Prof Tom Lam **8610 Final Exam** **Scott Denning**

Dependencies:
pip install ordered-set

Setup
Copy the files from each of the folders on Github into the correspond folders in the provided exam_data folder. The paths used by the code expect the original provided folder hierarchy and will not be able to find the input files if the directory structure changes.

Part 1
I wrote a Naive Bayes classifier that does a pretty good job with the limited data available. The primary limitation is that there are not enough words in the training data for robust statistics for predicting the stars. Even so, it provides 99.7% accuracy in training data and appears to do 93% in test. As can be seen in the code, stop words and non-predictive words were stripped out of the bags of words created for each star level early in the processing

I created an sklearn logit classifier to compare the Naive Bayes classifier with. To do this I created a data frame with a column for each word in the allWords collection created from the reviews. Each training review was then loaded into a row of the data frame as a series of 1's and a 0's for each word in the review. I used 3 different logit models, one for each star level. The idea being to compare the probability from each model choosing the highest model probability as the predictor of the number of stars.

I was skeptical that the logit routine would be able to handle the more than 2500 variables but it does so with ease; the performance is excellent! I found this method to be fast and 100% accurate. This model runs slower than the Naive Bayes, but its accuracy makes it a winner.

Following are the command lines for running the fit and predict routines for part 1:

python3 part1Bayes.py -i training_data/dataset.csv -m fit
python3 part1Bayes.py -i sample_new_data/sample_new.csv -m pred
python3 part1Logit.py -i training_data/dataset.csv -m fit
python3 part1Logit.py -i sample_new_data/sample_new.csv -m pred

Part 2
I implemented part 2 by borrowing heavily from a Keras image processing routine I

found on one of Tensorflow's image classification tutorials. The provided model structure worked great, all of the changes I tried only degraded performance. I found 30 epochs to be optimal. I used the full resolution images in the training set and the fit routine completes fairly quickly. I have found that this Keras model is usually 100% accurate, or very close, in identifying the buildings, dogs, and faces, whether predicting on the training data or the test data.

When the accurate profile image recognition is combined with the statistics from the training data in 'dataset.csv', the results are not always accurate. This is due to the profile types of building, dog, face each having many instances of each star level. While there is a majority stars number for each, when using the probability for image class in calculations for the test data there are some prediction errors.

Following are the command lines for running the fit and predict routines for part 2:

python part2.py -i training_data/dataset.csv -m fit
python part2.py -i sample_new_data/sample_new.csv -m pred

Part 3
I setup the application for part 3 to use the logit text classifier developed in part 1 along with the image classifier in part 2. The probabilities for both classifiers for each star level are used. Accuracy is 100% on the test data.

Following are the command lines for running the fit and predict routines for part 3:

python3 logit.py -i training_data/dataset.csv -m fit
python3 part3.py -i sample_new_data/sample_new.csv -m pred

Conclusion
I have learned a lot this semester and it has been an amazing exercise creating these 4 models. I don't think I've ever written this much code in only 4 days before, wow am I beat! The two biggest surprises I've had were the logit text classifier and the Keras image classifier working so well. I believe you will find my code quality and results to be very good.

Thank you for all the amazing prep you've put into each lecture.