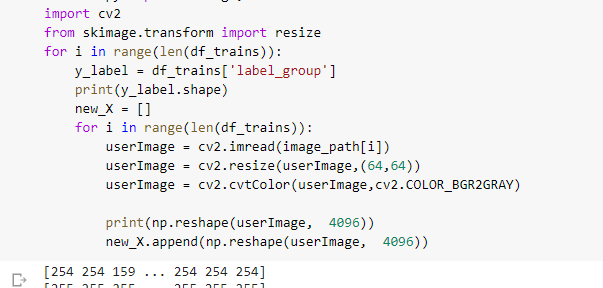
Project Description:

This project consists of images that could be sold in an online marketplace and includes duplicate images, images of the same item that are taken at different angles, and images where the item is in a different place from the original picture. For our project, we wanted to use algorithms that allowed us to see how accurate they are when looking for similar projects without using the CSV file to match names or product IDs. Although this project contains 34250 lines of products in the CSV and 32412 images in the training folder, due to memory and time limitations, we tested the algorithms on datasets consisting of 500 images. The purpose of the project is to use machine learning algorithms on the 500 images to find the most accurate algorithm with the ability to identify an image. Google collab was used in order to process the high amounts of data for the project. The images consisted of the same image of each object but at different angles and/or different places. Once the algorithm was trained, it was able to identify similar images as the same object. The algorithms used were decision tree, random forest, ada boost, xgboost and MLP classifier. For the use of Google Colab, we used ~100 image samples because of the RAM limitations with Colab.

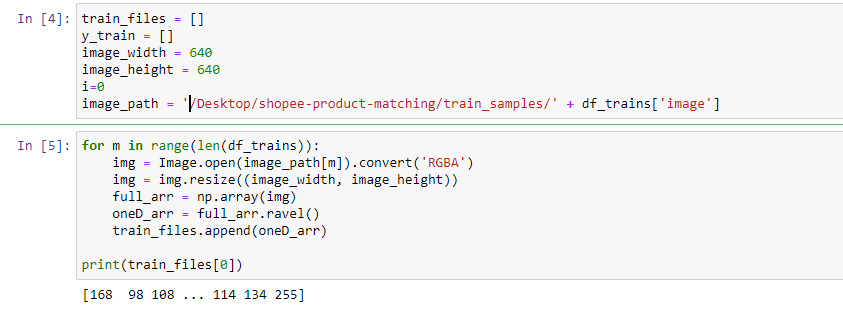
Developed Methods:

Developed methods: The method we first developed was a method that tried to convert the images into an array, so we could normalize, fit, and train the algorithm. The struggle with this, was the different sizes of the images, the most common size of image was (640,640,3), so we tried to use a function to resize the thumbnail and ultimately the image to 160\*160\*3, where we resized all pictures to 640\*640\*3 then used a ratio of 4 to lower the size of the image. We then tried to convert that image to a 3D array. We ran into problems fitting this array into the algorithms we were comfortable using, so we had to scrap this and restart.

The next trial, we used our algorithm from Assignment 3 to resize and reshape the images into a 1D array. It used CV2 to read and resize the image to 64 \* 64, and reshaped it to the size 4069. While this did work and give us smaller sized data, the data was a lot more inaccurate as our next method of converting the image to an array.

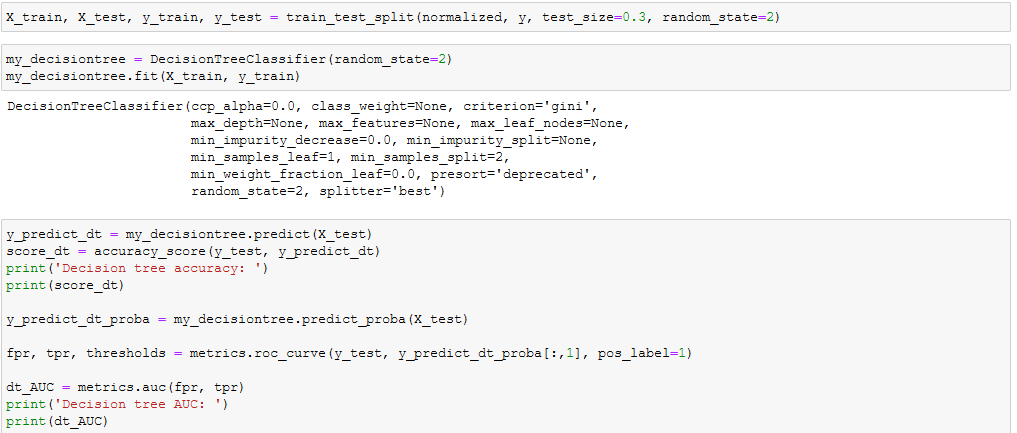


The final method we developed was using PIL to convert out image to an array by resizing it to 640\*640 without resizing it to a lower size to see if a large image contained more data which would allow our results to be better. It flattened the 3D array we got and turned the data into a 1D array and append it to our list or images. And although they did turn out to be marginally better, the time required to run each algorithm was around 20 minutes for the Random Forest Classifier, which was the fastest, and 2-4 hours, which was the XGBoost Classifier. 2-4 hours was the average run time for each algorithm we had run, using the dataset of 500 images.

  
Data:

Each image was turned into a 1D array based on the RGB value of each pixel. Each pixel was considered as a feature and the labels was the name of each image. This data was then fed into the algorithm to train and test it so it can identify the same object on different images regardless of its orientation.

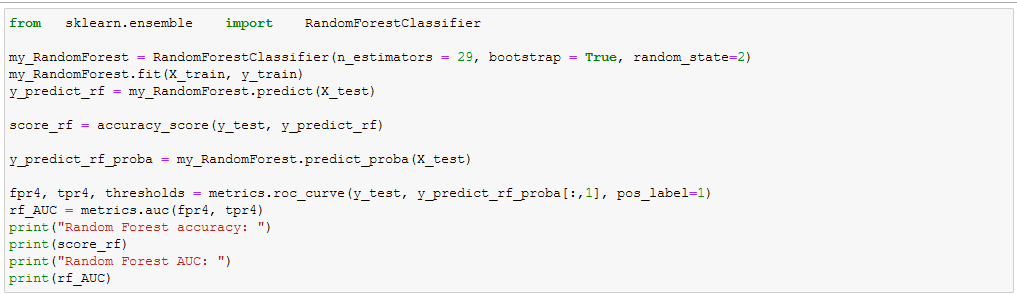
* Decision tree:



Decision tree accuracy: 0.06666666666666667

Decision tree AUC: nan

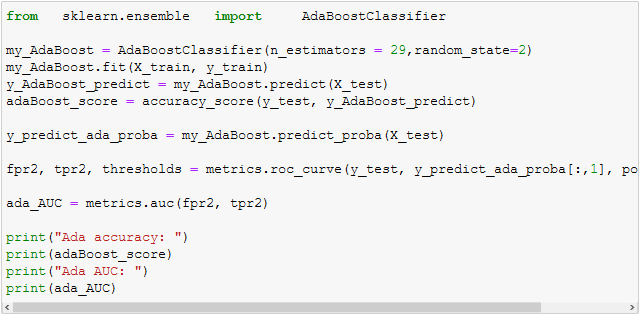
* Random Forest:



Random Forest accuracy: 0.1

Random Forest AUC: nan

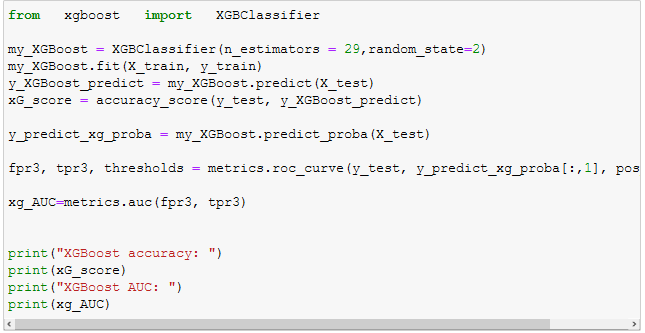
* Ada Boost:



Ada accuracy: 0.0

Ada AUC: nan

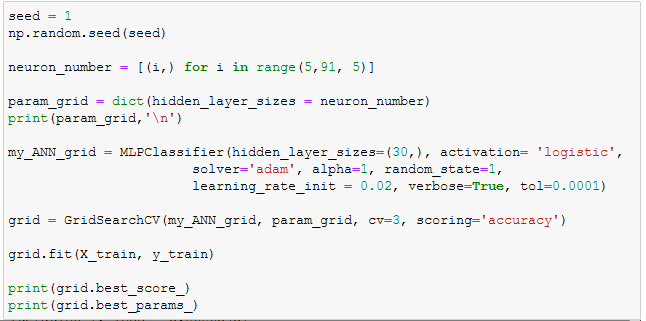
* Xgboost



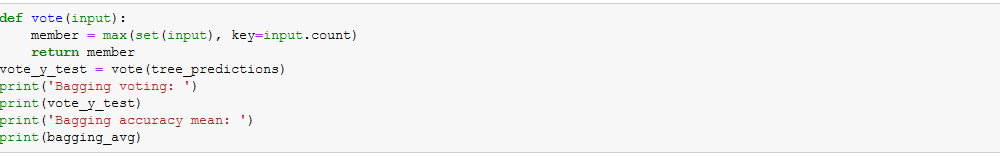
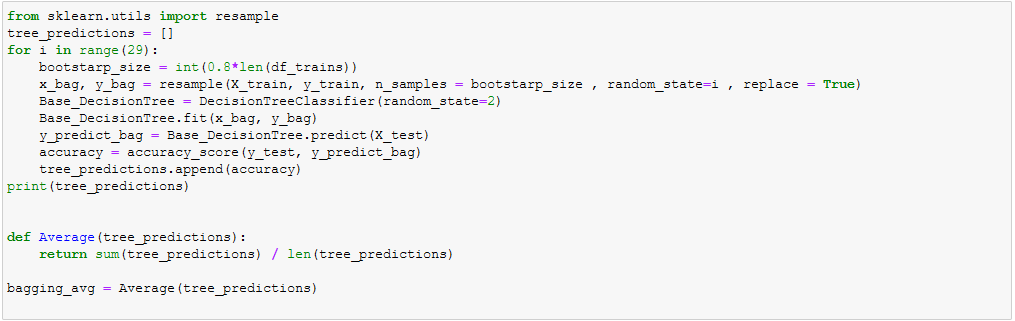
XGBoost accuracy: 0.1

XGBoost AUC: nan

* MLP and Grid Search



* Decisions Tree bagging and voting



Bagging voting: 0.06666666666666667

Bagging accuracy mean: 0.047126436781609195

Codes to address project requirements:

The hardest part was getting the images to fit into an 1D array, and allow us to get good results and an acceptable runtime. Another problem we had encountered was that the CSV for the dataset and number of images provided was mismatched. The images were also not the same dimensions. We had to find a good solution that provided a good runtime for resizing, reshaping, and appending to our image features array. We could not get ROC curves because all the methods used resulted in NaN AUC values.

Responsibilities:

Joshua Perez:

Find an algorithm to convert images to an 1D array.(shared responsibility), research on PIL and CV2 library to create 1D arrays that would be accepted to the algorithms we used this semester, research on algorithms that allow us to find identical pictures and accuracies.

Kuong Thong:

Find an algorithm to convert images to an 1D array.(shared responsibility), research on TensorFlow to find ways to convert images to arrays (dropped because of the memory and hardware problems that occurred), trained and tested algorithms used for the 1D array whenever we came up with a new method to make a 1D array.