr)[0]):
        arr_feature.append(FtrExtractColorHist(arr[i]))
    return arr_feature
```

–This code will load the files with string name ending with ".png" in the test folder–

```
def loadimage(arr,n,name_of_fruit):
    label=[]
    for i in range(n):
        strr = "test/"+name_of_fruit+"_"+str(i+1)+"/*.png"
        #print(strr)
        for file in glob.glob(strr):
            img=np.asarray(plt.imread(file))
            #original  img1=cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
            img1=cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
            img2= cv2.resize(img1, (72, 72))
            arr.append(img2)
```

```
            label.append(name_of_fruit)
    return arr,label

freshapples=[]
freshbanana =[]
freshoranges=[]
rottenapples=[]
rottenbanana =[]
rottenoranges=[]

–This code will store the appended images to array freshapples and the label to array–
label_freshapples
freshapples,label_freshapples=loadimage(freshapples,1,"freshapples")
freshbanana,label_freshbanana=loadimage(freshbanana,1,"freshbanana")
freshoranges,label_freshoranges=loadimage(freshoranges,1,"freshoranges")
rottenapples,label_rottenapples=loadimage(rottenapples,1,"rottenapples")
rottenbanana,label_rottenbanana=loadimage(rottenbanana,1,"rottenbanana")
rottenoranges,label_rottenoranges=loadimage(rottenoranges,1,"rottenoranges")


freshapples_array = np.array(freshapples)
freshbanana_array = np.array(freshbanana)
freshoranges_array = np.array(freshoranges)
rottenapples_array = np.array(rottenapples)
rottenbanana_array = np.array(rottenbanana)
rottenoranges_array = np.array(rottenoranges)

–Create a DataFrame–
raw_atributte = {
   'CLASS': ['Fresh Apple', 'Fresh Banana', 'Fresh Orange', 'Rotten Apple', 'Rotten Banana', 'Rotten
Orange'],
   'NUMBER': [freshapples_array.shape[0], freshbanana_array .shape[0],
freshoranges_array.shape[0], rottenapples_array.shape[0],
            rottenbanana_array.shape[0], rottenoranges_array.shape[0]]
}
atributte = pd.DataFrame(raw_atributte, columns=['CLASS', 'NUMBER'])

–Display the DataFrame–
print(atributte)
```

	CLASS	NUMBER
0	Fresh Apple	1693
1	Fresh Banana	1581
2	Fresh Orange	1466
3	Rotten Apple	2342
4	Rotten Banana	2224
5	Rotten Orange	1595

–This code is just to show the images in the dataset–

```
print('Full Dataset')
fig = plt.figure()
ax1 = fig.add_subplot(3,3,1)
ax1.set_title('Fresh Apple')
ax1.set_axis_off()
ax1.imshow(cv2.cvtColor(freshapples[0], cv2.COLOR_HSV2RGB))

ax2 = fig.add_subplot(3,3,2)
ax2.set_title('Fresh Banana')
ax2.set_axis_off()
ax2.imshow(cv2.cvtColor(freshbanana[0], cv2.COLOR_HSV2RGB))

ax3 = fig.add_subplot(3,3,3)
ax3.set_title('Fresh Orange')
ax3.set_axis_off()
ax3.imshow(cv2.cvtColor(freshoranges[0], cv2.COLOR_HSV2RGB))

ax4 = fig.add_subplot(3,3,4)
ax4.set_title('Rotten Apples')
ax4.set_axis_off()
ax4.imshow(cv2.cvtColor(rottenapples[0], cv2.COLOR_HSV2RGB))

ax5 = fig.add_subplot(3,3,5)
ax5.set_title('Rotten Banana')
ax5.set_axis_off()
ax5.imshow(cv2.cvtColor(rottenbanana[0], cv2.COLOR_HSV2RGB))

ax6 = fig.add_subplot(3,3,6)
ax6.set_title('Rotten Oranges')
ax6.set_axis_off()
```

ax6.imshow(cv2.cvtColor(rottenoranges[0], cv2.COLOR_HSV2RGB))

```
Full Dataset
<matplotlib.image.AxesImage at 0x196854898d0>
```

–This code is used to split the dataset into images that will be trained and images that will be tested with the trained classifier–

X_train, X_test, y_train, y_test = train_test_split(X_Shapedes, y_Shapedes, test_size=0.33, random_state=42)

–This code is to call the function preprocessing1 to process the X_train array. This will append a new set of images that was converted into GRAYSCALE and also resized–

X_trainp=preprocessing1(X_train)
X_testp=preprocessing1(X_test)

–This code will extract the HOG features of the arrays that was preprocessed–

X_trainftr=featureExtraction1(X_trainp)
X_testftr=featureExtraction1(X_testp)

–This calls the KNearestNeighbor Classifier and it trains the classifier on the array X_trainftr which is the array of the images and the y_train which is the array of the labels–

knn_clf = KNeighborsClassifier(n_jobs=-1, weights='distance', n_neighbors=7)
knn_clf.fit(X_trainftr, y_train)

```
KNeighborsClassifier
KNeighborsClassifier(n_jobs=-1, n_neighbors=7, weights='distance')
```

–This function is for processing the frames of the live feed and returning the prediction that was made with the KNN Classifier–

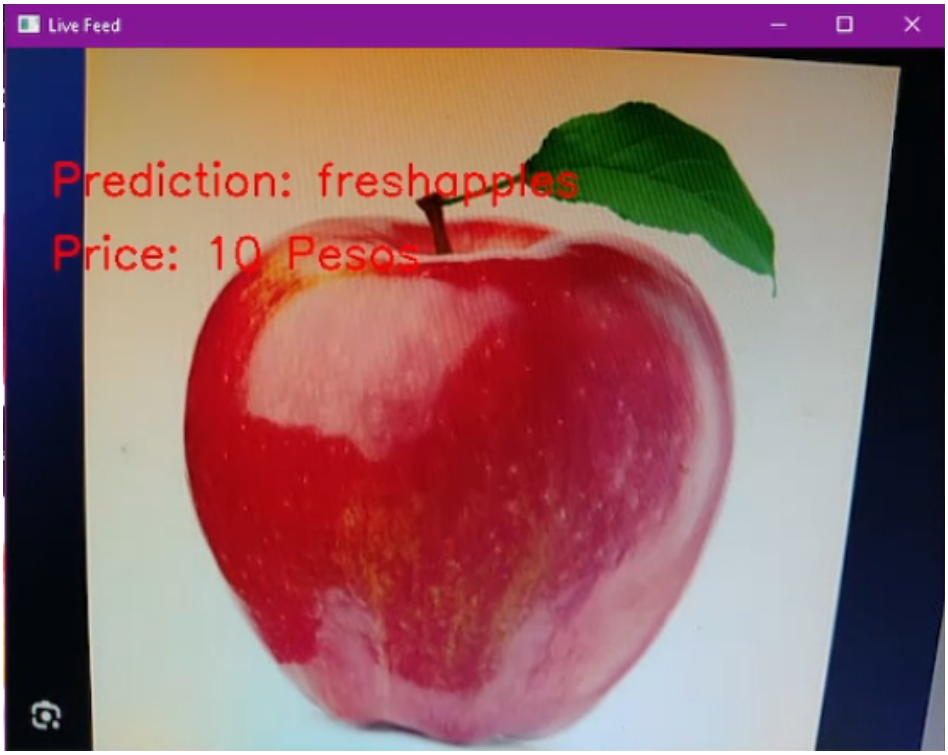
```
def process_frame(frame):
    arr = []
    img=np.array(frame)
    img1=cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
    img2= cv2.resize(img1, (72, 72))
    arr.append(img2)
    # Preprocess the frame
    frame_prep = preprocessing1(arr)
    # Extract features from the preprocessed frame
    frame_ftr = featureExtraction1(frame_prep)
    # Predict the class using the trained KNN model
    prediction = knn_clf.predict(frame_ftr)[0]
    return prediction
```

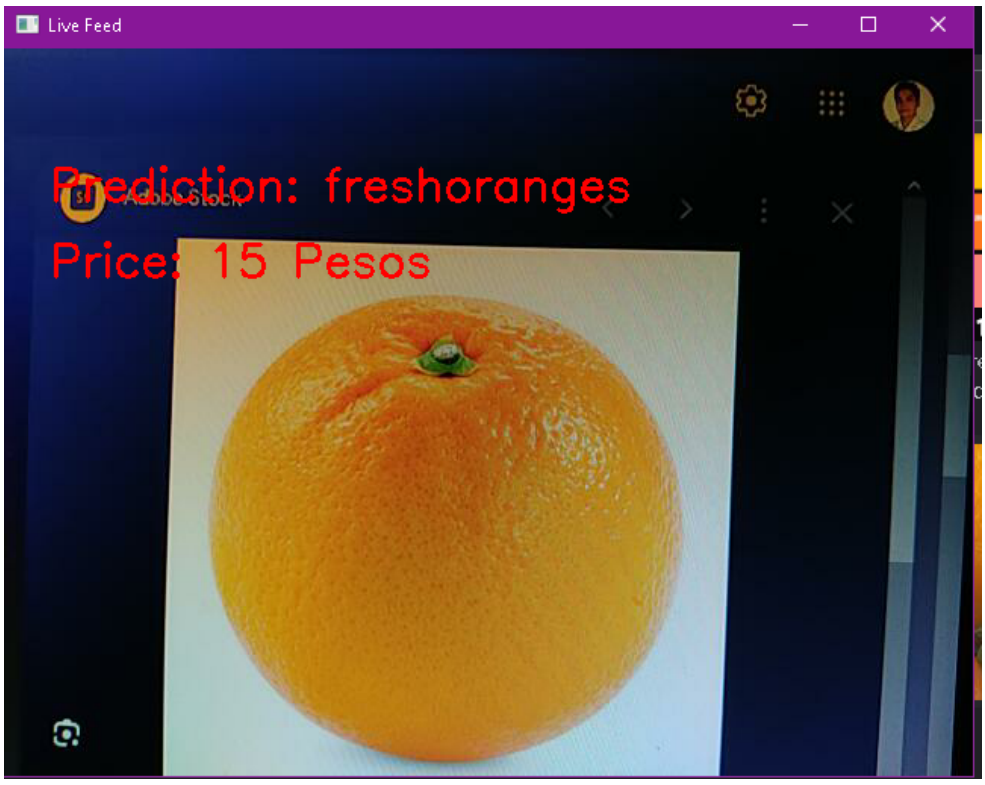
–This code is used for the running the window and the camera live feed. This code will get the frame, pass it to process_frame,and then it will test if the prediction variable is equal to the labels. It will display the prediction and then the price that was set–

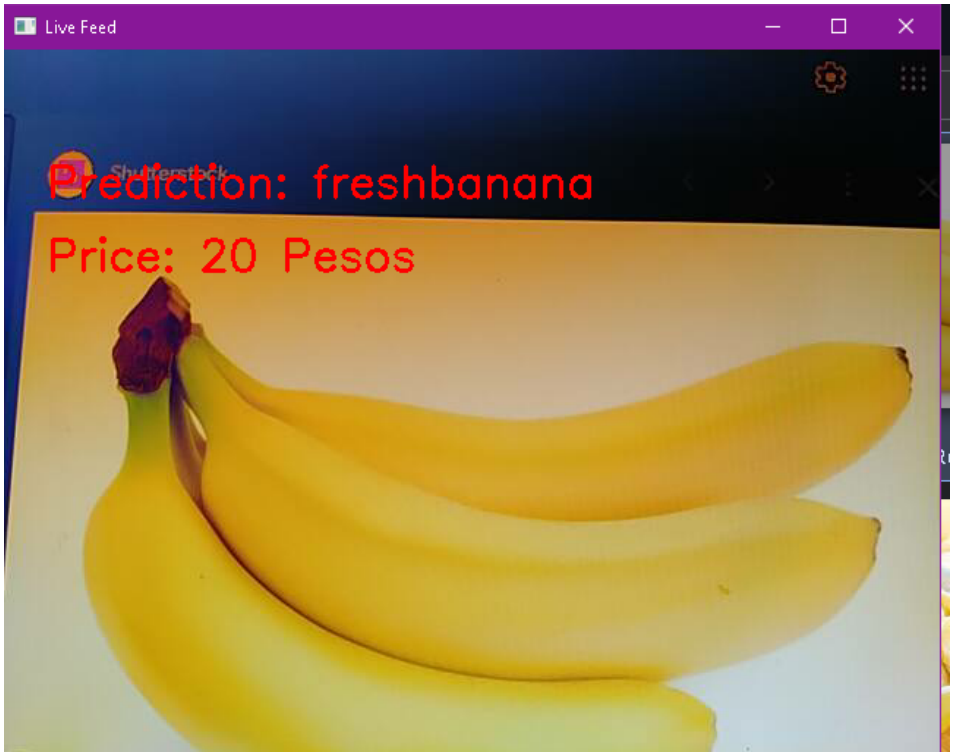
```
cap = cv2.VideoCapture(2)

while True:
    # Capture frame-by-frame
    ret, frame = cap.read()
    if frame is None:
      break

    # Process the frame
    prediction = process_frame(frame)

    if prediction == "freshapples":
    # Display the result on the frame
      cv2.putText(frame, f"Prediction: {prediction}", (30, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
      cv2.putText(frame, "Price: 10 Pesos", (30, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
```

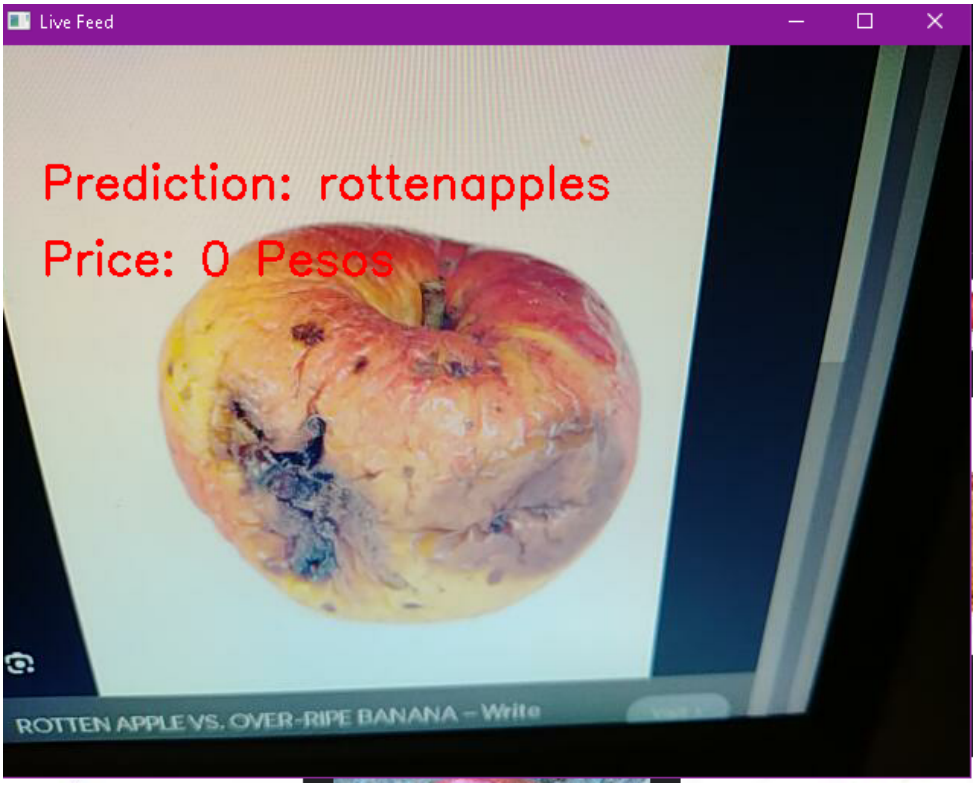
```
    elif prediction == "freshoranges":
        cv2.putText(frame, f"Prediction: {prediction}", (30, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
        cv2.putText(frame, "Price: 15 Pesos", (30, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    elif prediction == "freshbanana":
        cv2.putText(frame, f"Prediction: {prediction}", (30, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
        cv2.putText(frame, "Price: 20 Pesos", (30, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    elif prediction == "rottenapples":
        cv2.putText(frame, f"Prediction: {prediction}", (30, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
        cv2.putText(frame, "Price: 0 Pesos", (30, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

    elif prediction == "rottenoranges":
        cv2.putText(frame, f"Prediction: {prediction}", (30, 100), cv2.FONT_HERSHEY_SIMPLEX, 1,(0, 0,
255), 2)
        cv2.putText(frame, "Price: 0 Pesos", (30, 150), cv2.FONT_HERSHEY_SIMPLEX, 1,(0, 0, 255), 2)

    elif prediction == "rottenbanana":
        cv2.putText(frame, f"Prediction: {prediction}", (30, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2)
        cv2.putText(frame, "Price: 0 Pesos", (30, 150), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    # Display the frame

    cv2.imshow('Live Feed', frame)

     --Break the loop if 'q' key is pressed--
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

--Release the video capture object and close all windows--
cap.release()
cv2.destroyAllWindows()
```

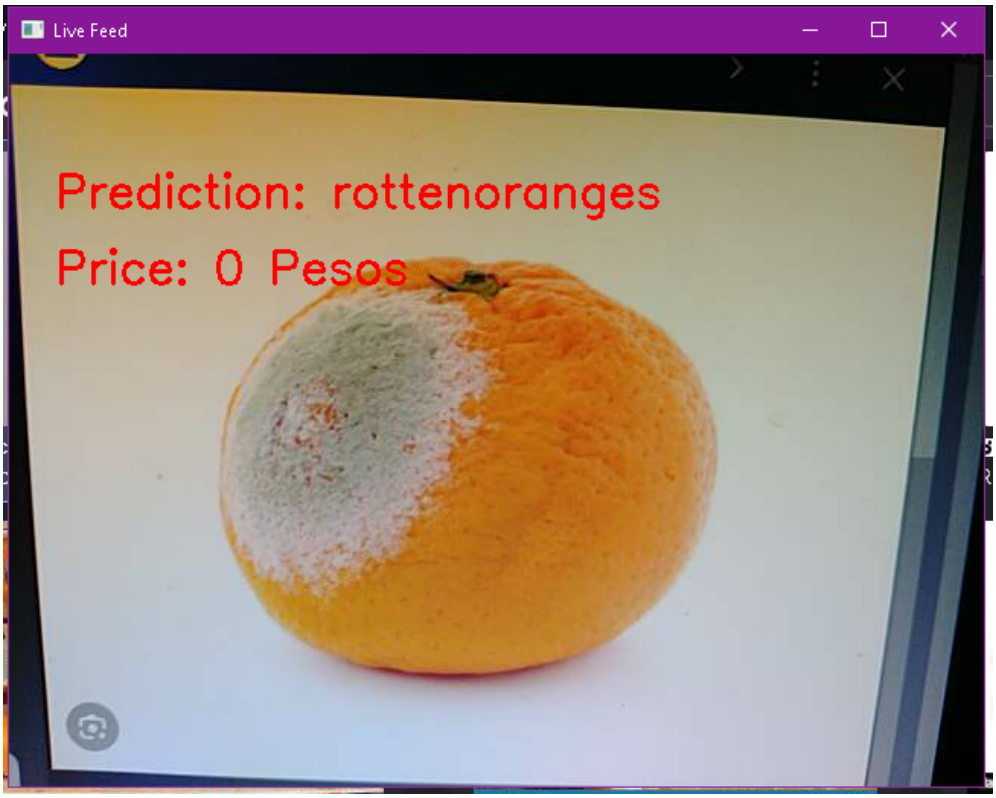
Tests Made:

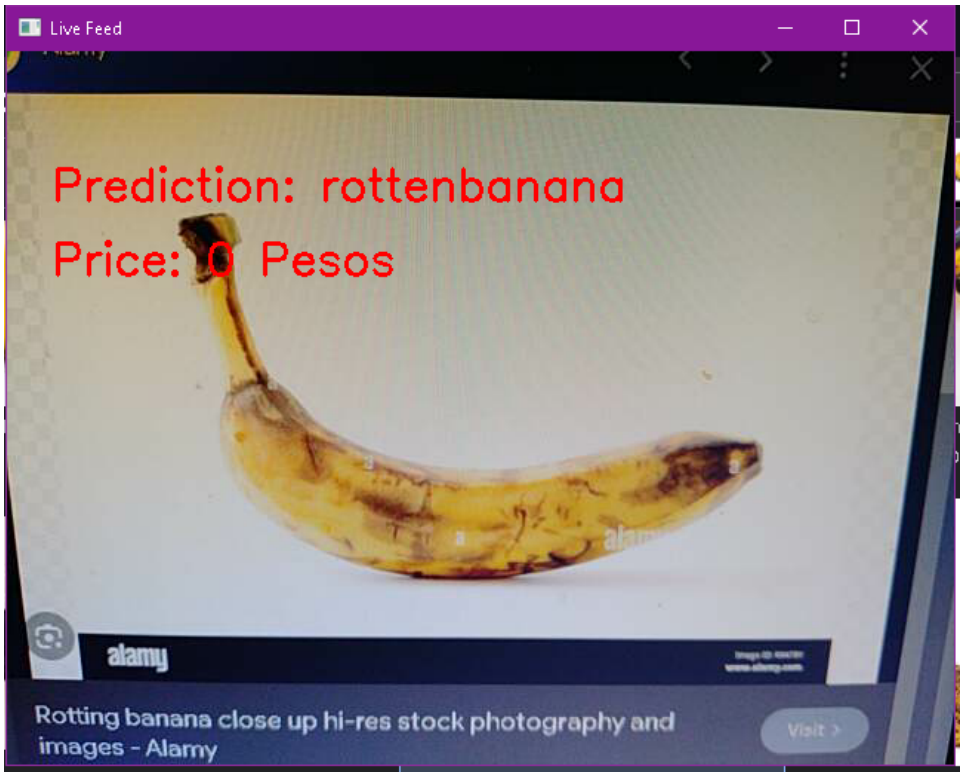

Live Feed
Prediction: freshapples
Price: 10 Pesos


Live Feed
Prediction: freshoranges
Price: 15 Pesos


Live Feed
Prediction: freshbanana
Price: 20 Pesos


Live Feed
Prediction: rottenapples
Price: 0 Pesos
ROTTEN APPLE VS. OVER-RIPE BANANA – Write


Live Feed
Prediction: rottenoranges
Price: 0 Pesos


Live Feed
Prediction: rottenbanana
Price: 0 Pesos
alamy
Rotting banana close up hi-res stock photography and images - Alamy
Visit

–This code is used for testing the other part of the dataset–

```
y_knn_pred = knn_clf.predict(X_testftr)
```

–This code is used to see the accuracy of the matching of the classifier–

```
print(accuracy_score(y_test, y_knn_pred)*100,'%')
```

–This code is used to show the objects that was matched with the labels using the KNN Classifier–

```
slice = 15

plt.figure(figsize=(16,8))
for i in range(slice):
    plt.subplot(1, slice, i+1)
    plt.imshow(cv2.cvtColor(X_test[i], cv2.COLOR_HSV2RGB), interpolation='nearest')
    plt.text(0, 0, y_knn_pred[i], color='black',
             bbox=dict(facecolor='white', alpha=1))
    plt.axis('off')
```

VIDEO OF OUR GROUP/ PROOF OF COLLABORATION

- **This is the video presentation that we made where we presented the problem and how we solved it. We also explained the code line by line and showcased the working program at the end of the video.**

MEMBERS

	**Capalungan, James Victor** qjbcapalungan@tip.edu.ph
	**Catorce, Mayah Mae** qmmacatorce@tip.edu.ph
	**Cortez, Angelica** qa-cortez@tip.edu.ph
	**Cuevas, Christian Jay** qcjlcuevas@tip.edu.ph

de Leon, Bryan Paul
qbpadeleon@tip.edu.ph