FRUIT RECOGNITION

BY

ADRIAN SHEN LOONG A/L SENG YEAN

MABELLYN LIEW LINZHEN

OOI CHIN HAO

WAN KAR HOU

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology

(Kampar Campus)

MAY 2020

ABSTRACT

Done by: Wan Kar Hou

In 21st century, the world had entered Industrial Revolution 4.0 which heavily focus on interconnectivity, automation, machine learning, and real-time data through high technology. Computer vision was one of the critical parts of automation systems in Industry 4.0, it had been applied in many sectors of industries such as agriculture industry and even daily life of human being.

In this report, computer vision technology was applied to assists those who are visually challenged and to reduce problems caused by human errors in a working environment when conducting specific task such as classification of fruit in farm. It developed to perform classification of fruit using the images taken from webcam.

In purpose to achieve the objectives of this application, Naïve Bayes classifier was applied on this application. As for the first step of this system, image of fruit will be taken using webcam. Next, certain technique such as Gaussian Blurring, Image segmentation were applied to draw out actual part of fruit from images. Then the system will classify the type of fruit using Naïve Bayes classifier. Finally, after analyzing and predicting by the system, it will display the result with name of the fruit.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BBO	Biogeography-based Optimization
CNN	Convolution Neural Network
DWT	Discrete Wavelet Transform
FNN	Feedforward Neural Network
FSCABC	Fitness-Scaled Chaotic Artificial Bee Colony
GA	Genetic Algorithm
GNB	Gaussian Naïve Bayes
HSI	Habitat Suitability Index
MSE	Mean-Squared Error
OpenCV	Open Source Computer Vision Library
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
SCV	Stratified Cross Validation
SIV	Suitability Index Variables
SVM	Support Vector Machine
WE	Wavelet Entropy

Chapter 1 Introduction

1-1 Introduction

Done by: Mabellyn Liew LinZhen

In the very beginning, all of the fruits are manually pick up and classify by farmers or workers. It is a very challenging task as there are many kinds of fruits and some of the fruit look alike but it is different. Human is difficult to classify those fruits that look alike such as ripe yellow tomato and persimmon. In last decade, many researchers' attention had been attracted by the fruit classification based on machine learning and computer vision. In their studies, they used the external quality descriptors that are color, texture, size and shape. With the help of machine learning and computer vision, the accuracy of classify right fruits is higher than the manual fruit classification as the errors done by farmer or worker are decreased. In this study, we try to use Naïve Bayes method to classify the fruits.

1-2 Problem Statement

Done by: Adrian Shen Loong A/L Seng Yean

The existing manual fruit classification system has plenty of problems.

The first problem is individuals who suffer from color blindness hard to separate objects based on the color of object. One of the examples is separating red apples from green apples. Such individuals are not uncommon because there are about 300 million individuals who are suffering from color blindness which is close to the entire population in the USA (Colour Blind Awareness, n.d.).

The second problem is human limitation whereby when a task is done repeatedly by humans, the chance that errors to occur will rise in which a farmer might accidentally categorize the wrong fruit due to fatigue and repeating the same task. This kind of mistake might seem harmless but there are some fruits that looks edible but are actually harmful (poisonous or can cause death) for humans. According to a study conducted, the likelihood that a human error to occur on a particularly repeated task will increase by 48.8% due to repetition, stress, fatigue etc. (Jian Ai Yeow et al. September 2014).

1-3 Motivation

Done by: Adrian Shen Loong A/L Seng Yean

The problems that mentioned in Chapter 1-2 are still happening all over the world. Thus, in this project, we are here to propose a better algorithm which is Naive Bayes Algorithm to classify the fruits. Through this algorithm, we can reduce the problems caused by human errors in a working environment and this will save the farmer's or worker's time as they have more time to pick up the fruits.

1-4 Project Scope

Done by: Ooi Chin Hao

This project is focus on developing a fully functional fruit recognition system that aid in classification of fruit. To achieve this target, the system will make use of OpenCV technique together with some other technologies such as webcam to obtain and process the fruit image.

At first, user need to input a fruit image into the system. Then, the system will preprocess the image and classify it based on the output. The name of the fruit will be the final output of the system.

1-5 Project Objectives

Done by: Adrian Shen Loong A/L Seng Yean, Ooi Chin Hao

The system aims to assist those who are visually challenged and to reduce the errors caused by humans in a particular workplace. It will identify an object, in this case fruits and provide feedback in response to the object given. The following are the objectives for the project:

- The system is able to obtain the fruit image and proceed to image processing.
- The system will identify the fruit provided and provide feedback by returning what type of fruit it is.
- The technique implemented in the system lead to the high accuracy in recognition of fruit.
- The system must at least be able to identify what color is the fruit in a background that has different contrast with the fruit.

1-6 Impact, Significance and Contribution

Done by: Wan Kar Hou

There were limitations in human ability to perform specific task effectively such as recognition of fruit accurately due to many factors. Some human beings were born with color blindness which unable to distinguish specific color of fruits, thus affecting their judgement or recognition on fruits. Besides that, during the process for classification of fruit in farm by workers or farmers, mistake will occur normally as farmers would wrongly classify fruits. Automation of fruit recognition with computer vision was the solution to deal with these limitations or problem arises

As for those persons whom born with color blindness, their abilities to perform daily activities were restricted and limited compared to normal person, they face difficulties in identify type of fruits based on the color texture. The design of this fruit recognition system solved their daily problem and raising their confidence and self-esteem, through uploading the photo of the image of fruit taken to the system, the system automatically classifies the type of fruit for the them.

Smart agriculture is already becoming more commonplace among farmers, and high-tech farming is quickly becoming the standard in modern world. Thus, automation on recognition of fruit is one of the key elements in this evolution. As in a large commercial fruit farm which normally consists more than 10 types of fruits, with the implementation of automation fruit recognition, workers of the farm could be aided in the recognition of fruit by using this new technology, any mistake could be avoid by the workers. For instance, workers might confuse about the type of a fruit, as some fruit are look similar in terms of texture and size, identification for the type of fruit through computer vision provided an accurate answer for workers in terms of fruit name, fruit feature and fruit description. In such way, productivity and the efficiency for the business process of the farm improved and reduced the overall cost of lost for the farm owners.

1-7 Background Information

Done by: Mabellyn Liew LinZhen

Fruit recognition implements computer vision techniques. Computer vision techniques include feature extraction and pattern classification to enable user to recognize the fruit from an image.

Computer vision is a way of machine simulates human intelligence that can trains computers to interpret and know about the visual world. It lets computers check the extract image contents or content of multidimensional data generally to facilitate solving specific problem such as pattern classification problem. Computer vision works in 3 steps which are acquire images, process images, and understand images. Examples of application that used computer vision are industrial robots, autonomous vehicle, detection of tumor in medical field, and monitor crops growth in agriculture field.

Feature extraction is a method to abstract the image into feature to represent the image using less memory and more representing. Huge number of variables analysis needs a lot of memory and computation that cause overfits training samples. It constructs combinations of variables to overcome problems when describing data with acceptable accuracy.

Pattern classification is a way that use for high-level information of object, extracted feature include data that related to gray scale, texture, shape, or context in pattern recognition then the extracted features are assigned the object to a category.

Chapter 2 Literature Review

2-1-1 Fruit Classification by Wavelet-Entropy and Feedforward Neural Network

Done by: Mabellyn Liew LinZhen

A combination technique that used for fruit classification is Wavelet-Entropy and Feedforward Neural Network Trained by Fitness-Scaled Chaotic ABC and Biogeography-Based Optimization (Shuihua et al., 2015, p. 4). The aim of Wavelet-Entropy is to estimate the degree of order or disorder of the fruit image with high time-frequency resolution. Feedforward Neural Network technique is used because of its outstanding performance.

First of all, four-step pre-processing method was being used. They captured or obtained the fruit graphic from search engine and labelled it according to names and colour. They used split-and-merge algorithm to remove images background. They used square window to captured the fruits and centred it. They also down sampled the graphics to 256 * 256.

Next, they used discrete wavelet transform (DWT) technique which was an implementation tool using the dyadic positions and scales to generate a 2D fruit graphics. From equation (1), they deduced the continuous wavelet transform of the signal *x*(*t*) relative to a particular wavelet. From equation (2), wavelet *u*(*t* | *as*, *at*) was based on wavelet *u*(*t*) by dilation and translation. Discretization of equation (1) will generated DWT. From equation (3), it iterated with approximations being decomposed successively which is signal was broken down to the required level that meets the expected resolution. When the decomposition level increases, the coarser approximation components were obtained will be more compact.

$$C_u(a_s, a_t) = \int_{-\infty}^{\infty} x(t)u(t|a_s, a_t)\mathrm{d}t \qquad (1)$$

$$u(t\Big|a_t, a_s) = \frac{1}{\sqrt{a_s}}\psi(\frac{t-a_t}{a_s}) \qquad (2)$$

$$L(n\Big|k,j) = \mathrm{D_S}[\sum\nolimits_n l_j^*(n-2^j k)x(n)] \qquad (3)$$
$$H(n\Big|k,j) = \mathrm{D_S}[\sum\nolimits_n h_j^*(n-2^j k)x(n)]$$

After that, they used wavelet entropy. There are three-level decomposition was employed for each channel R or channel G or channel B. Each channel had 10 features and total number of features was 30 features. Principal component analysis (PCA) decrease features number that explained more than 95% of original features.

After PCA, they used five-fold stratified cross validation (SCV) to enhance the generation capability of FNN. They divided the graphics into five distributed folds and let *i*-th fold be test set while the rest four folds be training set. They recorded the accuracy of test set and proceeded to others folds until the fifth fold being tested. After done all the folds, they also calculated the average accuracy of these five folds.

Then, they fed the graphics into feedforward neural network (FNN) that classified nonlinear separable patterns and approximated arbitrary continuous function. FNN training selected the optimal weights to make mean-squared error (MSE) minimal. Genetic algorithm (GA) Simulated annealing (SA), artificial bee colony (ABC) and particle swarm optimization (PSO) were used to train weights. They proposed to use both fitness-scaled chaotic artificial bee colony (FSCABC) and biogeography-based optimization (BBO) because FSCABC was time-consuming.

For FSCABC, they evaluated and initialized the population. New solutions were produced in the neighborhood to choose the best solutions. New solution was generated for onlookers based on population group of last steps and selected best scaled fitness values probability. The discarded solution was produced that replaced random number generator with chaotic number generator and output the results.

For BBO, they initialized BBO parameters that included mapping problem solutions to suitability index variables (SIV) and habitats. They initialized population by generate random set of habitats and computed habitat suitability index (HSI) to each habitat. The ecosystem was modified and mutated by migration using mutate probabilities. The elitism was implemented and the best habitat was output.

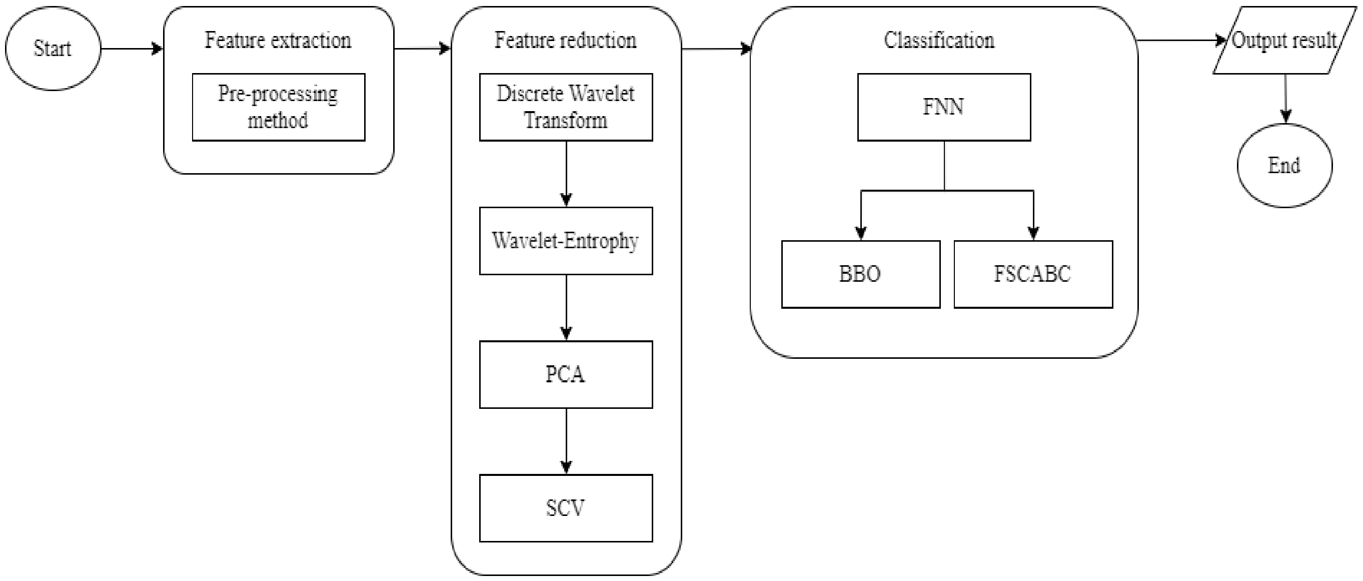


Figure 2.1.1.1: Flow chart of Wavelet-Entropy and Feedforward Neural Network

2-1-2 Fruit Classification using Multiclass SVM

Done by: Adrian Shen Loong A/L Seng Yean

In this classification, a dataset consists of 1653 images of 18 different fruit was used as an input for the experiment. To prepare the dataset, an image segmentation technique called Split-and-Merge Algorithm was used to separate the fruit from the background. First, the image was split into four son-squares if it is inhomogeneous. If the four son-squares are homogeneous, they will then be merged together. This process if repeated until no it can no longer be split and merged. Figure 2-1-2-1 shows the process with comparison with Otsu’s method.

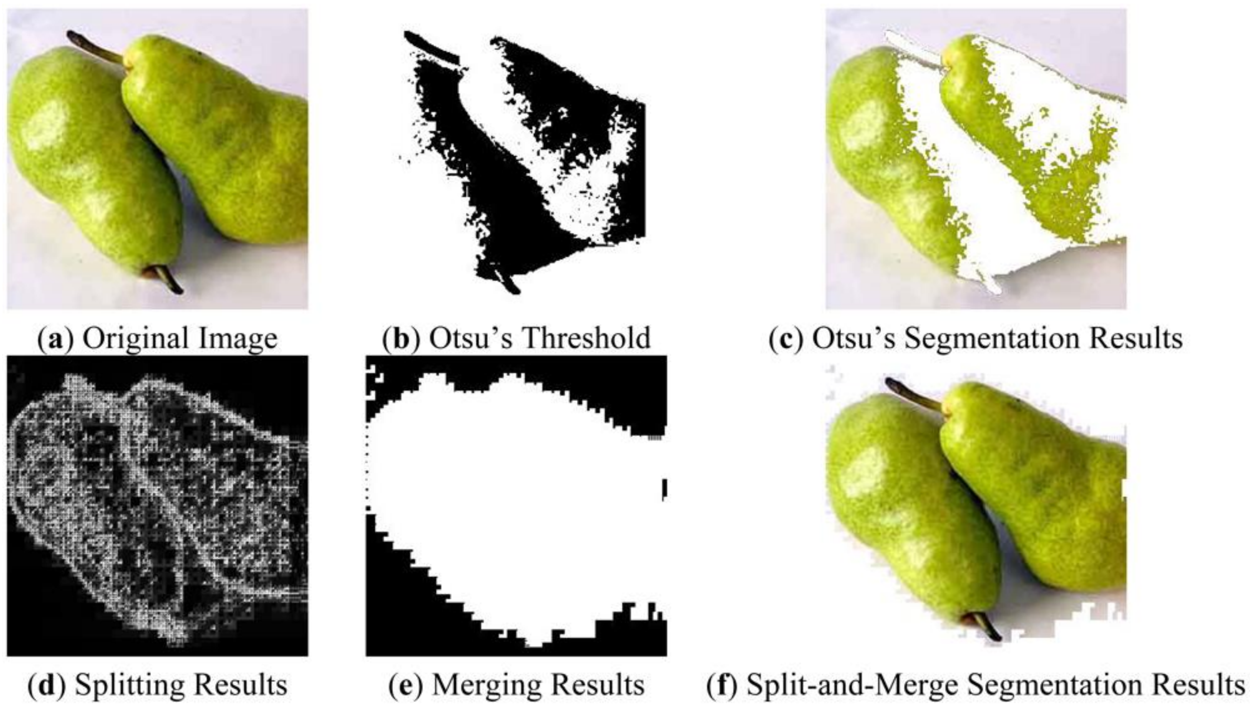


Figure 2-1-2-1 Comparison of Otsu’s and Split-and-Merge method

Next, a hybrid classification system to separate the color, appearance and shape of the fruit. To separate the colors, a color histogram was used to indicate the color distribution of the fruit images. This method compares histogram signatures between two images and match the color content with one another. Unser feature vector was then used to differentiate the texture of the images. To extract features of the shape, eight measures which was convex hull, convex area, solidity extracted with Graham Scan method, minor length, major length using an ellipse, area, perimeter and Euler number which was taken directly from the object.

The classification was then done on 4 different SVM: Kernel SVM, Winner-Takes-All SVM, Max-Wins-Voting SVM and Directed Acyclic Graph SVM. Out of 1653 images 1322 was used to train the models, the remaining 331 was used to test the models. Next, 5-fold cross validation was used to evaluate the average error of each models and the performance was analyzed on a Confusion Matrix. If the results are unsatisfiable, the model will be retrained and reevaluated until acceptable result was presented. Figure 2-1-2-2 shows the flow on how the model was trained and evaluated where the input is the image of the fruit and the output is the confusion matrix of the models.

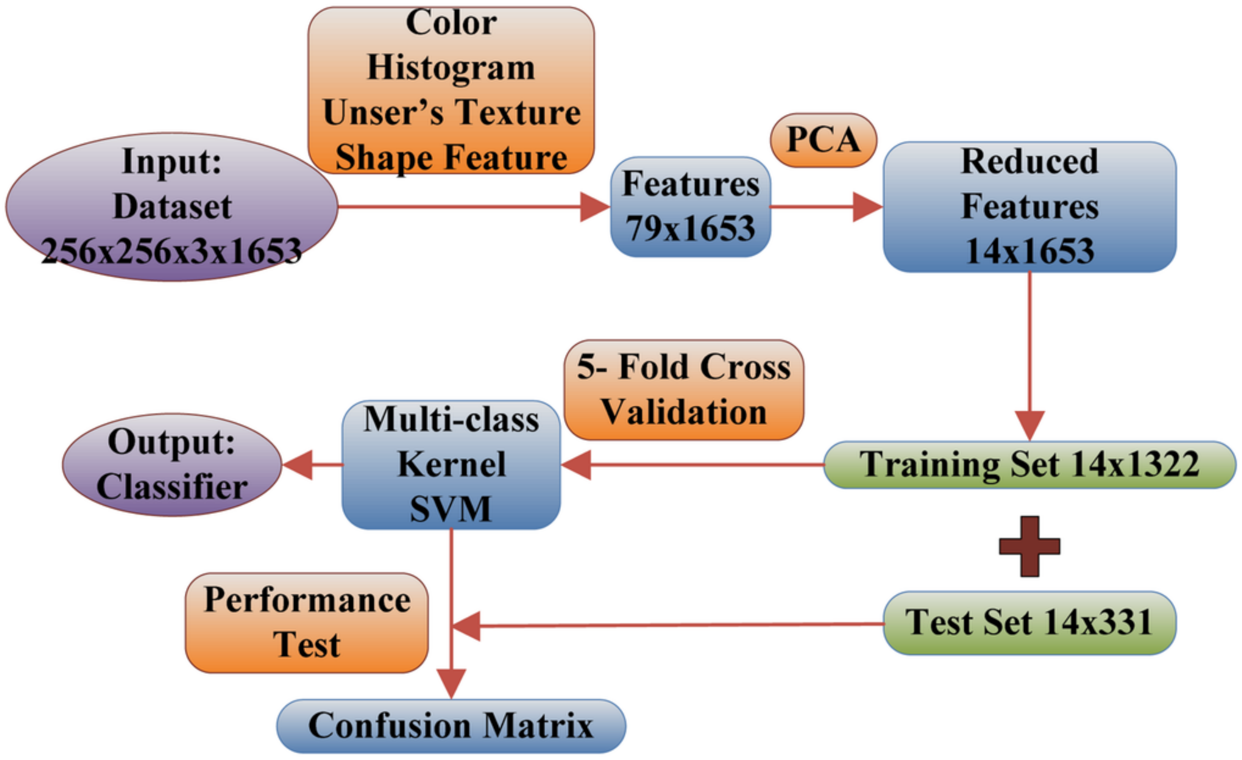


Figure 2-1-2-2 Flow Diagram

2-1-3 Fruit Recognition Based On Convolution Neural Network

Done by: Wan Kar Hou

An approach to classify fruit using convolution neural network (CNN). Convolution Neural network is algorithm with capability of assigning important various aspects from any input image and differentiate one from another. The stages of this approach are shown in Figure 2-1-3-1. At first, input of fruit images is process to selective search algorithm to obtain around 20 regions from each fruit images. Some of the regions removed based on certain criteria such as any regions with length or width are 0.2 times less than the original length or width and calculated entropy of each region is less than 6.77. Eventually, useful regions of fruit images are proceeded to next process. Extracted regions resized to 32x32, follow by subtracting mean activity over the training set from each pixel. Next, RGB values of the pixels in each region extracted out and respective categories labeled with the name of the specific type of fruit. After labels and regions are well prepared, it will proceed as inputs to train the convolution neural network using stochastic gradient descent with parameters such as momentum of 0.9, learning rate of 0.0001, and weight decay of 0.0005. Parameters of convolutional neural network are adjusted to obtain desired result. Finally, the type of the fruit image identified by referring categories with the highest numbers of votes obtained in all regions involved.

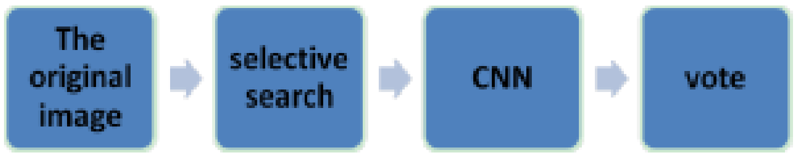


Figure 2-1-3-1 The flow chart of the fruit algorithm

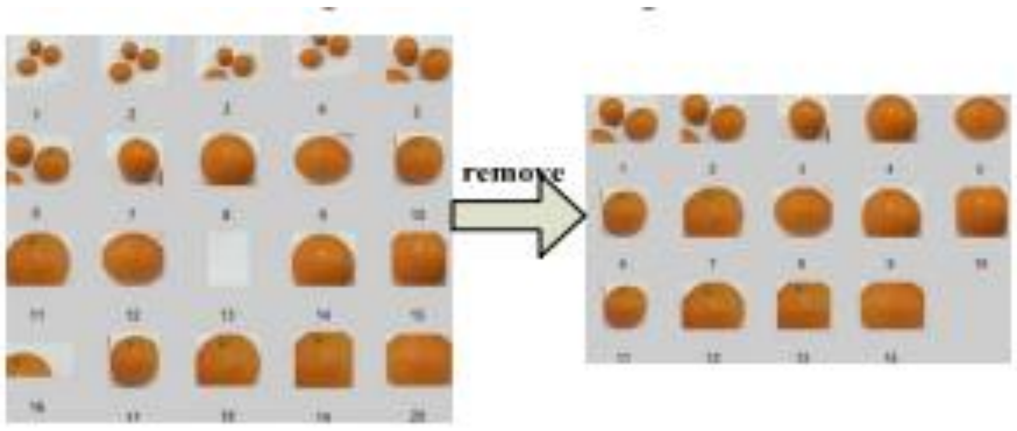


Figure 2-1-3-2 Result of optimally selected regions

2-1-4 Fruit Classification Using Computer Vision and Feedforward Neural Network

Done by: Ooi Chin Hao

An approach to classify fruit by mixing fitness-scaled chaotic artificial bee colony (FACABC) algorithm and feedforward neural network (FNN) (Zhang et al., 2014). FNN is a classifier that can classify non-linear separable patterns and approximate an arbitrary continuous function. At first, a digital camera is used to obtain the fruits images. Then, the fruits images are preprocessed using split-and-merge algorithm. The fruits images are segmented from 1024x768 to 256x256.

Next, color, texture, and shape features are extracted from the fruits images to reduce the dimensions of the images. Principal component analysis (PCA) is used to further reduce the features and those features with less than 95% of original features are removed. After that, the data is divided into a training set and a testing set. Stratified K-fold cross validation is applied on the training set to enhance the generation capability of FNN.

In the next section, training data is submitted to the FNN classifier. After the classification using FNN classifier, the median square error (MSE) of the difference between output value and target value is obtained. The average MSE, which is equal to the fitness function, is then used by the FSCABC algorithm to optimize the weights for FNN classifier. The last step is to use the trained classifier to calculate the confusion matrix using testing set.

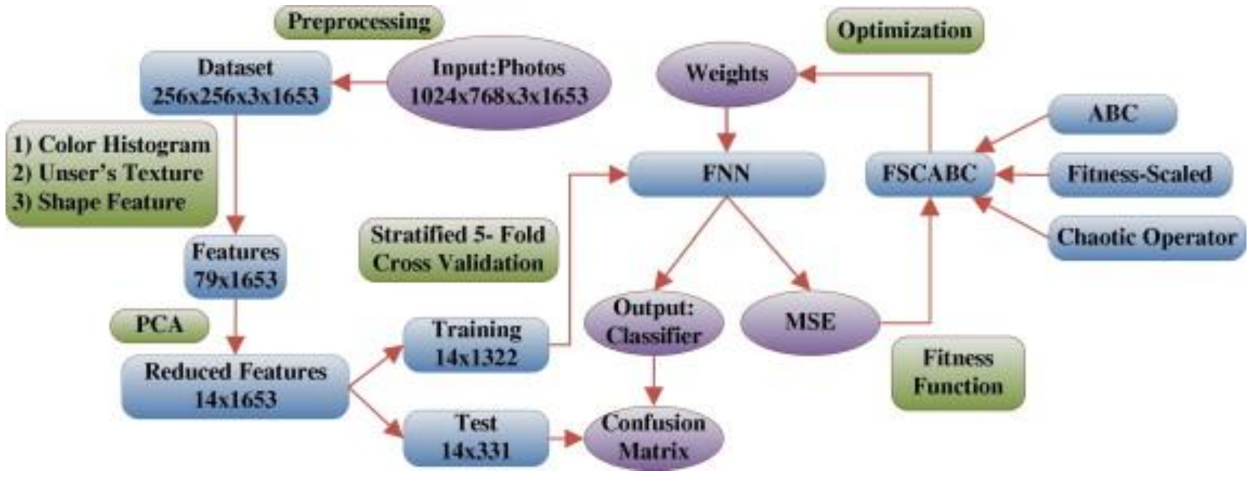


Figure 2-1-4-1 Flowchart of the processing steps

2-2 Advantages and Disadvantages

Done by: All Members

The advantage of these combination technique presented in 2-1-1 is saving more cost on spending in sensors because it did not require any expensive sensors such as invisible light sensor, chemical sensor or heat sensor. Next, it consumed less time for the result by using BBO techniques compare to FSCABC techniques. BBO techniques used 26s while FSCABC used 31s to obtain the results. This is because BBO is an efficient swarm-intelligence method. Furthermore, it obtained high accuracy which is 89.5%.

The disadvantage of these combination technique is it cannot detect the fruits that are dries, canned, sliced or partially covered. Besides, it had small data size of fruits. It only had 1653 images of 18 different fruit classes. Next, the images of fruit required to set up nicely and center the fruit. This is because they used four-step pre-processing method rather than convolutional neural network (CNN) technique in this research. CNN is suitable for big data size and the images no need to set up nicely.

The techniques presented in 2-1-2 allows the system to effectively recognize fruits by their colors, shape and texture. This was largely due to the combination of different combination methods used to extract the features. Next, different models were used to perform image classification and the model with the highest accuracy was chosen. This method allows the researchers to ensure that their results are as accurate as possible since one model might be less suitable for the problem at hand. However, such techniques are very time consuming since extracting features and training models in a large dataset requires much more preparation, processing time and analysis than using a single method and model.

One of the strengths of approach method in 2-1-3 is CNN could perform pattern classification and feature extraction simultaneously which get rid of time wasted compared to others approach. Besides that, CNN is excellent in learning ability compared to others approaches, it could learn essential features from any input images and provide a more accurate classification result. On the other hand, one of the weakness for the approach method in 2-1-3 is the approach method did not take in external environment such as light as considerations. As in this approach method does not consists of noise removal method such as Gaussian Blurring which could remove noise in images, if the lightness are too bright or dim, it will affected the quality of images and eventually leads to low accuracy.

One of the advantages in 2-1-4 approach is that it has a simple structure and easy to implement. Besides, FSCABC-FNN acquired 89.1% of classification accuracy in experimental results, which was higher compared to other existing algorithms such as GA-FNN, PSO-FNN, ABC-FNN, and kSVM. On the other hand, the high classification accuracy of FSCABC-FNN is only limited to specific fruits. For instances, the classification accuracy of Hass avocados, passion fruits, and red grapes are all lower than 80% in the experimental results.

CHAPTER 3: SYSTEM DESIGN

3-1-1 System Creation

Done by: Adrian Shen Loong A/L Seng Yean

To ensure the smooth creation of the system, our system will follow the procedure in the images shown below. Figure 3-1-1 and Figure 3-1-2 shows the flow on the implementation of the system.

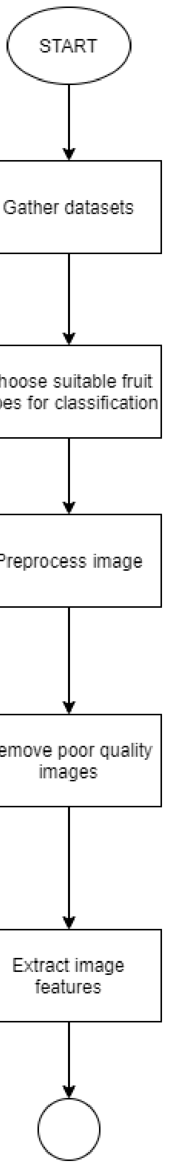

START
Gather datasets
Choose suitable fruit types for classification
Preprocess image
Remove poor quality images
Extract image features

Figure 3-1-1

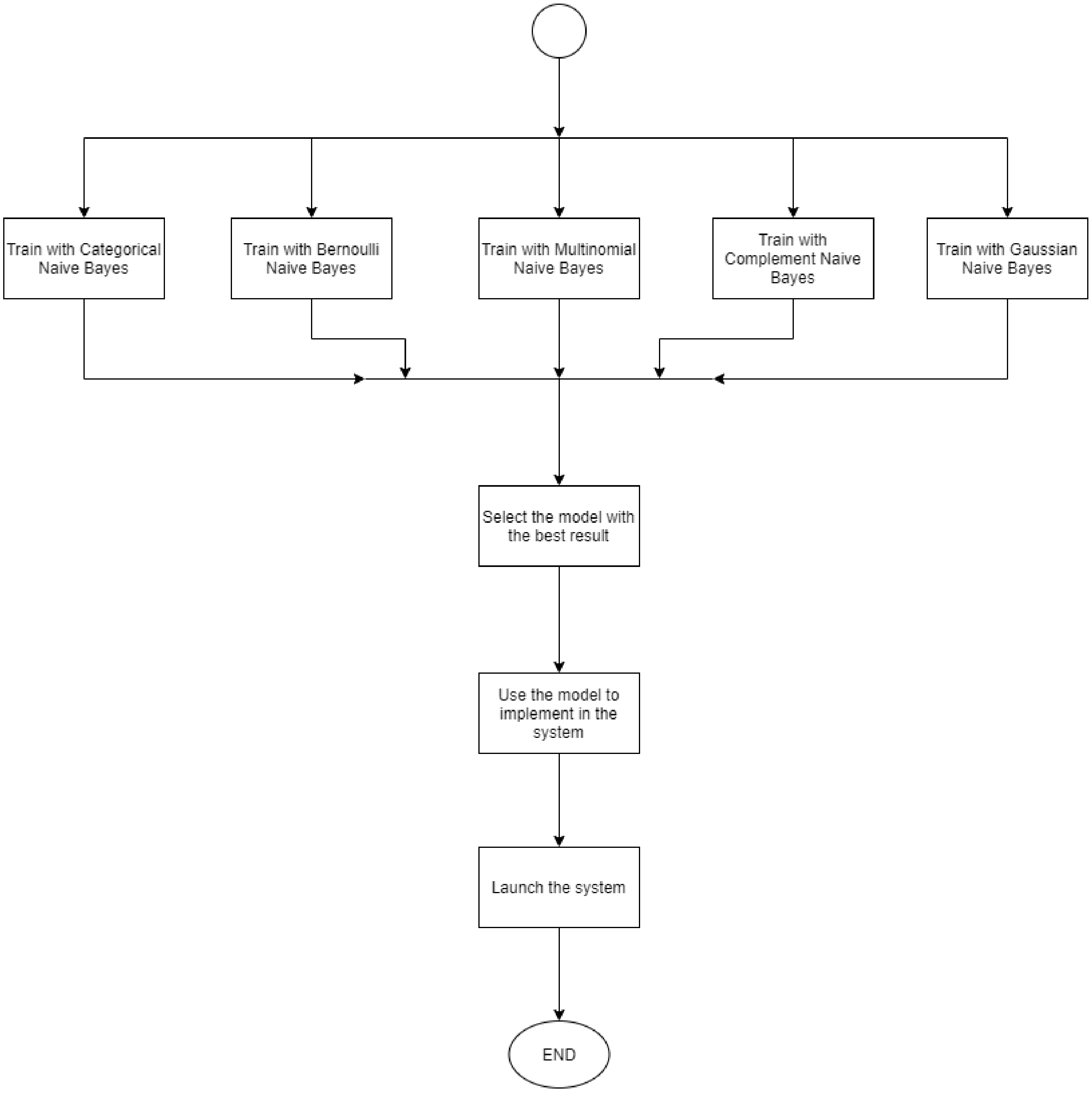

Train with Categorical Naive Bayes
Train with Bernoulli Naive Bayes
Train with Multinomial Naive Bayes
Train with Complement Naive Bayes
Train with Gaussian Naive Bayes
Select the model with the best result
Use the model to implement in the system
Launch the system
END

Figure 3-1-2

We first select 14 fruit images from a pool of datasets namely, Apple Crimson Snow, Apple Granny Smith, Apple Pink Lady, Avocado, Grape Blue, Grape Pink, Grape White, Lemon, Lemon Meyer, Lychee, Mango, Mango Red, Orange and Papaya. Then, we will preprocess all the images using Gaussian blur to remove the noise and perform segmentation on all the images to help the models in recognizing the fruit. After segmentation, we will remove any poorly preprocessed images in the dataset as it will cause inaccuracies in the result later on. Finally, we will extract the color features from the image and the dataset will be used to train the models.

Next, we will perform training and testing on 5 different models and select whichever model performs the best. After evaluation, the model will be embedded into the system and release for the users.

3-1-2 Use-Case Diagram

Done by: Adrian Shen Loong A/L Seng Yean

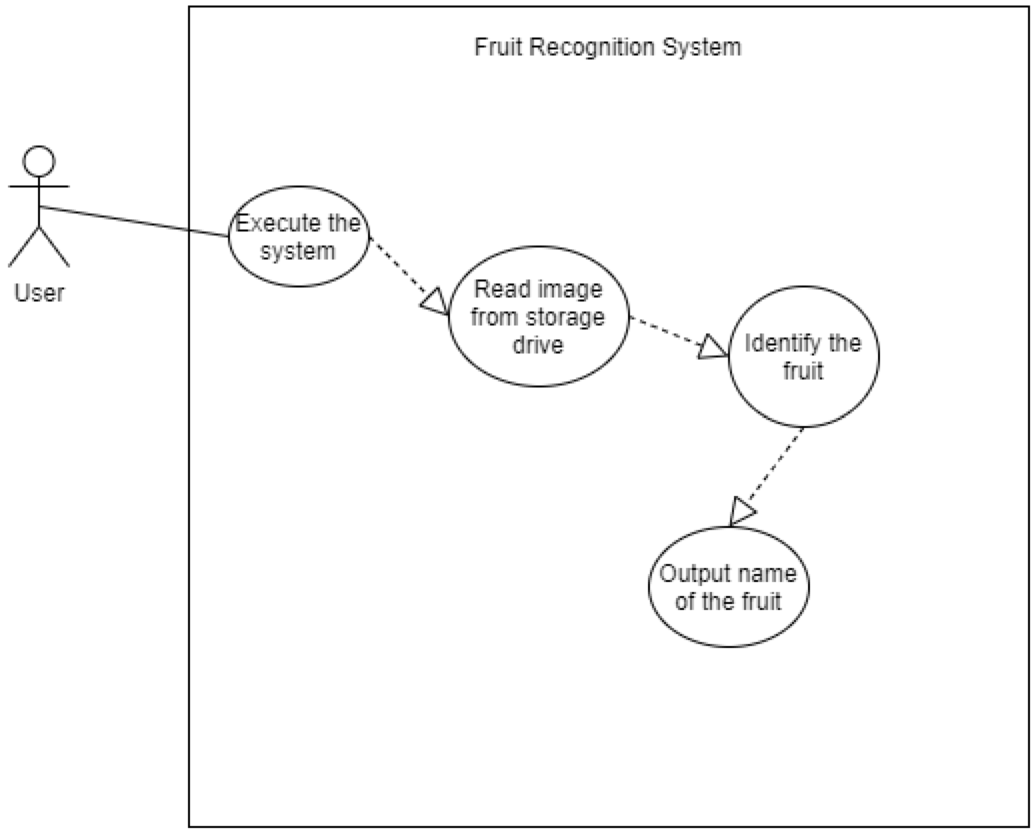


Figure 3-2 Use-case diagram

Figure 3-2 shows how will the system works when the user uses the system. After launching the system, the user will be required to import the fruit image they wish to identify into the system. The system will then attempt to identify the fruit and provide feedback to the user by displaying the fruit name.

3-1-3 Hardware and Software:

Done by: Wan Kar Hou

<u>Hardware:</u>

1. Laptop

Operating System	Windows 10
Processor	Intel Core i7 7th Gen 7Y75
RAM	8GB
Graphic Card	Intel Integrated HD Graphics

Table 3-1-2-1: Laptop Hardware Specification

2. Logitech C920 Webcam

Photos	15.0MP
Video	1920 x 1080
USB Type	USB 2.0
System Requirements	1 GHz processor or faster. 512 MB RAM or more. 200 MB hard drive space
OS Support	Windows XP, Windows Vista, or Windows 7

Table 3-1-2-2: Webcam Hardware Specification

<u>Software:</u>

Some of the software selected for the development of this system were :

1. Microsoft Visual Studio 2017

   Microsoft Visual Studio is an integrated development environment which used to develop certain types of program through programming language such as C++, C# and others.

2. Open CV

   Open CV is an online open source library which widely used in developing real-time computer vision.

   Open CV supports:

   i) Real-time captured

   ii) Import of video file

   iii) Detection of objects

3. Python IDE

   Python offers many kinds of libraries and supports many machine learning packages such as OpenCV and TensorFlow which reduces developing time.

3-1-4 System Performance Definition:

Done by: Wan Kar Hou

High accuracy was required to detect the image of fruit under certain kind of situation and complex background.

In our proposed system and original version of system, Naïve Bayes classifier applied to perform classification of fruit. Naïve Bayes classifier was probabilistic in nature as it every pair of features being classified is independent of each other, thus it shall provide a better accuracy rate on the independent features such as shape, color and others which will be extracted in this system. The performance of the system evaluated based on the accuracy rate of the system. The performance of the system considered acceptable if it achieved over 80% of accuracy rate while considered not acceptable if the accuracy rate was below 80%.

Some of main changes of proposed system over original version of system were additional of Gaussian Blurring and Hu Moment. Gaussian Blurring applied on image preprocessing to remove image noise and allow system to extract the actual part of fruit during segmentation process. Hu moment applied describe and characterize shape of an object in an image. These additional modules could improve the accuracy on recognition of fruit, eventually leads to higher performance of system.

Major target of improvements for proposed system over original version of system was accuracy on classifying fruit. Best result obtained using the original version of system with a classification accuracy of 89.1 %. It was predicted that the accuracy of the new version system could achieve 92 % and above.

3-2 Implementation Issues and Challenges

Done by: Ooi Chin Hao

There are some issues and challenges found during the implementation of the system. The very first issue found is the present of image noise. Most of the fruit images input are taken by a common camera, which means that the images' quality would not be too high. Hence, the fruit images probably contain a lot of noise which will then affect the system output. Although the preprocessing techniques implemented in the system resolved some of the image noise, a more robust technique is required to further minimize the image noise and improve the system performance.

Other than that, another issue was found during the segmentation process of fruit images. The segmentation process works desirably when applying on fruits with smooth surface such as grape, mango, and orange. When it is applied on fruits with rough surface, which are fruits like carambola and pineapple that have more details on their skin, the output images will have missing parts.

3-3 Timeline

Done by: Mabellyn Liew LinZhen

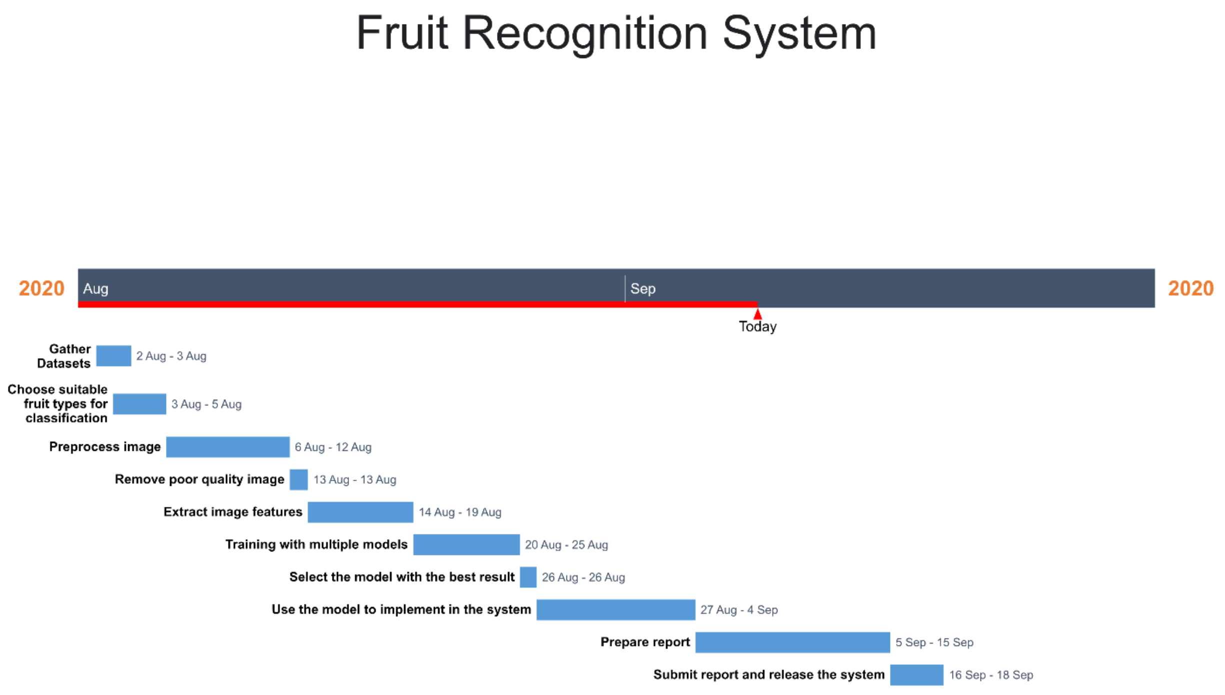


Figure 3-4-1 Gantt Chart

Chapter 4 Image Preprocessing

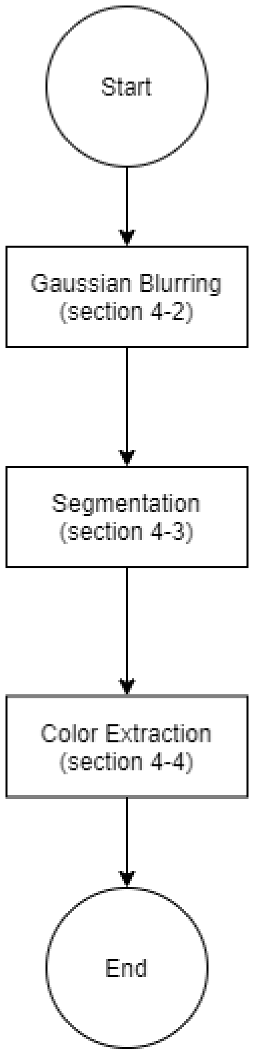

Start
Gaussian Blurring (section 4-2)
Segmentation (section 4-3)
Color Extraction (section 4-4)
End

Figure 4-1 Image preprocessing flowchart

4-1 Image Dataset

Done by: Mabellyn Liew LinZhen

14 types of fruit image had been chosen as training and testing dataset in this project, which include Apple Crimson Snow, Apple Granny Smith, Apple Pink Lady, Avocado, Grape Blue, Grape Pink, Grape White, Lemon, Lemon Meyer, Lychee, Mango, Mango Red, Orange, and Papaya. Each of these fruit subsets contains around 400 sample images. Fruits that contain more details on their surface like Carambola and Rambutan are not chosen as dataset. This is because the preprocessing techniques implement in this project are more suitable to apply on fruit with smooth surface. Some of the fruits are perfectly fit to their images and result in failure of edge detection. One line of code stated below is inserted at the beginning of the preprocessing code to add a white border in the fruit image and solved the problem.

```
copyMakeBorder(srcI, srcI, 4, 4, 4, 4, BORDER_CONSTANT, Scalar(255, 255, 255));
```

4-2 Gaussian Blur

Done by: Mabellyn Liew LinZhen

Gaussian blur technique is used to decrease the noise and detail of fruit images. The figure below shows before and after applying Gaussian blur on Apple Crimson Snow image.

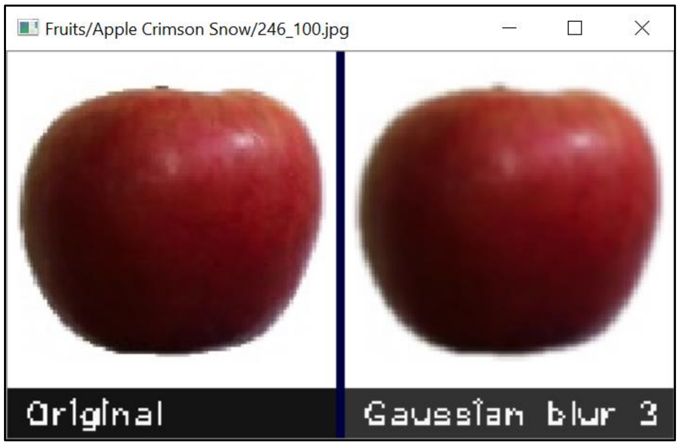


Figure 4-2-1 Gaussian blur

4-3 Segmentation

Done by: Ooi Chin Hao

The output of the Gaussian blur implementation will be the input of segmentation. Edge detection is employed to segment the blurred fruit image using Canny edge detector. The fruit image is then undergone dilation to eliminate any possible gap at the fruit boundary. Next, an opposite erosion is applied on the fruit image to recover the original boundary of the fruit. All the contours of the fruit are converted to different color to differentiate among each other. After that, the longest contour of the fruit will be identified. As a desired result, the longest contour should be the outer boundary of the fruit. The center of fruit is then computed and be used to do white color filling in the fruit boundary to form the mask of fruit. The last step of segmentation is to apply erosion on the mask of fruit and combine the output image with the original blurred image. Figure below show all the output image for each of the step in segmentation.

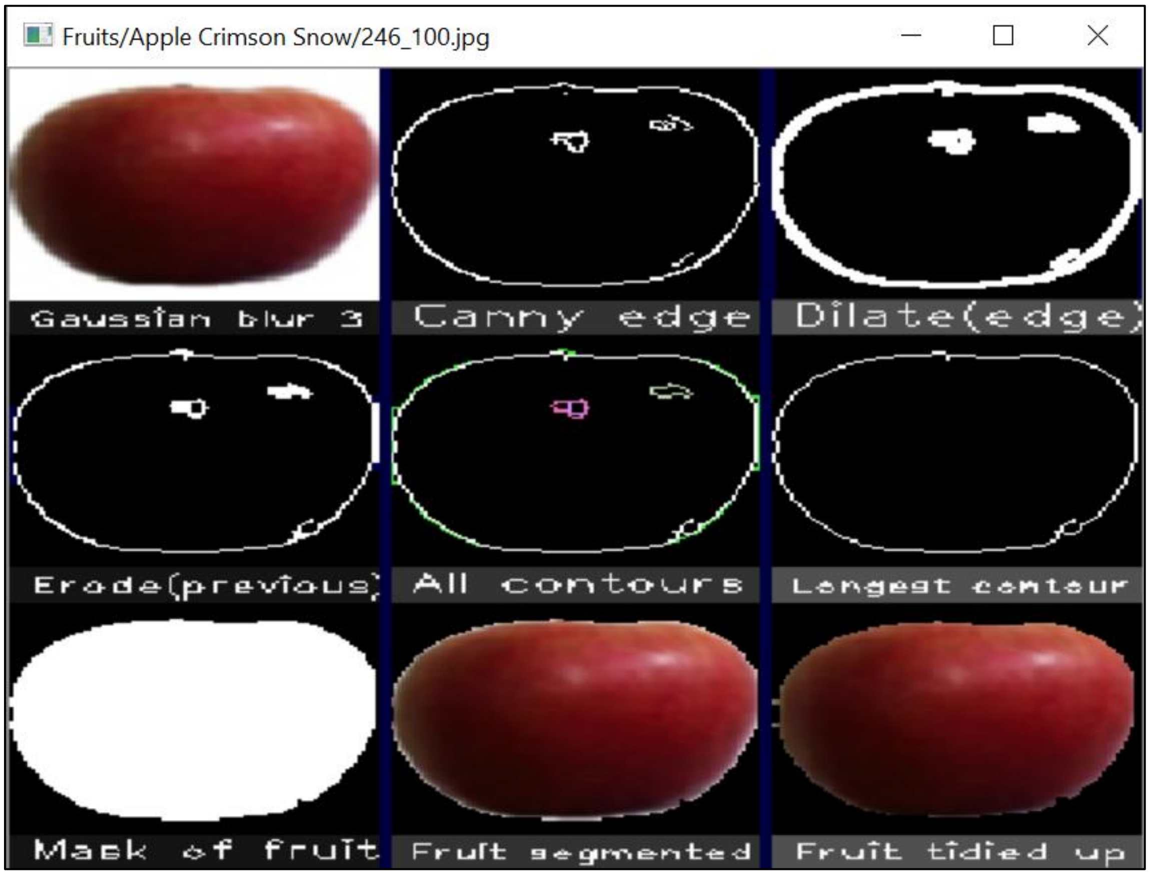


Figure 4-3-1 Segmentation

4-4 Hu Moment

Done by: Wan Kar Hou

Output of segmented image will be the input of Hu moment implementation. Seven moments derived by Hu for shape description. The first 6 moments will be invariant to scale, translation, reflection and reflection. While the 7th moment's sign changes for image reflection. The value of each moments extracted out as features to help in shape identification for fruits images.

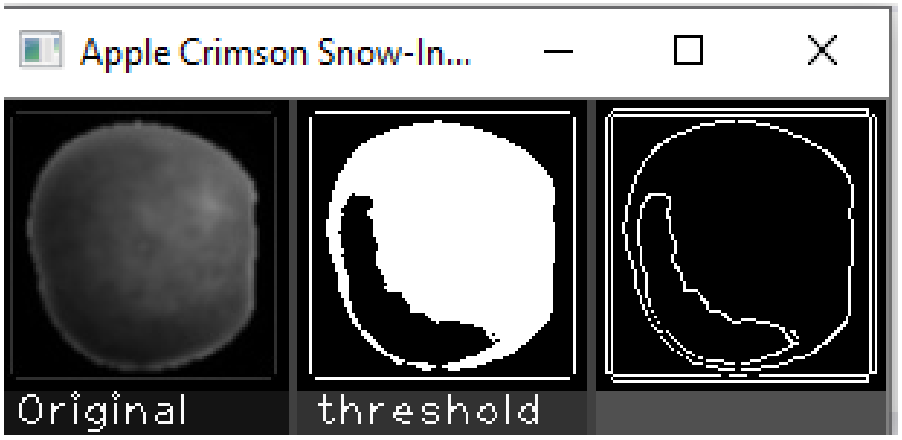
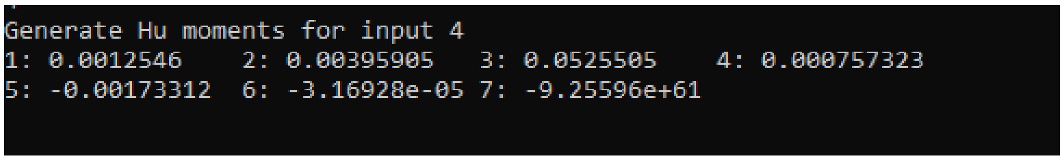
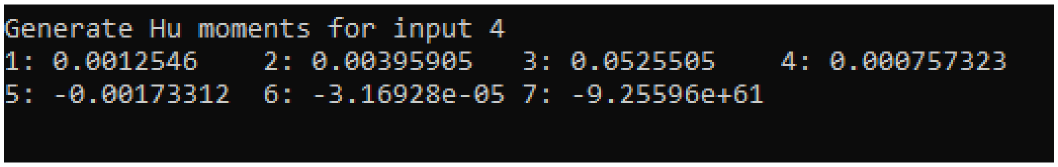


Figure 4-4-1 Original image and threshold



```
Generate Hu moments for input 4
1: 0.0012546    2: 0.00395905   3: 0.0525505    4: 0.000757323
5: -0.00173312  6: -3.16928e-05 7: -9.25596e+61
```

Figure 4-4-2 Value of Hu Moment from region



```
Generate Hu moments for input 4
1: 0.0012546    2: 0.00395905   3: 0.0525505    4: 0.000757323
5: -0.00173312  6: -3.16928e-05 7: -9.25596e+61
```

Figure 4-4-3 Value of Hu Moment from edge

4-5 Extracting Color Features

Done by: Adrian Shen Loong A/L Seng Yean

Since many fruits have similar shapes and size, relying solely on these features are unreliable. Hence, to be able to identify accurately, the system must also be able to differentiate fruits based on their colors.

To do that, we extract the color features of the fruit as in hue and saturation while generate the color histogram for the system to identify the fruit. Color histogram counts the number of similar pixels in an image. The algorithm will then generate histogram values for that image and store them in a database to be used for future reference. When a new fruit image is introduced into the system, it will then reference the histogram values of the image from the database to identify what type of fruit it is. The figures below show the color features in HSV and color histograms for an apple.

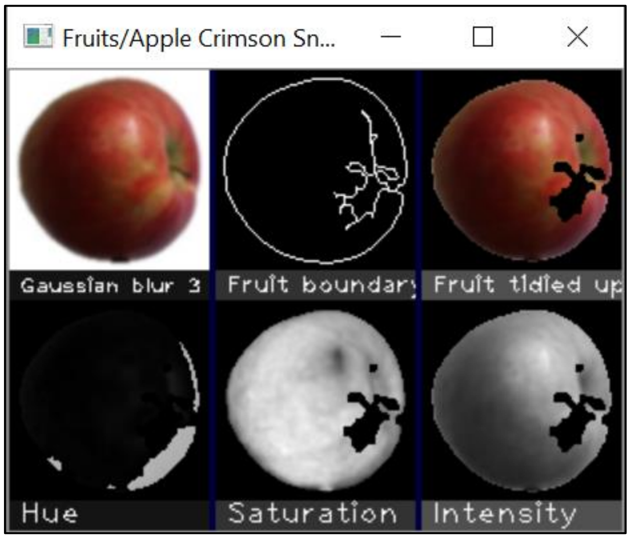


Figure 4-5-1 Color features in HSV

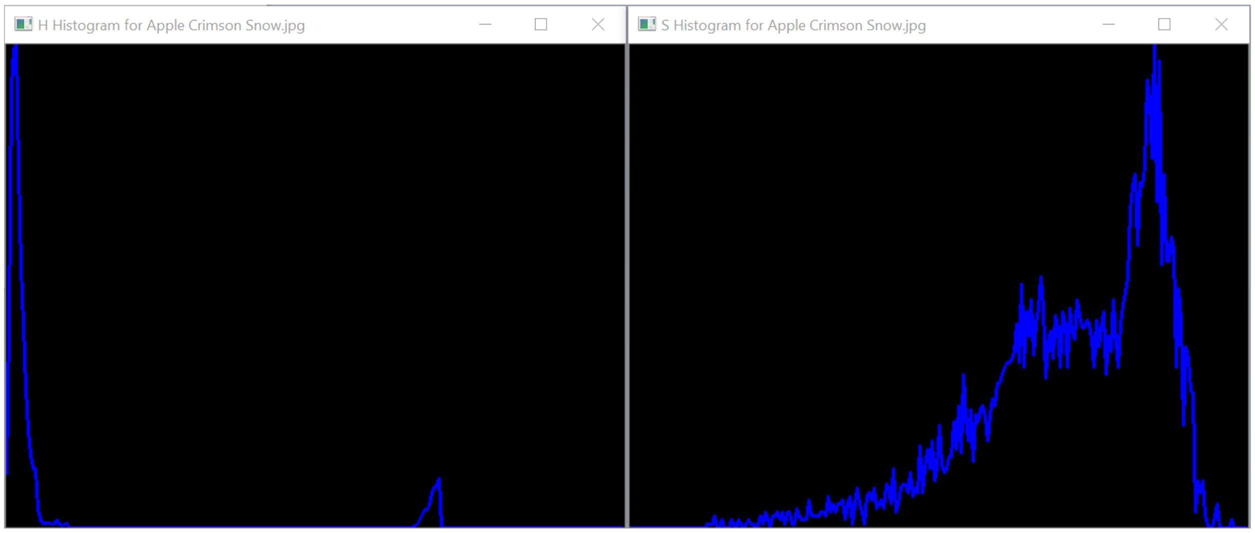


Figure 4-5-2 Color histograms

Chapter 5 Model Training

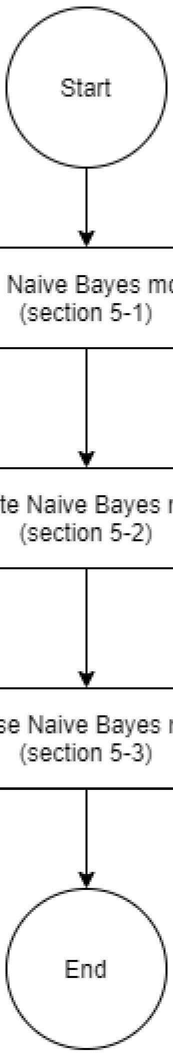

Start
Train Naive Bayes models (section 5-1)
Evaluate Naive Bayes models (section 5-2)
Choose Naive Bayes model (section 5-3)
End

Figure 5-1 Model training flowchart

5-1 Train Naïve Bayes models

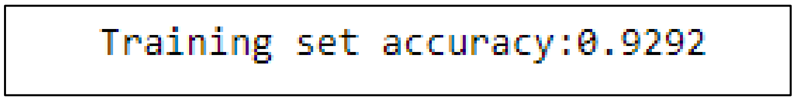
Done by: Wan Kar Hou

Data from image processing was seperated to training set and testing set with ratio of 80:20 . Different models of Naives Bayes was trained using training set to gauge its accuracy.

5-1-1 Training Set Accuracy of Naïve Bayes Models

i) Training set accuracy of Gaussian Naïve Bayes model

The accuracy of the prediction on training set with Gaussian Naïve Bayes model reach 92.92% as show in Figure 5-1-1-1

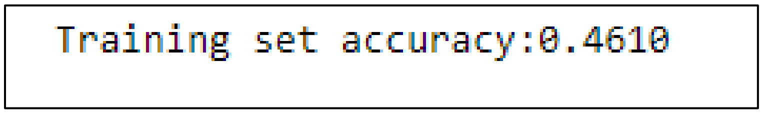


```
Training set accuracy:0.9292
```

Figure 5-1-1-1 Training set accuracy of Gaussian Naïve Bayes model

ii) Training set accuracy of Complement Navie Bayes model

The accuracy of the prediction on training set with Complement Naïve Bayes model reach 46.10% as shown in Figure 5-1-1-2

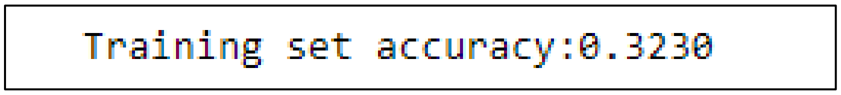


```
Training set accuracy:0.4610
```

Figure 5.1.1.2 Training set accuracy of Complement Naïve Bayes model

iii) Training set accuracy of Multinominal Naïve Bayes model

The accuracy of the prediction on training set with Multinomial Naïve Bayes model reach 32.30% as shown in Figure 5-1-1-3.

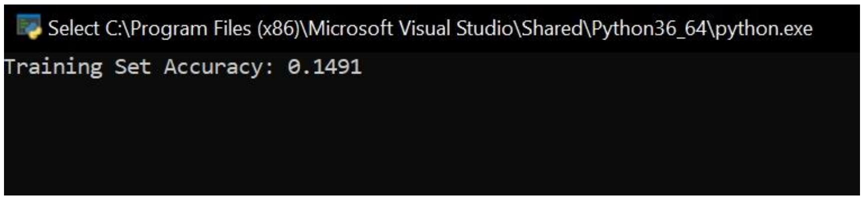


```
Training set accuracy:0.3230
```

Figure 5-1-1-3 Training set accuracy of Multinomial Naïve Bayes model

iv) Training set accuracy of Categorical Naïve Bayes classifier

The accuracy of the prediction on training set with Categorical Naïve Bayes model reach 14.91% as shown in Figure 5-1-1-4

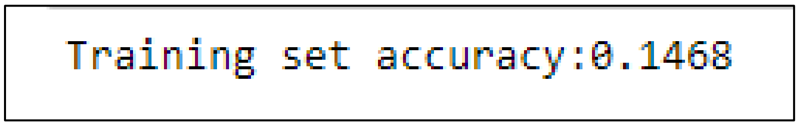


```
Select C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Training Set Accuracy: 0.1491
```

Figure 5-1-1-4 Training set accuracy of Categorical Naïve Bayes model

v) Training set accuracy of Bernoulli Naïve Bayes classifier

The accuracy of the prediction on training set with Bernoulli Naïve Bayes model reach 14.68% as shown in Figure 5-1-1-5



```
Training set accuracy:0.1468
```

Figure 5-1-1-5 Training set accuracy of Bernoulli Naïve Bayes model

5-1-2 Comparison on Training Set Accuracy of Naïve Bayes Models

By comparing the training set accuracy of each Naïve Bayes model as shown in Table 5-1-2-1, Gaussian Naïve Bayes model obtain the highest training set accuracy of 92.92% while Bernoulli Naïve Bayes model obtain the lowest training set accuracy of 14.68%.

Naïve Bayes Model	Training set accuracy (%)
Gaussian	92.92
Complement	46.10
Multinomial	32.30
Categorical	14.91
Bernoulli	14.68

Table 5-1-2-1 Training Set Accuracy Comparison

5-2 Evaluate Naïve Bayes models

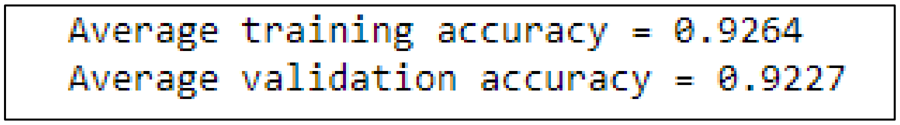
Done by: Ooi Chin Hao & Mabellyn Liew LinZhen

Each Naïve Bayes model evaluated by referring the result of 5- fold cross validation for each Naïve Bayes model. 5-fold cross validation performed using training datasets. The result of 5- fold cross validation will be the average training accuracy and the average validation accuracy.

5-2-1 Results of 5-fold Cross Validation for Naïve Bayes Models

i) Result of 5-fold cross validation for Gaussian Naïve Bayes

Average training accuracy and average validation accuracy of 5-fold cross validation for Gaussian Naïve Bayes were 92.64% and 92.27%, as shown in Figure 5-2-1-1.

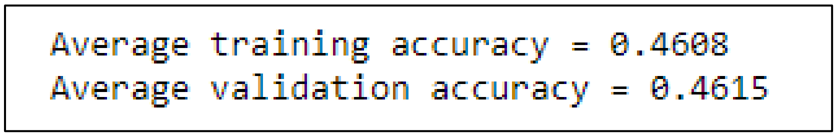


```
Average training accuracy = 0.9264
Average validation accuracy = 0.9227
```

Figure 5-2-1-1 Average training accuracy and validation accuracy of Gaussian Naïve Bayes model

ii) Result of 5-fold cross validation for Complement Naïve Bayes

Average training accuracy and average validation accuracy of 5-fold cross validation for Complement Naïve Bayes were 46.08% and 46.15%, as shown in Figure 5-2-1-2.

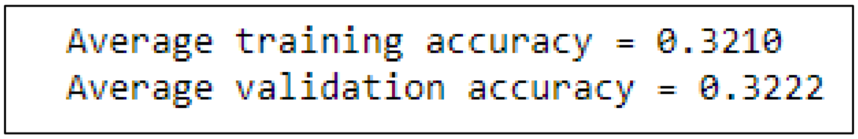


```
Average training accuracy = 0.4608
Average validation accuracy = 0.4615
```

Figure 5-2-1-2 Average training accuracy and validation accuracy of Complement Naïve Bayes model

iii) Result of 5-fold cross validation for Multinomial Naïve Bayes

Average training accuracy and average validation accuracy of 5-fold cross validation for Multinomial Naïve Bayes were 32.10% and 32.22%, as shown in Figure 5-2-1-3.

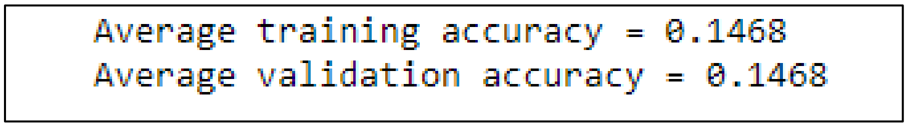


```
Average training accuracy = 0.3210
Average validation accuracy = 0.3222
```

Figure 5-2-1-3 Average training accuracy and validation accuracy of Multinomial Naïve Bayes model

iv) Result of 5-fold cross validation for Bernoulli Naïve Bayes

Average training accuracy and average validation accuracy of 5-fold cross validation for Bernoulli Naïve Bayes were 14.68% and 14.68%, as show in



```
Average training accuracy = 0.1468
Average validation accuracy = 0.1468
```

Figure 5-2-1-4 Average training accuracy and validation accuracy of Bernoulli Naïve Bayes model

5-2-2 Comparison on Results of 5-fold Cross Validation for Naïve Bayes Models

By comparing the average training accuracy and average validation accuracy from the result of 5-fold cross validation for each Naïve Bayes model as shown in Table 5-1-2-1, Gaussian Naïve Bayes model obtain the highest value for average training accuracy and average validation accuracy with the scores of 92.64% and 92.27%. On the other hand, Bernoulli Naïve Bayes model obtain the lowest value for average training accuracy and average validation accuracy with the scores of 14.68% for both accuracies.

Naïve Bayes Model	Average training accuracy (%)	Average validation accuracy (%)
Gaussian	92.64	92.27
Complement	46.08	46.18
Multinomial	32.10	32.22
Bernoulli	14.68	14.68

Table 5-2-2-1 Average Training Accuracy and Validation Accuracy Comparison

5-3 Choose Naïve Bayes model

Done by: Adrian Shen Loong A/L Seng Yean

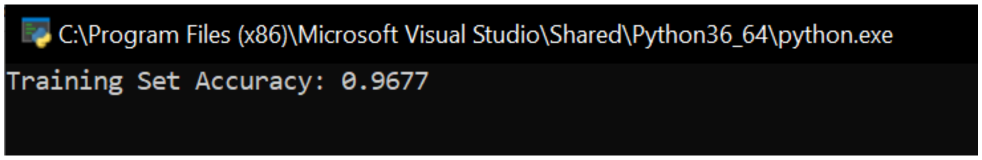
Main criteria to select most suitable Naïve Bayes model among all types of Naïve Bayes model is by referring training set accuracy and result of cross validation. Naïve Bayes model with the highest value for training set accuracy and highest value for average training accuracy, average validation accuracy in cross validation selected. Thus, Gaussian Naïve Bayes model selected as it scores highest value 92.92% in training set accuracy while score highest values 92.64% and 92.27% for average training accuracy and average validation accuracy in cross validation.

Chapter 6 Experimental Results

6-1 System Performance

Done by: Ooi Chin Hao

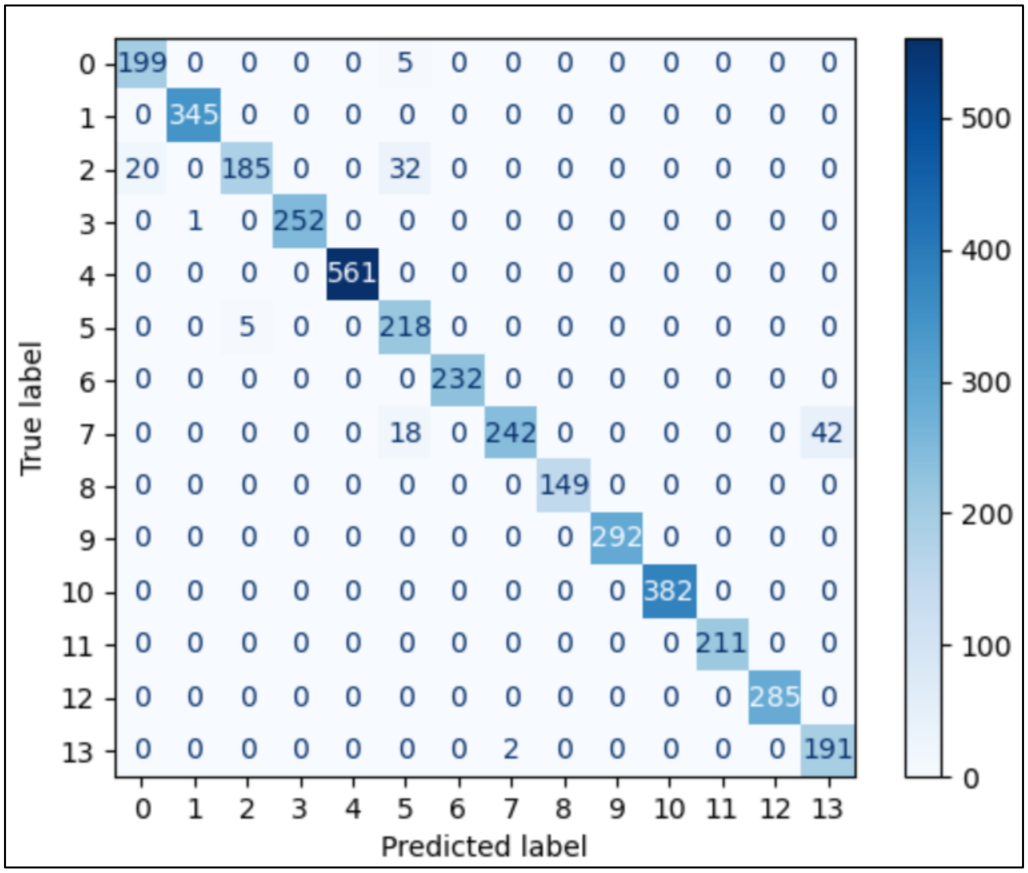
With the implementation of Gaussian Naïve Bayes classifier, the accuracy of the prediction on training set managed to reach 96.77% as shown in Figure 6-1-1.



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Training Set Accuracy: 0.9677
```

Figure 6-1-1 Accuracy of Training Set

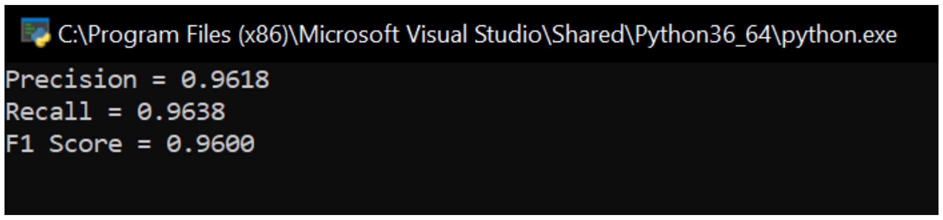
Next, a confusion matrix is plotted by using training dataset to build a graphical view of the result.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	199	0	0	0	0	5	0	0	0	0	0	0	0	0
1	0	345	0	0	0	0	0	0	0	0	0	0	0	0
2	20	0	185	0	0	32	0	0	0	0	0	0	0	0
3	0	1	0	252	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	561	0	0	0	0	0	0	0	0	0
5	0	0	5	0	0	218	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	232	0	0	0	0	0	0	0
7	0	0	0	0	0	18	0	242	0	0	0	0	0	42
8	0	0	0	0	0	0	0	0	149	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	292	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	382	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	211	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	285	0
13	0	0	0	0	0	0	0	2	0	0	0	0	0	191

Figure 6-1-2 Confusion Matrix

The precision, recall, and F1 score values are calculated as shown in Figure 6-1-3.



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Precision = 0.9618
Recall = 0.9638
F1 Score = 0.9600
```

Figure 6-1-3 Precision, Recall, and F1 Score

Then, Apple Crimson Snow was taken out from the label column to act as binary classification. The precision-recall graph of Apple Crimson Snow is plotted as shown in Figure 6-1-4.

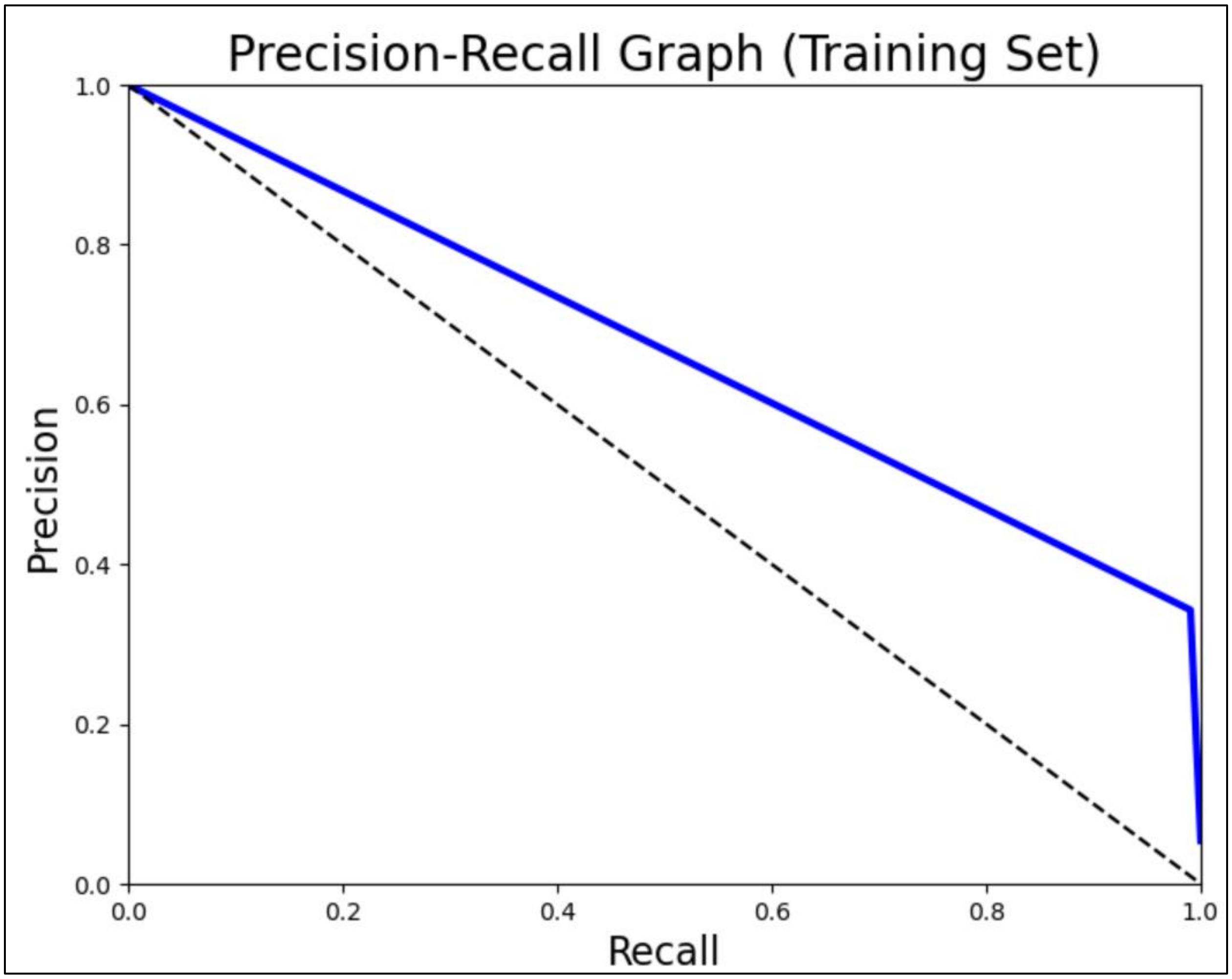


Figure 6-1-4 Precision/Recall Curve

The ROC curve of Apple Crimson Snow is also plotted with the value of area under curve stated at the bottom right of graph.

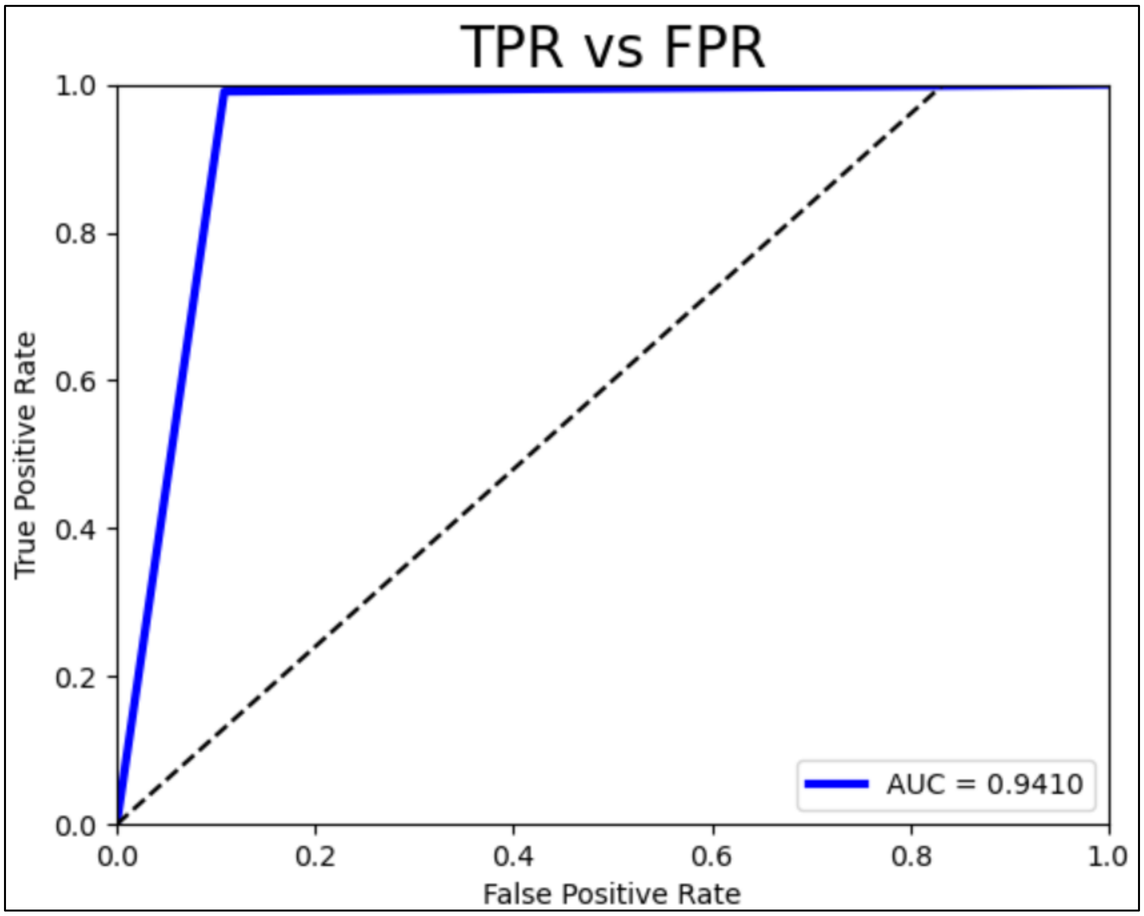
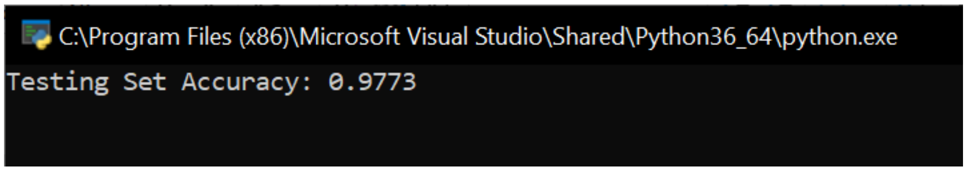


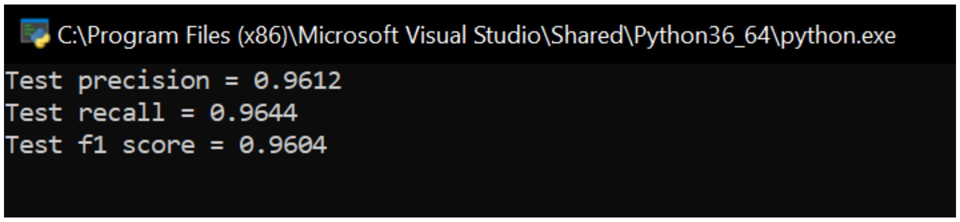
Figure 6-1-5 ROC Curve with AUC

As the output of the model trained is satisfiable, the model is used to make prediction on testing set. The accuracy of the prediction is 97.73%, which is quite similar with the accuracy of training set. No overfitting or underfitting occur.



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Testing Set Accuracy: 0.9773
```

Figure 6-1-6 Accuracy of Testing Set



```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Python36_64\python.exe
Test precision = 0.9612
Test recall = 0.9644
Test f1 score = 0.9604
```

Figure 6-1-7 Precision, Recall, F1 Score of Testing Set

6-2 Improvement

Done by: Mabellyn Liew LinZhen

In this section, we compared our GNB algorithm with GA-FNN, PSO-FNN, ABC-FNN, kSVM, and FSCABC-FNN algorithms. The classification accuracy of each algorithm is shown in Table 6-2-1. GNB algorithm achieved the highest accuracy among the others.

Algorithm	Classification accuracy (%)
GA-FNN Zhang, et al. (2014)	84.8
PSO-FNN Zhang, et al. (2014)	87.9
ABC-FNN Zhang, et al. (2014)	85.4
kSVM Wu (2012)	88.2
FSCABC-FNN Zhang, et al. (2014)	89.1
GNB (our)	97.7

Table 6-2-1 Classification Accuracy Comparison

Table 6-2-2 shows that both of FSCABC-FNN and BBO-FNN algorithms with 12 reduced features have a classification accuracy of 89.5%. GNB algorithm with 30 features has a classification accuracy of 97.7%.

Algorithm	# of Features	Accuracy
WE + PCA + FSCABC-FNN Wang, et al. (2015)	12	89.5%
WE + PCA + BBO-FNN Wang, et al. (2015)	12	89.5%
GNB (our)	30	97.7%

Table 6-2-2 Classification Accuracy Comparison 2

6-3 Error Analysis

Done by: Adrian Shen Loong A/L Seng Yean

From the previous confusion matrix results in section 6-1, most of the error is caused by the wrong prediction of fruit label #2 and label #7. Fruit label #2 is Apple Pink Lady. Out of 237 Apple Pink Lady training images, 20 images are wrongly predicted as Apple Crimson Snow and 32 images are wrongly predicted as Grape Pink. Fruit label #7 is Lemon. Out of 302 Lemon training images, 18 images are wrongly predicted as Grape Pink and 42 images are wrongly predicted as Papaya. This is mainly due to the feature extraction of our system. This system extracts hu moments, hue, and saturation of the fruit images as features to classify fruits. This method increases the occurrences of error when there are fruits with likely similar hu moments, hue or saturation values.

6-4 Contributions

Done by: Wan Kar Hou

This system increased the classification accuracy of fruits through incremental innovation. We implemented GNB algorithm to classify fruits in our system. We found that other researchers never apply this algorithm in fruit classification before. As shown in Section 6-1, our system manages to achieve good performance with its implementation. Although our system can only classify 14 types of fruit, it has the highest classification accuracy compared to other

Chapter 7 Conclusion

7-1 Project Review

Done by: Wan Kar Hou

In a nutshell, the problems of manual fruit classification are individual who suffered from color blindness and human errors. The individuals who suffered color blinds are difficult to differentiate the color of fruits and cannot recognize the type of fruit. Besides, there is a limitation when a task is done repeatedly by individuals. When the individuals feel tired, they may be categorizing the wrong fruit such as they accidentally put the poisonous food to the edible category.

The motivation of this project is to help individuals who are visually challenged. By using this system, individuals can easily differentiate the types of the fruits without getting help from others. The next motivation is to reduce human errors in the working environment. By using this system, the farmers can easily detect the fruits whether it is poisonous or edible. They also can categorize the fruits quickly to save time.

7-2 Discussions

Done by: Mabellyn Liew LinZhen

Since the work from literature review have used SVM and Neural Network for fruit classification, our system will implement Naïve Bayes classification to identify the fruit as it had not been done from our literature review. Another technique that was not used in previous work is the Gaussian blur. We use it is because the images might have many noises such as inconsistent background. Hence, it will be implemented in the system.

For other techniques, we will be reusing from previous literatures because those techniques are much more suitable for fruit classification whereas, other techniques might cause certain issues to the result. Techniques that will be carried forward to our work are segmentation algorithm, model evaluation (k-fold cross validation) and color feature extraction. The results show that the accuracy of Gaussian Naïve Bayes classifier is higher than previous work which is 97.7%. Wrong prediction of Apple Pink Lady as Apple Crimson Snow and Grape Pink occurred on this classifier. Wrong prediction of Lemon as Grape Pink and Papaya also occurred. This is due to their hu moments, hue or saturation values are similar and causes higher occurrences of error.

7-3 Novelties and Contributions

Done by: Ooi Chin Hao

One of the novelties and contributions of this project is the algorithm used for fruit classification. We implemented GNB classifier in the system to classify the types of fruit. Based on Chapter 2, previous researchers used FNN, SVM, and CNN classifiers to classify fruits in their system. We found that GNB classifier is an existing classifier that no one used in fruit classification before. Other than that, our system can classify 14 types of fruit, which are Apple Crimson Snow, Apple Granny Smith, Apple Pink Lady, Avocado, Grape Blue, Grape Pink, Grape White, Lemon, Lemon Meyer, Lychee, Mango, Mango Red, Orange, and Papaya, with the highest classification accuracy compared to other fruit classification systems.

7-4 Future Work

Done by: Adrian Shen Loong A/L Seng Yean

After testing the system and several discussions were made, there are several improvements that can be made to make the system perform even better:

1) Extend the classification to identify more types of fruits
2) Distinguish fruit and non-fruit object
3) Optimize the system so that it can run faster on lower end hardware
4) Expand the system to use other kinds of model to achieve more greater result
5) Accurately identify multiple fruits in one image.

References

Clinton Eye (n.d.). Color Blindness. Available at: https://www.clintoneye.com/color-blindness.html

Hou, L, Wu, QX, Sun, QY, Yang, H, & Li, PF 2016, 'Fruit recognition based on convolution neural network', 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pp. 18-22.

Ji, GL, Phillips, P, Wang, SH, Zhang, YD 2014, 'Fruit classification using computer vision and feedforward neural network', Journal of Food Engineering, vol. 143, pp. 167-177.

Seng, WC, Mirisaee, SH 2009, 'A new method for fruits recognition system', 2009 International Conference on Electrical Engineering and Informatics, pp. 130-134.

Wang, SH, Zhang, YD, Ji, GL, Yang, JQ, Wu, JG, Wei, L 2015, 'Fruit Classification by Wavelet-Entropy and Feedforward Neural Network Trained by Fitness-Scaled Chaotic ABC and Biogeography-Based Optimization', Entropy 2015, vol. 17, no. 8, pp. 5711-5728.

Yeow, JA, Ng, PK, Tan, KS, Chin, TS, Lim WY 2014, 'Effects of Stress, Repetition, Fatigue and Work Environment on Human Error in Manufacturing Industries', Journal of Applied Sciences, vol. 14, pp. 3464-3471.

Zhang, YD, Wu, LN 2012, 'Classification of Fruits Using Computer Vision and a Multiclass Support Vector Machine', Sensors 2012, vol. 12, no. 9, pp. 12489-12505.

Zubrinic, K, Milicevic, M, Zakarija, I 2013, 'Comparison of Naive Bayes and SVM Classifiers in Categorization of Concept Maps', International Journal of Computers, vol. 7, no. 3, pp. 109-116.

Appendices

I. Source code

FeaturestoCSV.cpp

```
#include        <opencv2/opencv.hpp>
#include        <opencv2/highgui/highgui.hpp>
#include        <opencv2/imgproc.hpp>
#include        <iostream>
#include        <fstream>
#include        <experimental/filesystem>
#include        <memory>
#include        "Supp.h"

using namespace cv;
using namespace std;

void drawHSVHistogram(Mat HSV, Mat V, char s[]) {
        // Below is the step to generate a histogram (frequency count) of the input image intensity
        // excluding the dark points
        /// Establish the number of bins
        int histSize = 8;
        ofstream file;
        file.open("data(All fruits).csv", ios::app);

        /// Set the ranges for counting, i.e. intensity outside of this range is not counted
        float range[] = { 0, 256 };
        const float *histRange = { range };

        bool uniform = true; bool accumulate = false;
        Mat             histogram;
        int             emptyCount; // The count of dark point or empty area

                                        // Compute the histograms: its input parameters are
                                        // HSV: the input image or images
                                        // 2nd input: number of images in srcI
                                        // 3rd input: list of channels in each image for the
        computation, the first channel is 0
                                        // 4th input: optional mask; if it is empty, no effect
                                        // 5th: output matrix
                                        // 6th: Histogram dimensionality, 1D histogram, 2D
        histogram, ...
                                        // 7th: Array of histogram sizes in each dimension
                                        // 8th: Array of the dims arrays of the histogram bin
        boundaries in each dimension
```

```
                                                                // 9th: Flag indicating whether the histogram is
            uniform or not, i.e. uniform have one
                                                                //      range in each dimension. Else it has multiple
            ranges with no count on out of range intensity.
                                                                // 10th: Accumulation flag. If it is set, the histogram
            is not cleared in the beginning
            calcHist(&V, 1, 0, Mat(), histogram, 1, &histSize, &histRange, uniform, accumulate);
            emptyCount = histogram.at<float>(0);
            calcHist(&HSV, 1, 0, Mat(), histogram, 1, &histSize, &histRange, uniform, accumulate);
            histogram.at<float>(0) -= emptyCount;
            for (int i = 0; i < 8; i++) { // This part is important for you to access the histogram value
            at each bin
                        file << histogram.at<float>(i) << ',';
            }

            file.close();
}

int segmentObject(Mat threeImages[3], char name[]) {
            Mat                         srcI = imread(name), tmp, cannyEdge, huMom;
            ofstream file;
            file.open("data(All fruits).csv", ios::app);

            if (srcI.empty()) {
                        cout << "cannot open " << name << endl;
                        return -1;
            }

            int const          noOfImagePerCol = 2, noOfImagePerRow = 3; // create a 2X3 window
            partition
            Mat                         largeWin, win[noOfImagePerRow * noOfImagePerCol], // create
            the new window
                        legend[noOfImagePerRow * noOfImagePerCol]; // and the means to each sub-
            window
            int                         winI = 0;

            int                         ratio = 3, kernelSize = 3; // set parameters for Canny
            Mat                         B = (Mat_<unsigned char>(3, 3) << 1, 1, 1,  // define Bel 1
                        1, 1, 1,
                        1, 1, 1);

            createWindowPartition(srcI, largeWin, win, legend, noOfImagePerCol,
            noOfImagePerRow);

            copyMakeBorder(srcI, srcI, 4, 4, 4, 4, BORDER_CONSTANT, Scalar(255, 255, 255));
            //add a white border
```

```
resize(srcI, srcI, Size(100, 100));

GaussianBlur(srcI, win[0], Size(3, 3), 0, 0); // copy the blurred input to the first
subwindow
putText(legend[0], "Gaussian Blur 3", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1); //
place text in the first legend window
Canny(srcI, cannyEdge, 60, 60 * ratio, kernelSize);
dilate(cannyEdge, cannyEdge, B); // Make edge thicker to fill gap/break
erode(cannyEdge, cannyEdge, B, Point(-1, -1)); // Make edge thinner to get back the
original edge as much as possible

vector<vector<Point> >      contours;
vector<Vec4i>                   hierarchy;

findContours(cannyEdge, contours, hierarchy, RETR_LIST, CHAIN_APPROX_NONE);
RNG                         rng(12345);
Scalar              color;
int                         index, max = 0;

srcI.copyTo(tmp);
tmp = Scalar(0, 0, 0);

for (int i = 0; i < contours.size(); i++) {
        if (max < contours[i].size()) { // Find the longest contour as fruit boundary
                max = contours[i].size();
                index = i;
        }
}
drawContours(tmp, contours, index, Scalar(255, 255, 255)); // highlight fruit boundary
tmp.copyTo(win[1]);
putText(legend[1], "Fruit boundary", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);

vector<Point> vp = contours[index]; //access each contour point
Point2i             p;
int                         count = 0;

p.x = p.y = 0;
for (int j = 0; j < vp.size(); j++) // Add all point coordinates
        p += vp[j];
p.x /= vp.size(); // take average, i.e. center of fruit
p.y /= vp.size();
floodFill(tmp, p, Scalar(255, 255, 255)); // fill inside fruit boundary
erode(tmp, tmp, B, Point(-1, -1)); // attempt to eliminate empty space around boundary
win[2] = win[0] & tmp;
putText(legend[2], "Fruit tidied up", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
```

```
	cvtColor(win[2], huMom, COLOR_BGR2GRAY);
	threshold(huMom, huMom, 108, 255, THRESH_OTSU);

	Moments mom;
	double hu[7];

	mom = moments(huMom, true);
	HuMoments(mom, hu);
	for (int i = 0; i < 7; i++) {
		file << hu[i] << ','; // extract region hu moments
	}

	Canny(huMom, huMom, 30, 90, 3);

	mom = moments(huMom, true);
	HuMoments(mom, hu);
	for (int i = 0; i < 7; i++) {
		file << hu[i] << ','; // extract edge hu moments
	}

	file.close();

	Mat		hsv;

	cvtColor(win[2], hsv, COLOR_BGR2HSV);
	split(hsv, threeImages);
	cvtColor(threeImages[0], win[3], COLOR_GRAY2BGR);
	putText(legend[3], "Hue", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
	cvtColor(threeImages[1], win[4], COLOR_GRAY2BGR);
	putText(legend[4], "Saturation", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
	cvtColor(threeImages[2], win[5], COLOR_GRAY2BGR);
	putText(legend[5], "Intensity", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
}

int main(int argc, char** argv) {
	int i = 0;
	Mat		threeImages[3], hsv;
	ofstream file;
	file.open("data(All fruits).csv", ios::app);
	file <<
	"x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23
	,x24,x25,x26,x27,x28,x29,x30,y" << endl; // data column name

	namespace fs = std::experimental::filesystem;
	fs::path path("C:\\Fruits");
	for (auto& p : fs::directory_iterator(path))
```

```
    {
        for (auto& s : fs::directory_iterator(p))
        {
            const auto pathnameStr = s.path().string();


            char* c = const_cast<char*>(pathnameStr.c_str());
            segmentObject(threeImages, (char *)c);
            drawHSVHistogram(threeImages[0], threeImages[2], (char *) "H
    Histogram");
            drawHSVHistogram(threeImages[1], threeImages[2], (char *) "S
    Histogram");
            file << i << endl;
        }
        i++;
    }

    file.close();
    waitKey();
    return 0;
}
```

ModelTraining.py

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import joblib as jl
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import auc
from sklearn.metrics import roc_curve
from sklearn.metrics import precision_recall_curve
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import cross_validate
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB

# read data from csv
df = pd.read_csv(r'C:\Users\oka_w\Desktop\Project431\Project431\data(All fruits).csv')

# x as features, y as labels
x = df.iloc[:, 0:30]
y = df.iloc[:, 30:31]

# convert data to array
X = x[["x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "x14",
        "x15", "x16", "x17", "x18", "x19", "x20", "x21", "x22", "x23", "x24", "x25", "x26",
        "x27", "x28", "x29", "x30"]].to_numpy()
Y = y.values.ravel()

# split data into training set (80%) and testing set (20%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2)

# normalization of data
huminVal = X_train[:, [0,1,2,3,4,5,6]].min()
humaxVal = X_train[:, [0,1,2,3,4,5,6]].max()
X_train[:, [0,1,2,3,4,5,6]] = (X_train[:, [0,1,2,3,4,5,6]]-huminVal) / (humaxVal - huminVal)

hueminVal = X_train[:, [14,15,16,17,18,19,20,21]].min()
huemaxVal = X_train[:, [14,15,16,17,18,19,20,21]].max()
X_train[:, [14,15,16,17,18,19,20,21]] = (X_train[:, [14,15,16,17,18,19,20,21]]-hueminVal) /
        (huemaxVal - hueminVal)

satminVal = X_train[:, [22,23,24,25,26,27,28,29]].min()
satmaxVal = X_train[:, [22,23,24,25,26,27,28,29]].max()
```

```
X_train[:, [22,23,24,25,26,27,28,29]] = (X_train[:, [22,23,24,25,26,27,28,29]]-satminVal) /
          (satmaxVal - satminVal)

# save the values to normalize data
normVal =  [huminVal, humaxVal, hueminVal, huemaxVal, satminVal, satmaxVal]
jl.dump(normVal, 'NormalizeData.pkl')

# train Naive Bayes model
gnb = GaussianNB()
gnb.fit(X_train, Y_train)

# save the trained model
jl.dump(gnb, 'GaussianNB.pkl')

# output accuracy result of training set
pred = gnb.predict(X_train)
pred_train = accuracy_score(Y_train, pred)
print("Training Set Accuracy: {:.4f}".format(pred_train))
print('\n\n\n\n')

# 5-fold cross validation
kfold_result = cross_validate (gnb, X_train, Y_train, cv=5, scoring='accuracy',
          return_train_score=True)
print('Average training accuracy = {:.4f}'.format(kfold_result['train_score'].mean()))
print('Average validation accuracy = {:.4f}'.format(kfold_result['test_score'].mean()))

# confusion matrix
plot_confusion_matrix(gnb, X_train, Y_train, cmap=plt.cm.Blues)
plt.show()

Y_pred = cross_val_predict(gnb, X_train, Y_train, cv=5)

# precision, recall, and F1 score
print('Precision = {:.4f}'.format(precision_score(Y_train, Y_pred, average='macro')))
print('Recall = {:.4f}'.format(recall_score (Y_train, Y_pred, average='macro')))
print('F1 Score = {:.4f}'.format(f1_score(Y_train, Y_pred, average='macro')))
print('\n\n\n\n')

# y as Apple Crimson Snow(value = 0)
Y_1 = (Y == 0)
X_train1, X_test1, Y_1_train, Y_1_test = train_test_split(X, Y_1, test_size=0.2)

gnb1 = GaussianNB()
gnb1.fit(X_train1, Y_1_train)
```

```
# Display the precision-recall curve
y_scores_cv =cross_val_predict(gnb1, X_train1, Y_1_train, cv=5, method="predict")
precision, recalls, thresholds = precision_recall_curve(Y_1_train, y_scores_cv)

def plot_precision_vs_recall(precisions, recalls):
    plt.plot(recalls, precision, "b-", linewidth = 3)
    plt.plot(np.linspace(0, 1, 20), np.linspace(1, 0, 20), 'k--')
    plt.xlabel("Recall", fontsize = 16)
    plt.ylabel("Precision", fontsize = 16)
    plt.axis([0, 1, 0, 1])

plt.figure(figsize = (8, 6))
plot_precision_vs_recall(precision, recalls)
plt.title("Precision-Recall Graph (Training Set)", fontsize = 20)
plt.show()

# Display the roc curve
fpr, tpr, thresholds = roc_curve(Y_1_train, y_scores_cv)
roc_auc = auc(fpr, tpr)

def plot_roc_curve (fpr, tpr, style = 'b-', label = 'AUC = %0.4f' % roc_auc):
 plt.plot(fpr, tpr, style, linewidth = 3, label = label)
 plt.legend(loc = 'lower right')
 plt.plot([0,1], [0,1.2], 'k--')
 plt.axis([0, 1, 0, 1])
 plt.xlabel ('False Positive Rate')
 plt.ylabel ('True Positive Rate')
 plt.title('TPR vs FPR', fontsize=20)
 plt.show()

plot_roc_curve(fpr, tpr)

# normalize X_test
X_test[:, [0,1,2,3,4,5,6]] = (X_test[:, [0,1,2,3,4,5,6]]-huminVal) / (humaxVal - huminVal)
X_test[:, [14,15,16,17,18,19,20,21]] = (X_test[:, [14,15,16,17,18,19,20,21]]-hueminVal) /
          (huemaxVal - hueminVal)
X_test[:, [22,23,24,25,26,27,28,29]] = (X_test[:, [22,23,24,25,26,27,28,29]]-satminVal) /
          (satmaxVal - satminVal)

# output accuracy result of testing set
pred = gnb.predict(X_test)
pred_test = accuracy_score(Y_test, pred)
print("Testing Set Accuracy: {:.4f}".format(pred_test))
print('\n\n\n\n')

Y_pred_test = cross_val_predict(gnb, X_test, Y_test, cv=5)
```

```
# precision, recall, and f1 score of testing set
print('Test precision = {:.4f}'.format(precision_score(Y_test, Y_pred_test, average='macro')))
print('Test recall = {:.4f}'.format(recall_score (Y_test, Y_pred_test, average='macro')))
print('Test f1 score = {:.4f}'.format(f1_score(Y_test, Y_pred_test, average='macro')))
print('\n\n\n\n')
```

DemoCode.cpp

```
#include        <opencv2/opencv.hpp>
#include        <opencv2/highgui/highgui.hpp>
#include        <opencv2/imgproc.hpp>
#include        <iostream>
#include        <fstream>
#include        <experimental/filesystem>
#include        <memory>
#include        "Supp.h"

using namespace cv;
using namespace std;

void drawHSVHistogram(Mat HSV, Mat V, char s[], int enter) {
        // Below is the step to generate a histogram (frequency count) of the input image intensity
        // excluding the dark points
        /// Establish the number of bins
        int histSize = 8;
        ofstream file;
        file.open("image_input.csv", ios::app);

        /// Set the ranges for counting, i.e. intensity outside of this range is not counted
        float range[] = { 0, 256 };
        const float *histRange = { range };

        bool uniform = true; bool accumulate = false;
        Mat             histogram;
        int             emptyCount; // The count of dark point or empty area

                                                        // Compute the histograms: its input parameters are
                                                        // HSV: the input image or images
                                                        // 2nd input: number of images in srcI
                                                        // 3rd input: list of channels in each image for the
        computation, the first channel is 0
                                                        // 4th input: optional mask; if it is empty, no effect
                                                        // 5th: output matrix
                                                        // 6th: Histogram dimensionality, 1D histogram, 2D
        histogram, ...
                                                        // 7th: Array of histogram sizes in each dimension
                                                        // 8th: Array of the dims arrays of the histogram bin
        boundaries in each dimension
                                                        // 9th: Flag indicating whether the histogram is
        uniform or not, i.e. uniform have one
                                                        //      range in each dimension. Else it has multiple
        ranges with no count on out of range intensity.
```

```
                                                // 10th: Accumulation flag. If it is set, the histogram
        is not cleared in the beginning
        calcHist(&V, 1, 0, Mat(), histogram, 1, &histSize, &histRange, uniform, accumulate);
        emptyCount = histogram.at<float>(0);
        calcHist(&HSV, 1, 0, Mat(), histogram, 1, &histSize, &histRange, uniform, accumulate);
        histogram.at<float>(0) -= emptyCount;
        for (int i = 0; i < 8; i++) { // This part is important for you to access the histogram value
        at each bin
                if (i != 7) {
                        file << histogram.at<float>(i) << ',';
                }
                else {
                        if (enter % 2 != 0)
                                file << histogram.at<float>(i) << ',';
                        else
                                file << histogram.at<float>(i);
                }

        }

        file.close();
}

int segmentObject(Mat threeImages[3], char name[]) {
        Mat                     srcI = imread(name), tmp, cannyEdge, huMom;
        ofstream file;
        file.open("image_input.csv", ios::app);

        if (srcI.empty()) {
                cout << "cannot open " << name << endl;
                return -1;
        }

        int const       noOfImagePerCol = 2, noOfImagePerRow = 3; // create a 2X3 window
        partition
        Mat                     largeWin, win[noOfImagePerRow * noOfImagePerCol], // create
        the new window
                legend[noOfImagePerRow * noOfImagePerCol]; // and the means to each sub-
        window
        int                     winI = 0;

        int                     ratio = 3, kernelSize = 3; // set parameters for Canny
        Mat                     B = (Mat_<unsigned char>(3, 3) << 1, 1, 1,  // define Bel 1
                1, 1, 1,
                1, 1, 1);
```

```
createWindowPartition(srcI, largeWin, win, legend, noOfImagePerCol,
noOfImagePerRow);

copyMakeBorder(srcI, srcI, 4, 4, 4, 4, BORDER_CONSTANT, Scalar(255, 255, 255));
resize(srcI, srcI, Size(100, 100));

GaussianBlur(srcI, win[0], Size(3, 3), 0, 0); // copy the blurred input to the first
subwindow
putText(legend[0], "Gaussian Blur 3", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1); //
place text in the first legend window
Canny(srcI, cannyEdge, 60, 60 * ratio, kernelSize);
dilate(cannyEdge, cannyEdge, B); // Make edge thicker to fill gap/break
erode(cannyEdge, cannyEdge, B, Point(-1, -1)); // Make edge thinner to get back the
original edge as much as possible

vector<vector<Point> >          contours;
vector<Vec4i>                           hierarchy;

findContours(cannyEdge, contours, hierarchy, RETR_LIST, CHAIN_APPROX_NONE);
RNG                             rng(12345);
Scalar                  color;
int                             index, max = 0;

srcI.copyTo(tmp);
tmp = Scalar(0, 0, 0);

for (int i = 0; i < contours.size(); i++) {
        if (max < contours[i].size()) { // Find the longest contour as fruit boundary
                max = contours[i].size();
                index = i;
        }
}
drawContours(tmp, contours, index, Scalar(255, 255, 255)); // highlight fruit boundary
tmp.copyTo(win[1]);
putText(legend[1], "Fruit boundary", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);

vector<Point> vp = contours[index]; //access each contour point
Point2i                 p;
int                             count = 0;

p.x = p.y = 0;
for (int j = 0; j < vp.size(); j++) // Add all point coordinates
        p += vp[j];
p.x /= vp.size(); // take average, i.e. center of fruit
p.y /= vp.size();
floodFill(tmp, p, Scalar(255, 255, 255)); // fill inside fruit boundary
```

```
	erode(tmp, tmp, B, Point(-1, -1)); // attempt to eliminate empty space around boundary
	win[2] = win[0] & tmp;
	putText(legend[2], "Fruit tidied up", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);

	cvtColor(win[2], huMom, COLOR_BGR2GRAY);
	threshold(huMom, huMom, 108, 255, THRESH_OTSU);

	Moments mom;
	double hu[7];

	mom = moments(huMom, true);
	HuMoments(mom, hu);
	for (int i = 0; i < 7; i++) {
		file << hu[i] << ','; // extract region hu moments
	}

	Canny(huMom, huMom, 30, 90, 3);

	mom = moments(huMom, true);
	HuMoments(mom, hu);
	for (int i = 0; i < 7; i++) {
		file << hu[i] << ','; // extract edge hu moments
	}

	file.close();

	Mat		hsv;

	cvtColor(win[2], hsv, COLOR_BGR2HSV);
	split(hsv, threeImages);
	cvtColor(threeImages[0], win[3], COLOR_GRAY2BGR);
	putText(legend[3], "Hue", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
	cvtColor(threeImages[1], win[4], COLOR_GRAY2BGR);
	putText(legend[4], "Saturation", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
	cvtColor(threeImages[2], win[5], COLOR_GRAY2BGR);
	putText(legend[5], "Intensity", Point(5, 11), 1, 1, Scalar(250, 250, 250), 1);
}

int main(int argc, char** argv) {
	int i = 1;
	String inputImage;
	String pythonPath =
	"C:\\Users\\oka_w\\source\\repos\\NaiveBayesPython\\NaiveBayesPython\\DemoCode.p
	y"; // path to the python .py file
	Mat		threeImages[3], hsv;
	ofstream file;
	file.open("image_input.csv", ios::out);
	file <<
	"x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15,x16,x17,x18,x19,x20,x21,x22,x23
	,x24,x25,x26,x27,x28,x29,x30" << endl; // data column name

	cout << "Enter input image full path (Ex: C:/Users/user/Desktop/apple.jpg):";
	getline(cin, inputImage);
	cout << endl << endl;

	namespace fs = std::experimental::filesystem;
	fs::path path(inputImage);
	const auto pathnameStr = path.string();
	char* c = const_cast<char*>(pathnameStr.c_str());
	segmentObject(threeImages, (char *)c);
	drawHSVHistogram(threeImages[0], threeImages[2], (char *) "H Histogram", i);
	i++;
	drawHSVHistogram(threeImages[1], threeImages[2], (char *) "S Histogram", i);

	file.close();

	system(pythonPath.c_str()); // execute the python .py file

	waitKey();
	return 0;
}
```

DemoCode.py

```
import numpy as np
import pandas as pd
import joblib as jl
from sklearn import preprocessing
from sklearn.naive_bayes import GaussianNB

# load trained model
gnb = jl.load('GaussianNB.pkl') # trained model file path

# load values to normalized data
normVal = jl.load('NormalizeData.pkl') # values for normalization file path
huminVal = normVal[0]
humaxVal = normVal[1]
hueminVal = normVal[2]
huemaxVal = normVal[3]
satminVal = normVal[4]
satmaxVal = normVal[5]

# read user input data
df = pd.read_csv('image_input.csv') # input image features values file path

x = df.iloc[:, 0:30]
X = x[["x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10", "x11", "x12", "x13", "x14",
        "x15", "x16", "x17", "x18", "x19", "x20", "x21", "x22", "x23", "x24", "x25", "x26",
        "x27", "x28", "x29", "x30"]].to_numpy()

# normalize data
X[:, [0,1,2,3,4,5,6]] = (X[:, [0,1,2,3,4,5,6]]-huminVal) / (humaxVal - huminVal)
X[:, [14,15,16,17,18,19,20,21]] = (X[:, [14,15,16,17,18,19,20,21]]-hueminVal) / (huemaxVal -
        hueminVal)
X[:, [22,23,24,25,26,27,28,29]] = (X[:, [22,23,24,25,26,27,28,29]]-satminVal) / (satmaxVal -
        satminVal)

pred = gnb.predict(X)
if pred == 0:
   pred = "Apple Crimson Snow"
elif pred == 1:
   pred = "Apple Granny Smith"
elif pred == 2:
   pred = "Apple Pink Lady"
elif pred == 3:
   pred = "Avocado"
elif pred == 4:
   pred = "Grape Blue"
```

```
elif pred == 5:
    pred = "Grape Pink"
elif pred == 6:
    pred = "Grape White"
elif pred == 7:
    pred = "Lemon"
elif pred == 8:
    pred = "Lemon Meyer"
elif pred == 9:
    pred = "Lychee"
elif pred == 10:
    pred = "Mango"
elif pred == 11:
    pred = "Mango Red"
elif pred == 12:
    pred = "Orange"
elif pred == 13:
    pred = "Papaya"
print("This fruit is", pred)
```