Part 1 : Digit Recognition

a) Status and stopping point-

In the first dataset for optical recognition of digits, the maximum accuracy percentage is around ~97% which can be considered as an acceptable value. Multi Layer Perceptron Classifier is used in this case. Based on the varying values for solver, learning_rate_init, number of hidden layers and alpha values the prediction accuracy varies. The best results are coming when the sigmoid function is used and alpha is 0.001. Also, the number of hidden layers are pretty high compared to the number of features (200 in this case) and the learning_rate is 0.01. The model does not perform that well when the number of hidden layers are very less for example 1. This attenuates the accuracy drastically and can be considered as the stopping point for this model and dataset.

b) Additional functions and analysis:

Multi Layer Perceptron can be considered as a proper classifier to classify this type of dataset. The code seemed quite efficient in runtime perspective as the data size seemed quite fine. The time taken by the data for loading and manipulation is not very high when the number of hidden layers are low. In case where prediction accuracy is quite better and the number of hidden layers are more, it adds to the time complexity of the code. Few experiments and analysis are done which leads to the following inferences:

- The number of hidden layers can be said as directly proportional to the prediction accuracy of the model. However, The proportionality graph is not linear.
- Other parameters such as learning rate and activation function also affects the prediction accuracy of the classifier for this dataset.
- Sigmoid activation function gives the best results in terms of prediction accuracy.

c) The most challenging part is the data formation. Understanding and Importing the data and the proper use of libraries took time. So code must be efficient to run over those data. That is what I learnt; to handle data and manipulating the data using efficient code methodologies.

Part 2: Amazon Data Set

a) Status and stopping point -

With Amazon Data Set, the maximum accuracy percentage that I get is around ~68%. I tried using different solvers for this dataset. With lbfgs the maximum accuracy that I got was ~60%. This classifier was faster but accuracy wasn't that great. So then moved to other classifiers namely 'sgd' and 'adam'. Both had almost similar accuracy rate of about 68%. But the thing to mention is 'adam' requires higher number of iterations for successful convergence. Whereas 'sgd' gave similar results with less number of iterations so settled for 'sgd' function.

b) Additional functions and analysis -

After analysing data I realized that it's necessary to clean the data as data set had rows with no reviews. To get rid of those rows I used dropna() method on data frame.
Then review itself had some html markup words. To counter that I used Beautiful soup. And then in order to get rid of stop words such as ‘is’,’and’ I used nltk package.
If I don’t perform all these operations the accuracy of the code is lower than 54%.

c) Challenges -

Amazon data set is really huge and playing with such huge data was challenging for me. I wanted to use stemming and lemmatization on this dataset but I wanted to have clear comparison between decision tree classifier and MLP classifier. That’s why opted not to do that this time. Another challenge that I faced was to settling for any particular value for classifier. As it has various variables that affect the behaviour of classifier. I had to try lots of combinations.