The Problem: Picture recognition. Given a .jpg file, write a program that can output what the picture is. There are five possible pictures including a: smiley face, hat, hash mark, dollar sign, or heart.

The approach: Create a support vector machine to then teach the machine what each object is by loading in numerous pictures of each type of object. For example, load a large amount of smiley face .jpg files into the support vector machine and tell it that each .jpg is a 'smile' even though each smiley face may vary slightly. With this logic, the machine theoritcally should be able to predict that an object is a smile based off the smile data that was previously loaded into the machine. Repeat this loading process for each type of object.

The accuracy: Unfortunately, my implementation of this approach is not very accurate. If implemented correctly however, this approach would deem very accurate given a large enough data set loaded into the machine.

The reason for the inconsistent results my machine predicts is due to a failure of loading in the images properly. From google, I read the best implementation of a support vector machine in Python is by using the one from the sklearn library. The sklearn library creates a 'classifer' with the following code:

clf = svm.SVC(gamma=0.0001, C=100)

Where the machine has a gamma value of 0.0001 -¿ a small gamma means a small increment step which again, theoritically produces accurate results.

After creating the classifier I created an array of the pixel data of each picture, converted the array to a list, then appended it to a list of lists. Proceed to append that list of lists to another list which would be the list of 5 elements representing each kind of picture object. Repeat this process for each object. Finally, 'fit' all the data into the classifier with a target ID that correlates to the kind of object it is. I used the fit function included in the sklearn library with the following line:

clf.fit(arrayOfData, arrayofTargets)

This line is where my one and only error came to never allow a proper fit of the data. Whenever I tried to fit a list that had more than one kind of each picture, this fit function returned a ValueError, stating that I was 'setting an array element with a sequence'. The only time it did not return this error is when I reset the list of pixel data to empty in the same loop I appended the pixel data. Therefore, only one picture (the last one loaded in the loop) stayed in the array of data for each kind of object. I tried passing the pixel data to another function within the loop to maybe get around this problem but the ValueError appeared every attempt at solving this problem. After days of trying to figure out the same error I called it quits. I am unsure how the fit function wasn't working properly when it seemed the parameters

I passed it were the exact data types used in the examples and tutorials online.

Regardless, after the fit function, I called the predict function on the classifer with:

clf.predict(unknownPicture)

To predict a passed in picture from the command line. This only predicts the correct object some of the time since the classifer was only able to fit one of each picture object.

In conclusion, I fully understand the concepts meant to be learned here after the days of googling but still ended up with an imperfect program.