Student Name: Stefanos Vavoulas
Student ID: 1115201800014

Artificial Intelligence II
Assignment 1

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Important notes

The file I handed in includes variable test_df which should be initialized by passing the path of the test file.
The assignment was completed using python and google colab.

Section 1 - Preprocessing and Data Cleansing

In order to create a good model, I attempted to create a dictionary holding only lowercase words, excluding all newline characters, urls, mentions, emojis, punctuation, as well as a list of english stop words, meaning words that would be of no use to my model (such as: "a", "an", "the", "this", "that").
After the data cleansing, I'm printing the 10 most used words within the file.

Next, to complete the preprocessing, I keep all the reviews in a variable X, and all the corresponding ratings in a variable Y (Now modified to hold only 1=Positive, 0=Negative), and I split into a training set 85% and a validation set 15%.

Section 2 - Vectorization

After splitting into training and validation sets, the reviews held in variables X_train and X_val need to be vectorized into usable data. Thus, I vectorize using Count Vectorizer and Tfidf Vectorizer, with min_df=0.01, meaning words that have been rarely used in the reviews are disregarded, ending up with a dictionary of about 1600 words.
Generally, I found greater success using Tfidf Vectorizer, which is to be expected, since tfidf is an expansion of Count Vectorizer

Section 3 - Training and Results, Testing Vectorizers and Hyperparameters

Starting with an example using the hyperparameters, along with the vectorizer, that I found to give the best results after testing them on multiple random states of the shuffled data split. Includes precision and recall scores. Lastly, created plots showing convergence of the f1 scores from the training and the validation set.

Through testing, I found the most succesful vectorizer to be tfidf, paired with solvers newton-cg and lbfgs, penalty l2 and hyperparameter C = 0.1, that returned an f1 score larger than 87% for most random states without overfitting - The results on both the training and the validation set were quite close (around 0.005 difference). The precision and recall scores were also close to 87%,

showing that the amount of false positives was very close to the amount of false negatives, as we would've wanted in this particular assignment.
As expected, lower values on C caused my model to underfit, since training wasn't as dependant to the training set, while higher values caused it to overfit, with the training set getting far better f1 scores than the validation set.

With Count Vectorizer, I found that my model was more often overfitting, and so, by setting C=0.01 I managed to get results that were (I think) in the sweet spot between under and over fitting, as shown in the second plot.
I also found the rest of the solvers to not be as succesful, with sag and saga being slower as well.