Allison Campbell (851339938) and Chinmay Rajguru (853964913)

Fruit Detector Final Project Report

**INTRODUCTION**

Object recognition is used in a variety of applications such as for medical imaging, forensic sciences, and industrial automation. It is used to interpret pictorial information and to process scene data. Although widely used, object recognition software has long had significant pitfalls. It has high maintenance costs and is often not flexible or scalable. Thus, machine learning is now being experimented with to improve object recognition software.

One of the newest tools in this growing area is SciKit Learn, developed by Google. This project aims to prove this tool’s effectiveness through a simple supervised learning application.

The Fruit Detector is a tool developed with Python, using the SciKit Learn and Tkinter libraries. The tool “detects” which type of fruit is contained in an image provided by the user. Behind the scenes, the tools trains using examples of each fruit type provided by the developers. It is then able to accept a fruit .JPG file from the user’s computer or from a url provided. The result will be displayed on the simple GUI with the user provided image.

The Fruit Detector is a tool that provided direct application of supervised learning concepts taught in class and also provided an opportunity to learn a new tool set. It proved SciKit’s effectiveness on the small scale, and could quite easily be scaled up or extended for use in a different application.

**ALGORITHM DESCRIPTION**

Our algorithm is based on the RGB color space as feature vector which is provided by Scikit-Learn SVM classifier. Step by step explanation of Algorithm:  
1. Before launching the window(GUI), the application will fetch all of the images (data) from directories which are provided as the ‘Training sets’ for the further comparison.  
2. This will be on the basis of supervised learning concept of Machine Learning. This learning will include training a classifier by the pixel values of all images, and it will be stored as the processed directory for all training paths.

3. Once the learning is done, the GUI will be launched and the user will get two options to provide an input to the system. The system will receive an image from user by either:  
 I] Fetching the URL of an online image [.JPG format only]  
 II] Selecting an Image from computer’s directory

4. Once the image is received either way, the system will read that image

5. The system will search for the best classifier within the search space and return it

6. Once the system returns the output as a numerical value, that output will be used to select appropriate String value which is assigned to that number, and it will be displayed.  
7. Simultaneously, the image which the user provided as an input, will be converted into an image with dimensions of [ 256 x 256 ], and it will be displayed below the String output.

8. The user can repeat the process ‘n’ number of times with different inputs.

**USER’S MANUAL**

1. Run fruit\_detector.py
2. View the type of fruits that Fruit Detector supports that are displayed at the top of the GUI.
3. Either provide a URL to a jpg image or click “Browse” to select a jpg from your computer.
4. If a URL was entered, click “Submit” to process the request. If “Browse” was clicked and an image selected, you do not need to click any other buttons.
5. Wait momentarily while the system determines which fruit was provided.
6. View your results! The name of the fruit detected will appear above the image you provided.

**EXAMPLE PROBLEMS**

**.JPG Example:**

The user clicks “Browse” and selects the mango .JPG file. (Figure 1)

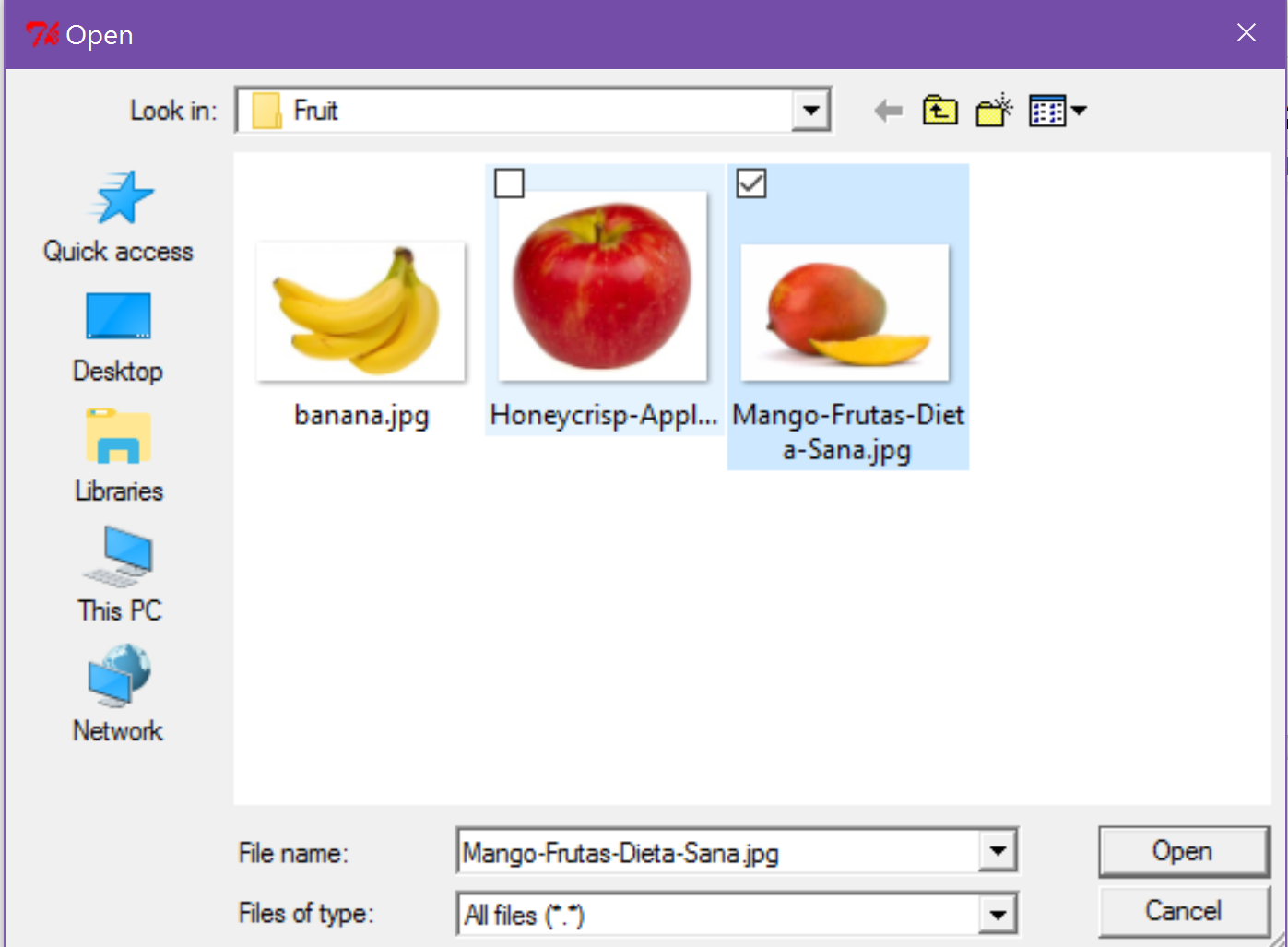


Figure 1

The Fruit Detector displays the correct output. (Figure 2)

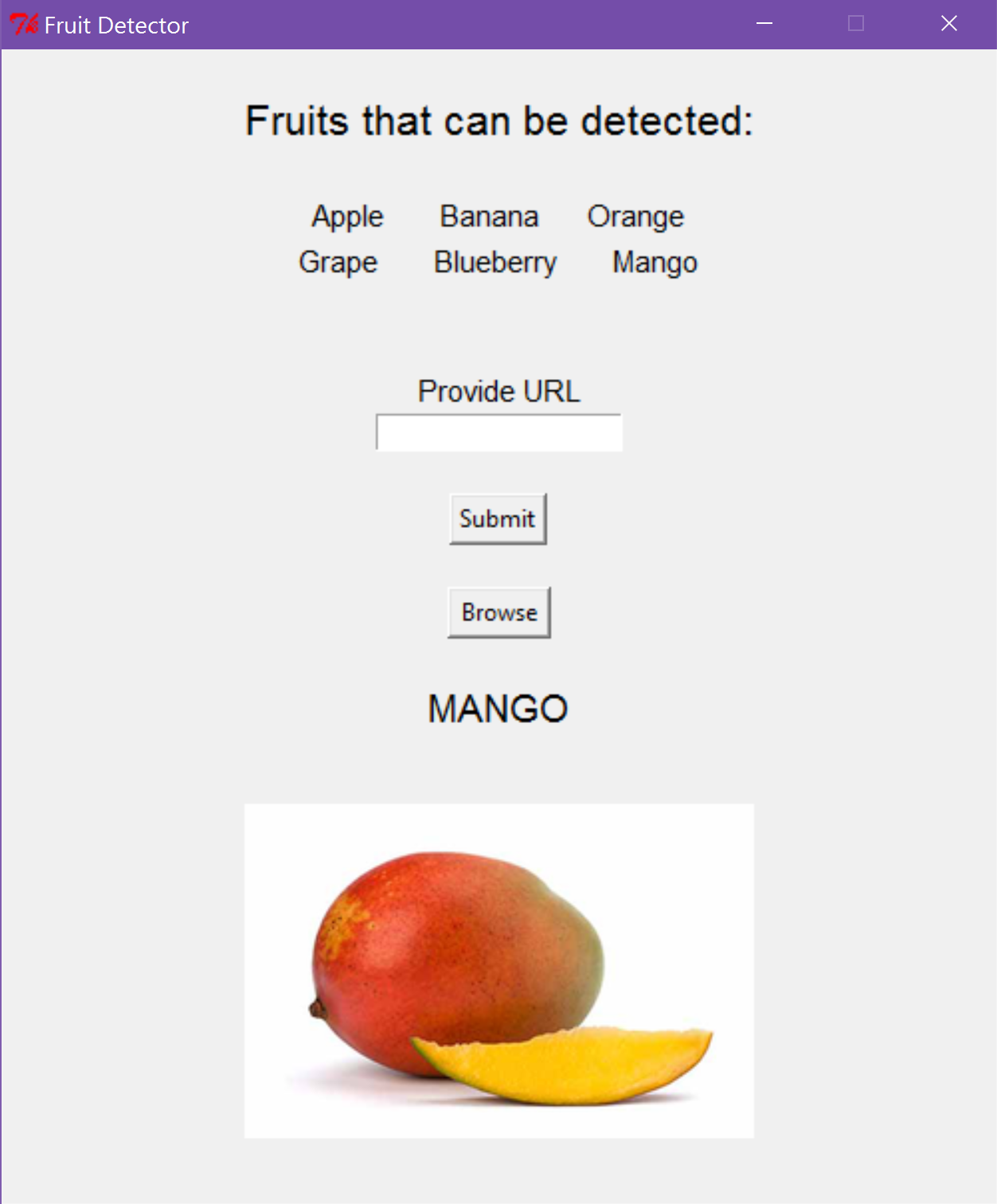


Figure 2

**URL Example:**

The user copies the URL of the image below and pastes it into the “Provide URL” user input. (Figure 3)



Figure 3

The Fruit Detector displays the correct output. (Figure 4)

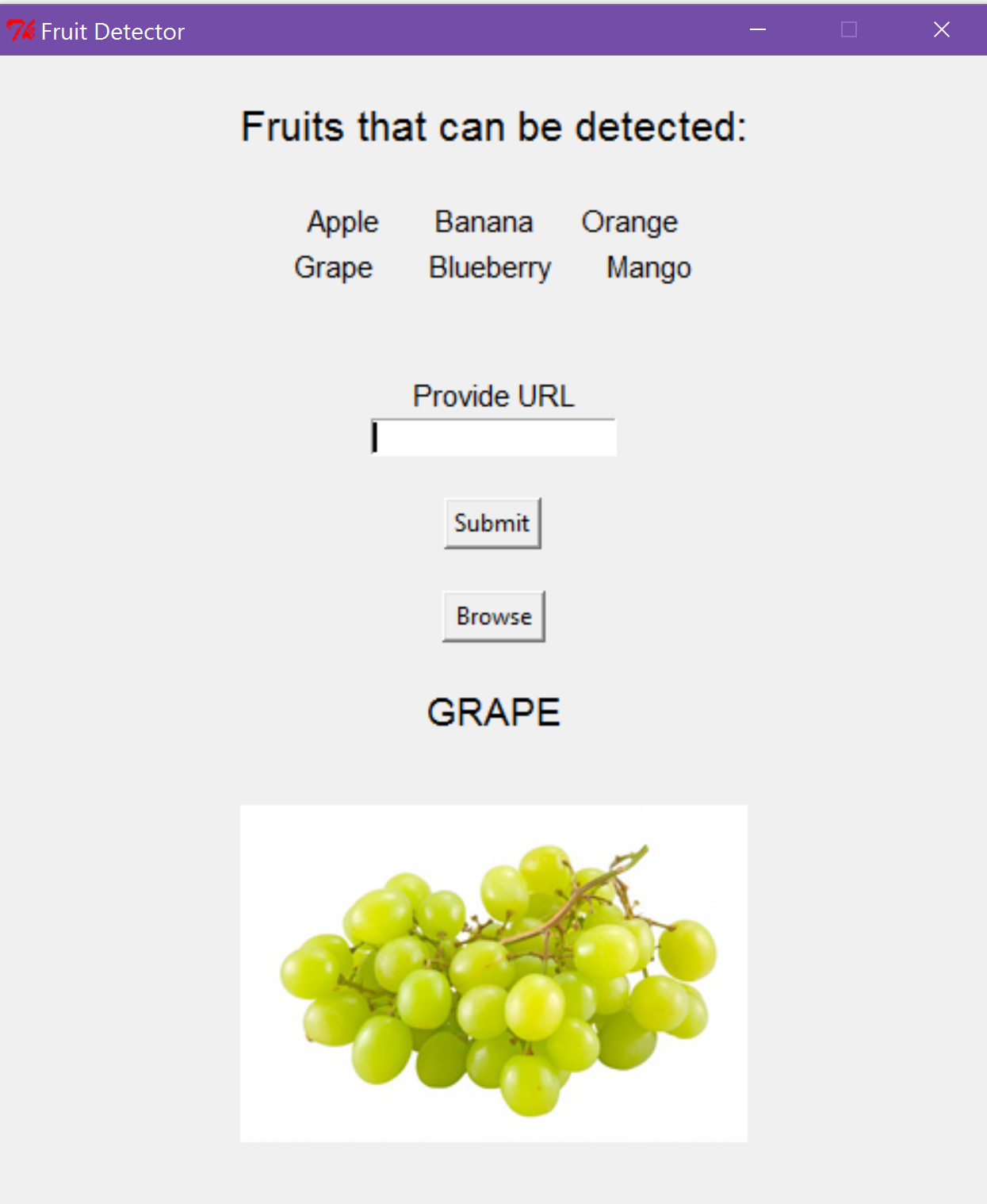


Figure 4

**COMPUTER CODE DESCRIPTIONS**

The Fruit Detector utilizes the SciKit Learn library to implement the Machine Learning functionality. The Support Vector Machine (SVM) component is a supervised learning model that produces a classifier.

The Fruit Detector is made up of a series of functions that control its functionality. They are described in detail below.

* **def process\_directory(directory)**: Accepts a full path for a directory that contains a training set for an individual fruit type. This directory should have roughly ten .JPG images of the fruit type. For each file in the directory, the function *process\_image\_file* is called. And, if this function returned any features, they are appended to an array and returned.
* **def process\_image\_file(image\_path):** Accepts a full path for an image file. It opens the file and calls the *process\_image* function. It returns the feature vector for the image that was returned by the *process\_image* function.
* **def process\_image(image, blocks=4):** Accepts an image object and does RGB pixel calculation on it. It returns a feature vector which contains floats that describe the RGB pixel values.
* **def process\_image\_url(image\_url):**Given an image URL it returns its feature vector. ‘ image\_url (str) ‘ takes the URL of image to process. It returns a float value of feature vector. It takes the image and will resize it with dimensions of [512 x 512] as well as it antialias the image and returns the processed image. It raises an IOError if the image URL is not valid or proper.
* **def train(training\_path\_, print\_metrics=False):**It trains a classifier. (e.g.:training\_path\_a and training\_path\_b should be directory paths) Each of the paths should not be a subdirectory of the other one. Training paths are processed by process\_directory().

Example: training\_path\_a (str): directory contains sample images of class A.

print\_metrics (boolean, optional): if True, print statistics about classifier performance.

Data : contains all the training data.

Target : It is the list of target classes for each feature vector.   
 example:'0' for class A and '1' for class B

Then the system defines the parameters and searches for the best match of classifier within the search space and returns the output.

* **def initialize()**: This is where we define the local paths for the directories that contain the training data. It creates a classifier by calling the *train* function and providing it the training directories. It returns nothing.
* Class FruitDetector
  + **def \_\_init\_\_(self, master):** Defines the initial state of the GUI using Tkinter. It defines its window parameters, layout, user inputs, buttons, and design. The Tkinter ‘pack’ functionality was used for the layout design. It defines that when the ‘Browse’ button is clicked, the *self.process\_file\_input* function will be called. And, when the ‘Submit’ button is clicked, the *self.process\_user\_input* function will be called.
  + **def process\_file\_input(self):** Uses TkFileDialog to accept a file from the user and then provides that file to the *process\_image\_file* function. After the features of the image are returned, it calls the *get\_fruit\_name* function with the input ‘classifier(predict(features)).’ This prediction is what the system has determined to be the output based on the supervised learning training data. Then, the name of the fruit is set to a Tk string and displayed. The input image is resized and displayed using ImageTk.
  + **def process\_user\_input(self):** Gets the user input URL from the Tk user input and passes it to the *process\_image\_url* function. After the features of the image are returned, it calls the *get\_fruit\_name* function with the input ‘classifier(predict(features)).’ This prediction is what the system has determined to be the output based on the supervised learning training data. Then, the name of the fruit is set to a Tk string and displayed. Urllib2 is used to parse the URL and display the resized .JPG using ImageTk.
  + **Def get\_fruit\_name(self, int\_val):** If the value of the integer which is received after processing is 0 then the fruit will be ‘Apple’. Like that all the integer values are compared and we assigned each fruit to every digit from 0 to 5. So, as system detects that number, output will be the String value equivalent for that number.

**CONCLUSION:**

To learn implementation of Supervised Learning from Machine Learning by using python’s GUI, we made an application which will learn from a provided dataset and will compare that data to the image provided by the user on run time. Looking at the pixel values of the input image, the system will compute the best match for those pixels and will show the output, by providing the image belongs to which dataset.

There are future possible modifications for this project such as to store learning data on cloud, make the system more mobile to use, user can train the data and add more options such as adding user’s own training data by uploading images into the directories. Since our system is just learning through the images, any comparative image sets can be used as a data for input to out system.