

Google Developer Groups on Campus

Asia Pacific University chapter

Creating a Machine Learning Drawing Classifier with Python

{Documentation}

| DATE | 28th February 2025 |
| --- | --- |
| DEPARTMENT | High Committee, GDGoC APU Lead |
| LED BY | Lai Tzi Syuen Suzanne |

WELCOME

Hi everyone! Suzanne here. This is the first workshop I’m teaching in 2025 as the new lead of GDGoC, so it’s a little nerve-wracking, but I’m assuming that you’re all present in my workshop physically as well. This documentation is meant to be used as guide, not step-by-step instructions, and I hope you’ll be able to put it to good use while trying to follow along with the workshop in real time as well.

If you ever get stuck on a certain step, do remember to look up the error indications as the limitations/settings on each computer is different and the in-person helpers might not have encountered the same bugs before. As of the date of the workshop, this guide has been tried and tested by several other members of the team and it is working!

That being said, good luck and enjoy :)

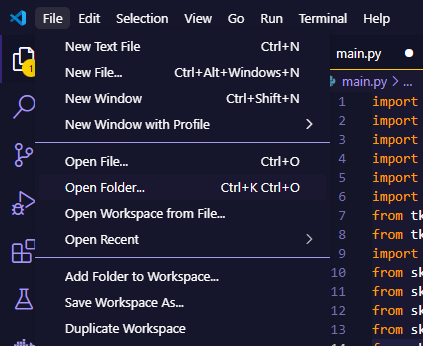


**Step 1: Set up your folder structure**

\*By the way, this guide assumes that you’ve had Visual Studio Code and Python installed already

I recommend you create a new folder in your **DESKTOP** so it’s easier to find. You can name it DrawingClassifier or something similar. As is with most project folders, it’s also recommended that you don’t use spaces or special characters in the name.

On your VSC, you can click on file -> and then click on Open Folder. Navigate to the folder you’ve created.

****

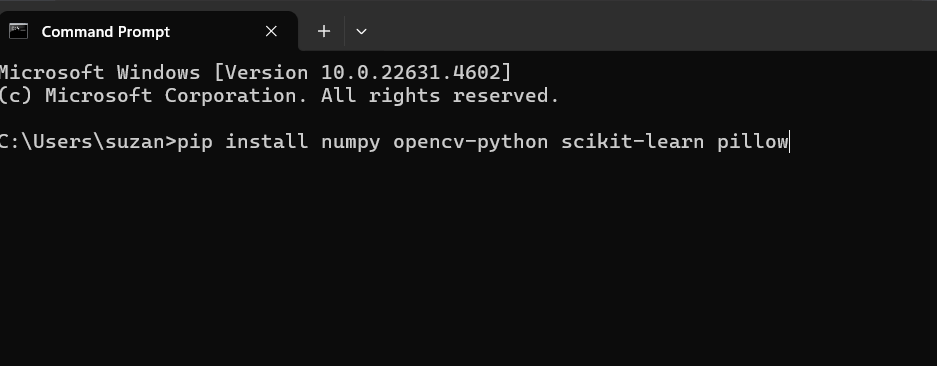
****

Once the folder is open, you’ll find it displayed as in the picture above. Click on the icon of the document with the folded ear.

Name this new file **main.py**

You have now successfully created a new Python file.

**Step 2: Installing and Importing**

****

Open up your command prompt and type in:

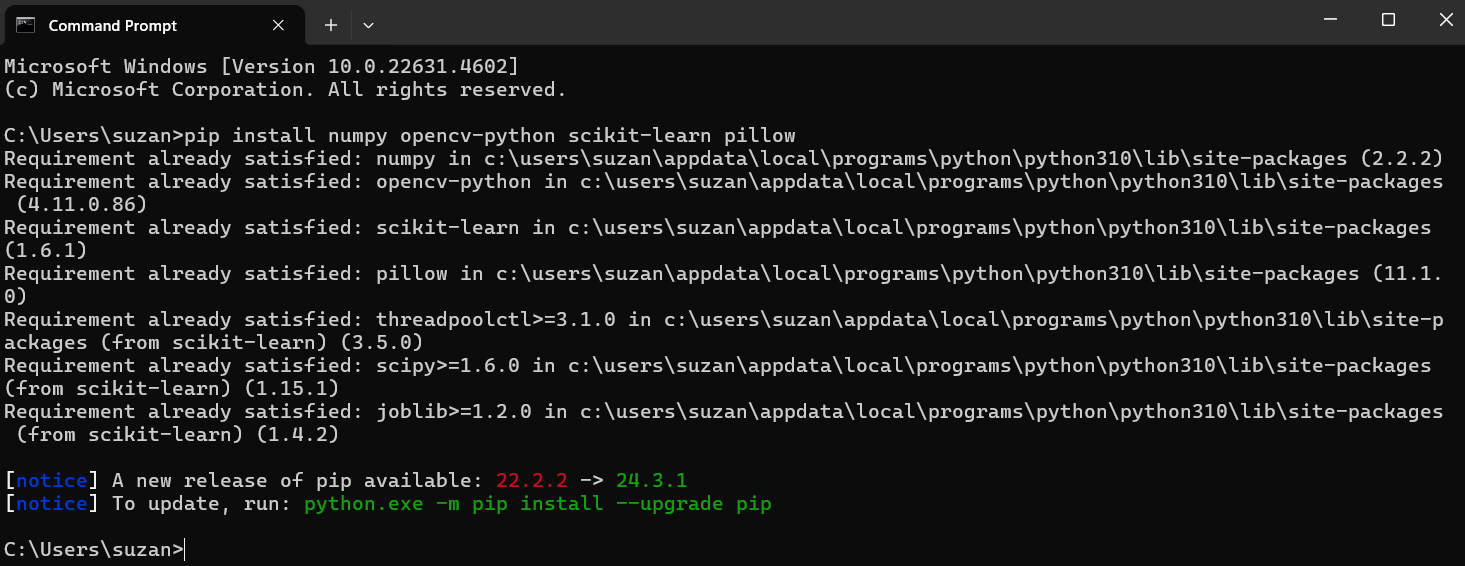
pip install numpy opencv-python scikit-learn pillow

Numpy: for manipulation of arrays and matrices, e.t.c.

Opencv-python: for manipulation and purposes regarding computer vision

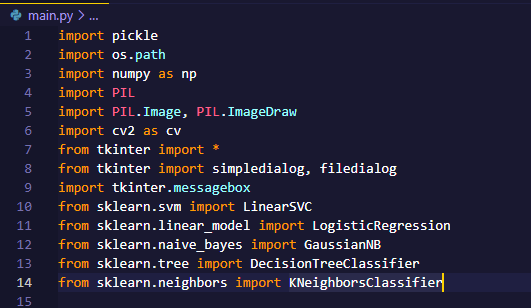
Scikit-learn: machine learning library

Pillow: library for the manipulation of images in multiple formats

****

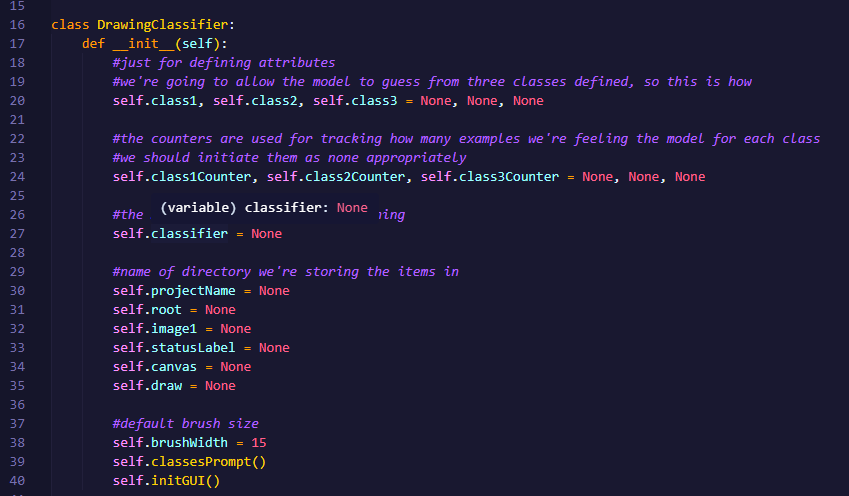
Once you press enter in your command prompt, it will check the libraries listed against the ones you’ve already installed into your computer and process the rest.

In my case, it says that all the requirements have been satisfied already.

****

Then, in **main.py**, import the libraries needed as shown above.

**Step 3: Define your class**

****

Start defining the class, DrawingClassifier, as shown above.

This class is going to be a “blueprint” of sorts for each of the drawing classifier project instances that we create. Because the drawing classifier is going to predict the drawing from a selection of three different classes, we will initiate 3 classes as class1, class2, and class3, but they should all be set to None at first since it will only be entered upon the creation of a new instance.

The counter is going to be used to count the amount of dataset/pictures that you enter for each of the classes. Initiate oen for each class and set them to None.

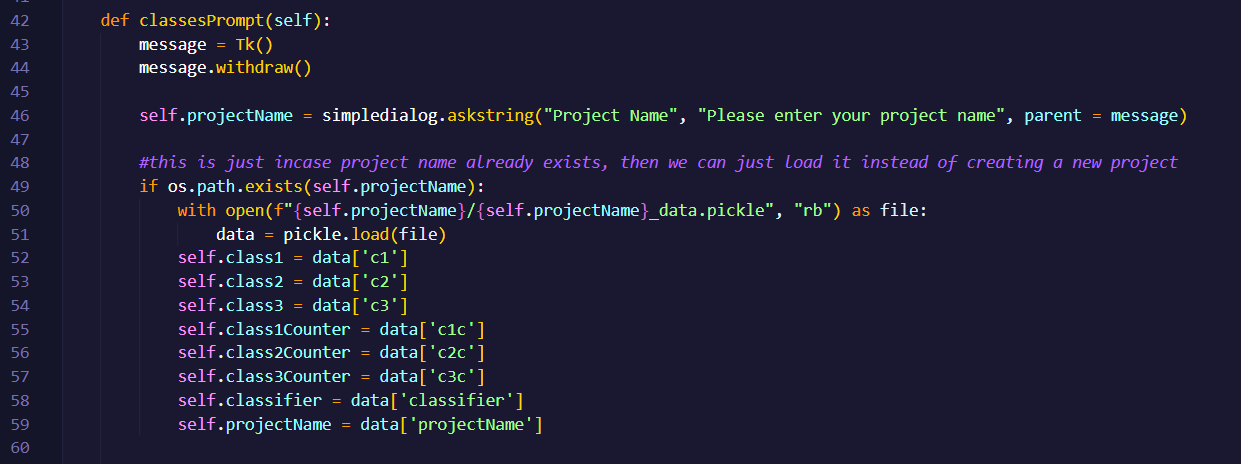
This class should also have a classifier which is used to train the model, but it can be set to None as well.

Next are the set of attributes required for the user to be able to draw on the canvas for both the purposes of training + prediction.

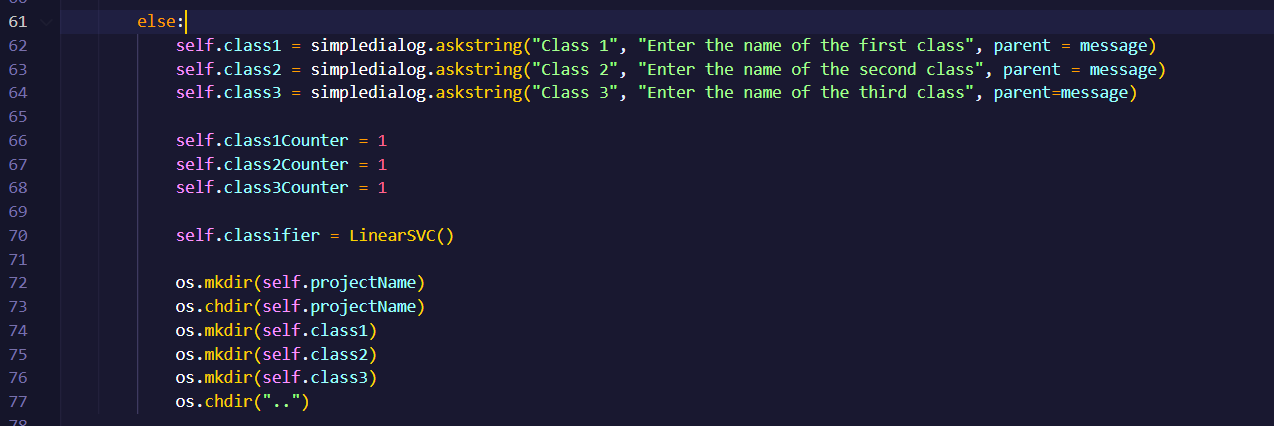
**Step 4: Create your initiation method for this class**

You need to have method within this class that allows the GUI library (TKinter) to prompt the users to enter the names of the three classes that they will train their model on.

Follow this code as shown below to create such a window for user input.

****

The section of the code as shown above shows that if the user creates a new project that already exists, it will use the previously saved class names as the class names in this instance.

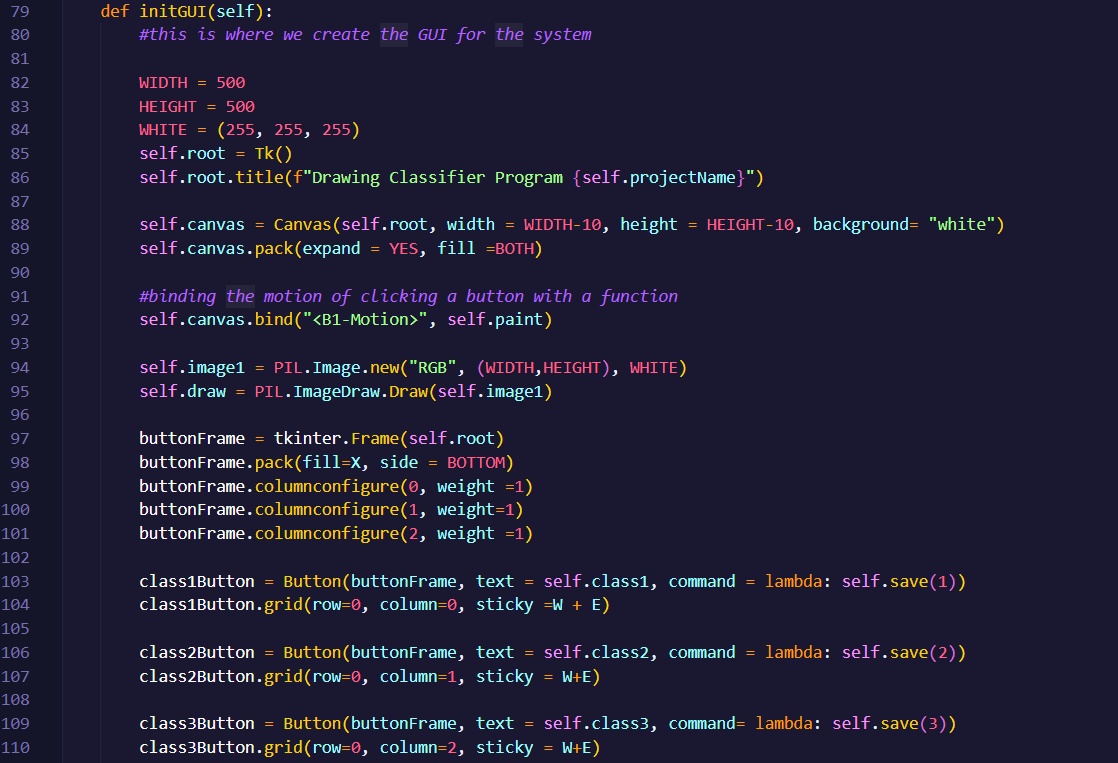


If the code checks for an existing project folder for the same name and turns up with nothing, it would mean that the project is considered new. Therefore, write the else code snippet above which will execute to prompt the user to enter the names of the first, second and third classes. The counters are initiated and the default classifier algorithm is chosen to be LinearSVC (you can change the classifier to whatever you wish to be default).

The last snippet of code from line 72 to line 77 are for the creation of the project directory. Then, you enter the newly created directory to create respective folders for class 1, 2, and 3 before changing it back to the main folder.

**Step 5: Create the initGUI method**

The initGUI method is used to create the GUI interface on which the users will be able to interact with the system and its various functionalities

****

First of all, the width and height of the window are initiated. TK is initiated so that we can create the GUI components for the system.

The title of the window is set to “Drawing Classifier Program” for whatever your project name is.

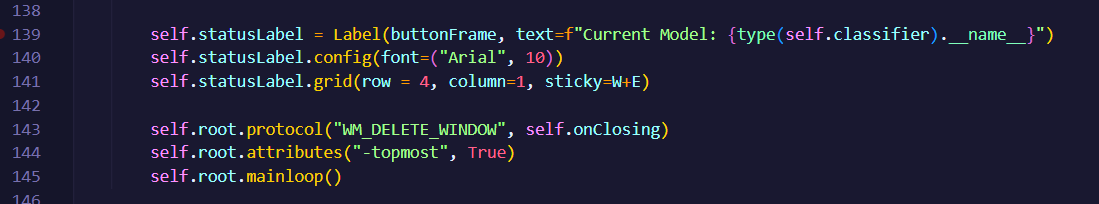
The canvas is then initiated with a white background. On the canvas, we are binding the mouse button 1 (left click button), to the painting method (which we will define soon).

At the bottom of the window, the buttonFrame is created with three columns so we can start creating our buttons to link to each functionality we will have.

The first three buttons are for saving the drawings as data for each classes, so we bind it to the save method which we will define later.



The rest of the buttons are created to bind them to methods such as brushminus, brushplus, clear, trainModel, saveModel, loadModel, changeModel, predictClass, and saveEverything.



The rest of this method would be to create the labels to complete the canvas on which the drawing will be completed.

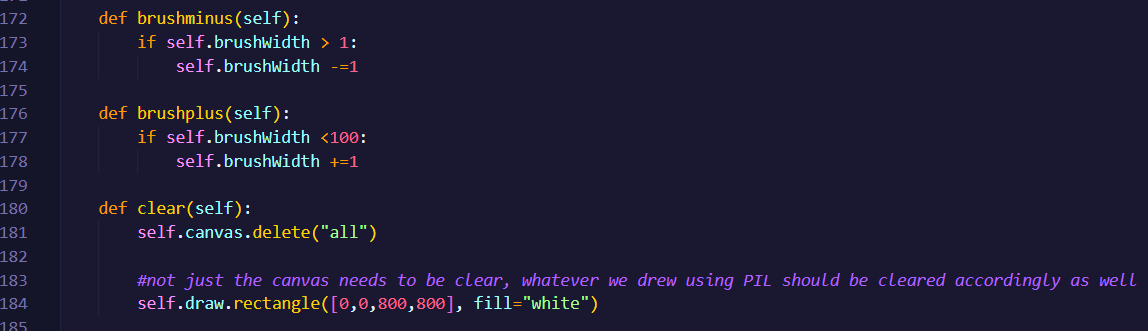
**Step 6: Defining the methods**

****

The code snippet above shows the implementations for paint and save. For paint, it takes in two parameters, self, and event.

The event parameter is taken in to record the brush strokes on the canvas. Lines 148 to 151 are written to define the rectangular area on which the drawing will be created). The rectangle/lines are draw with the brushWidth defined in this instance of the class.

The save method is defined to save the drawings on the canvas to their respective class folders. It takes in two parameters, one of which is the classNumber so that it knows which directory to save the picture of the drawing into. After saving the picture, the canvas will be cleared.



Next, the brushminus and brushplus methods are defined. The brushminus method ensures that so long as the brushWidth is wider than 1, the brushWidth can be reduced. The brushplus method is used to ensure that as long as the brushWidth is less than a 100, it can still be increased in size.

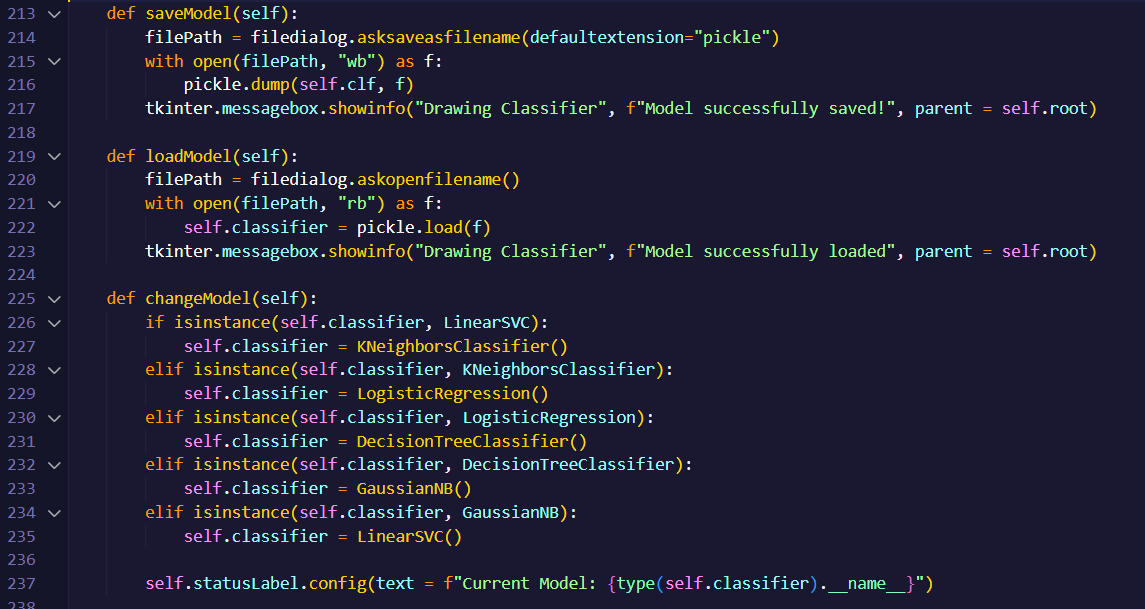
The clear method is defined to delete everything off of the canvas. But you have to make sure that whatever drawn using PIL is also cleared accordingly as well.



The trainModel method is defined to train the model using the set classifier as well as the set of photos that are available in the list of images for each class.

Firstly, two lists are declared for imageList and classList. For the range of photos available in each of the class folders, they are reshaped and appended into the list for training images.

The classifier is then fit with the images as well as the class names, and once training has been concluded successfully, a message prompting as much will pop up for the user to see.



The saveModel method is defined to allow the user to save the model that they have successfully trained into a pickle file. As previously done, if it is done successfully, a message indicating as such will be displayed.

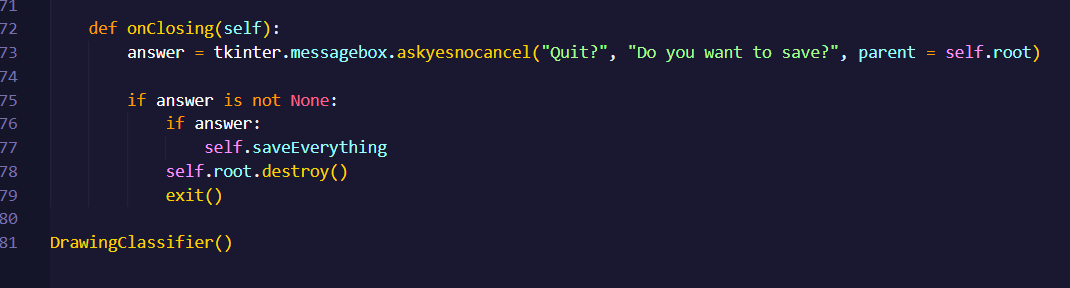
The loadModel method is defined to do the opposite, which is to load a previously saved model on the local computer into the current instance of the drawing classifider.

The changeModel method allows the user to change the classifiers used for the model using a self-created loop. That is, it starts from LinearSVC and moves onto KNN, and so on and so forth until it reaches LinearSVC again. When the classifier used is changed, the label will change to reflect the correct name as well.



The method predictClass is used to save the drawing created on the canvas as temp.png before it is passed into the classifier to be predicted as one of the classes. Depending on the result stored into the variable prediction, the system will display one of three messages which tells the user which class it is.

The saveEverything method is used to save the whole model, including the pictures, classes, counters, classifiers and project name used in the current instance of the class into a pickle file. The success message is also defined.



Finally, onClosing will prompt the user to confirm if they would like to save the model. If the answer is positive/True, the saveEverything method will be invoked. If not, the instance will be destroyed and the user exists the system.

**Step 7: Start the system**

****

If you’ve managed to successfully follow the tutorial, running the whole file should open the drawing classifier on your computer! If it’s your first time working with the project/opening it, it should take some time but don’t worry!

Thanks for following along the tutorial, and I hope you learned something from it <3