Emma Peatfield 009192534

Rank: 39 F1: 0.8306

Program 2 – Image Classification

The goal of this assignment was to intake a dataset of images and their features. The images were street images that included pictures of cars, trucks, people, cyclists, etc. The goal was to take an image and specify what type of object is in each photo based on the information that we could gather from the training set. The original dataset was supposed to include 887 features for each image, however, the new dataset only included 47 features for each photo, which was not so great when you're required to include dimensionality reduction. Either way, it was my first time working with image features for classification, so I was excited to see what I could do.

This dataset was interesting to work with because it was a lot smaller than intended, so I had to consider the idea that dimensionality reduction may not help my precision and recall scores at all. I started by choosing a few different types of dimensionality/feature reduction and a few different classification methods that we had learned in class. I tested quite a few using only my training set that was split into a train and test set to start with, and then proceeded to use the actual test data once the CLP portal was opened up. Most of the methods I tried worked pretty well, but some gave pretty poor results. Here is a little bit more detail of what I did for my dimensionality reduction and classification methods.

For dimensionality reduction, I started with Random Forest and Principal Component Analysis methods. When training, Random Forest seemed to give a higher accuracy score than Principal Component Analysis, so I decided to start focusing on it. I used the sklearn package for both Random Forest and Principal Component Analysis to work with my dimensionality reduction. The idea behind Random Forest is to create multiple decision trees from the training set by splitting it up, and from those trees, use them to analyze and pick out the most prominent features. The idea behind Principal Component transforms the original set of data into a new set using principal components based on the variance between the attributes.

Moving on from dimensionality reduction, I chose to use my k-Nearest Neighbor code from the previous assignment for classification. In this code, I first find the similarity between all of the images within the test and training sets by using the cosine similarity process.

$$sim(a, b) = \frac{a \bullet b}{||a|| * ||b||}$$

From there, I used the similarities to find the k-Nearest Neighbors for each image in the test set. Once found, I took the mode of the classes of the nearest neighbors to predict what the class of each test image could possibly be.

So far, the highest f1-measure for my code has been obtained by using Random Forest for dimensionality reduction and k-Nearest Neighbor for classification, with k=5. I did try using Principal Component Analysis with my k-Nearest Neighbor classification, but I got a much lower f1-measure, so I moved onto trying other things. Another thing I tried was using Random Forest for dimensionality reduction and classification, as well as using Principal Component Analysis for both reduction and classification methods. However, neither of those gave very high f1-measures, so I went back to the drawing board.

I looked into using a few different methods that we had used for in-class activities, such as the Gaussian Naïve Bayes Classifier and the Multinomial Naïve Bayes Classifiers. Those two were easy enough to implement, but they didn't give the best f1-measures. I suspect it is because Naïve Bayes is not an ideal implementation for classifying these objects due to its Naïve nature, and the fact that the data set was not very large. Naïve Bayes Classification performs best with large datasets where probability values are less likely to be zero.

Another method I used was using the Random Forest and k-Nearest Neighbor packages from sklearn. Both of these together did not improve my score when compared to my own k-Nearest Neighbor code, however. I also tried using the Nearest Centroid package from sklearn, but that did not seem to work too well either. As time for turning in this assignment was running out, I had less and less time to look into other options, so I tried multiple different mixtures of all of the packages above, but nothing seemed to raise my f1-measure. All of my results gave either an f1-measure of 0.8306, or something lower. A few were much, much lower.

For future projects involving image classification, I would like to learn how to actually take the image and obtain the features that were intended to be in the dataset. Image classification is growing and for certain industries that want to automate things, such as maintenance inspections, it needs to be as accurate and precise as possible. I would like to know all of the different types of features that you can extract from photos and use for analysis, and the best methods to classify objects.

A suggestion for this assignment would be to have more features for each image and to give us knowledge about the importance of these features and what they have to do with classifying an image. I know that the dataset was supposed to be larger, but there was a mishap, I would just like to redo this assignment with images that have many more valid features. I think it is so interesting that we can take these images and specify what types of objects are in the photos, such as people, cars, etc. Next time, it would be very nice to have a legitimate data set that gives us much more information for each image. Thus, I'll be able to truly know how well my code was working.

One thing to note about my code is that it might be hard to read. My code kind of jumps all over the place because I kept everything that I had used up to a certain point on the same document, so there are a lot of sections that are just completely commented out. I did this because I wanted to be able to mix and match a few different methods and see what things I could do to improve my scores.