CS523 Competition Write up - Leland Ling

The task given to us for the Bitgrit competition was a classification problem, where we would have to discern AI generated and real images. Images were presented to us as flattened preprocessed rows within a CSV. Each submission was scored using F1 scores. I achieved an accuracy of 97.4 taking leaderboard position 30.

Data Exploration:

The first step was exploring the data. I found that each image, originally being a 3 channel 20x20 image, was flattened to a 1200 vector. I wasn't sure exactly how the three channel data was flattened (i.e. were the color pixels sequentially flattened or scrambled?) and as such color data was not explored. Another aspect to the dataset was that it was an unbalanced dataset with a majority of the entries being classified as 1s. When uploading a submission file with only 0s, an F1 score of 0 was returned. With a submission file with only 1s, an F1 score of .4 was returned. The data was then plugged into a logistic regression model and trained. I then plotted the feature coefficients against the feature index to see a feature importance score. I created three separate datasets using this feature importance, only taking features with coefficients that had absolute values above 0.1 and 0.2. This was done in effort to denoise the data, but results were inconclusive. Another augmented dataset was created by flipping the individual image vectors horizontally and adding it to the original dataset. The intuition behind this was that flipping images to augment datasets generally increases training accuracy and adds more examples to the data, and flipping the vector would effectively invert the image.

Basic Model Exploration:

Basic and traditional machine learning methods were explored first to see how base models fare in classifying these images. I tried KNN, Random forest, XGBoost, Naive Bayes, Logistic regression, Neural Networks, and Decision Trees to classify the data. The only three that had accuracies above 0.8 were KNN, XGBoost, and Naive Bayes. KNN immediately achieved an F1 score of .9.

These models were then grid searched for optimal parameters, eventually reaching an accuracy of 95.4 with KNN. This KNN model had parameters {'metric': 'euclidean', 'n_neighbors': 7, 'p': 1, 'weights': 'distance'}. KNN was also used to evaluate each of the augmented datasets; 0.1 and 0.2 datasets found higher accuracy within the validation set, but lower accuracy than the original dataset when submitted. Furthermore, using the flipped images as extra training samples increased the accuracy of the KNN model further to 97.4. XGBoost was not able to be grid searched due to time constraints; single model training would take 15-20 minutes to complete, with the grid search taking estimated weeks to complete.

I also explored stacking these models, but results were inconclusive as stacked models were unable to achieve a score above 95.4.