**NEURAL NETWORK DEEP LEARNING**

**ASSIGNMENT 9**

**700748022**

**YAMINI SARASWATHI BORRA**

**GitHub:**

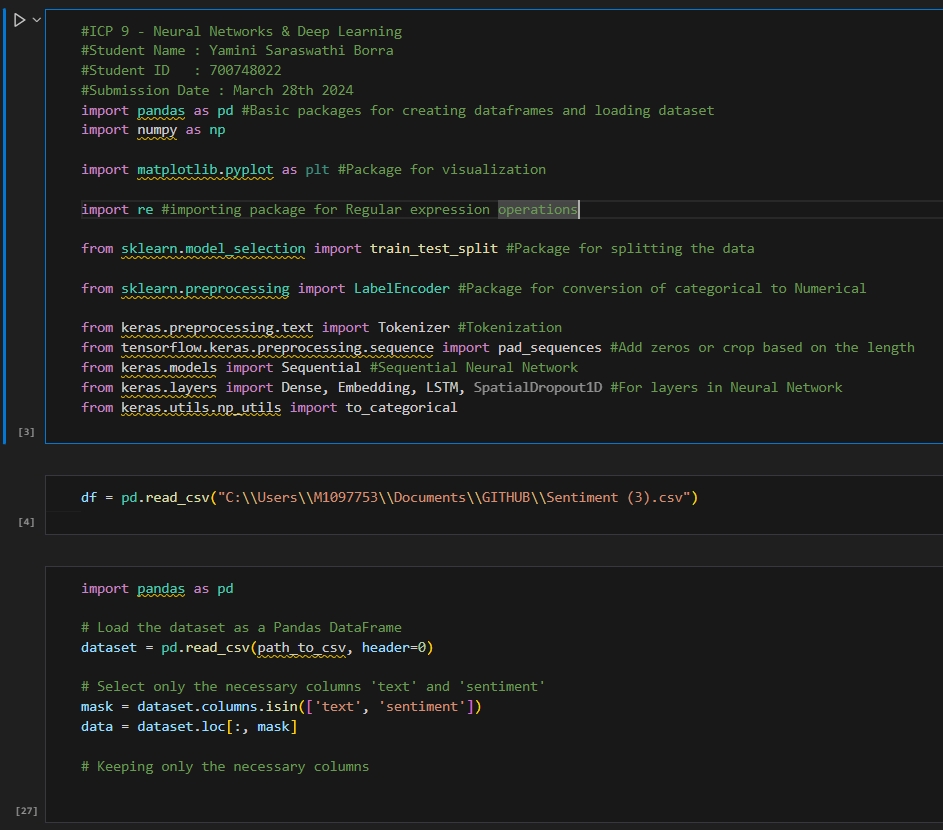
Repository URL for the source code:

<https://github.com/YaminiSai786/CS5720-Neural-Networks-Deep-Learning---ICP>

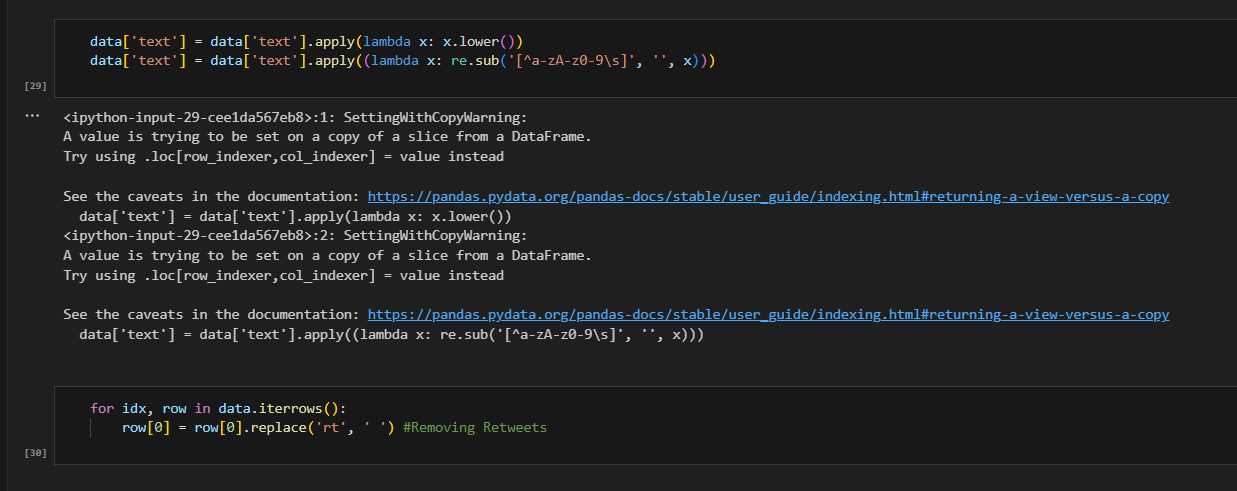
**Video Link:**

<https://github.com/YaminiSai786/CS5720-Neural-Networks-Deep-Learning---ICP>

**1. Save the model and use the saved model to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump”)**



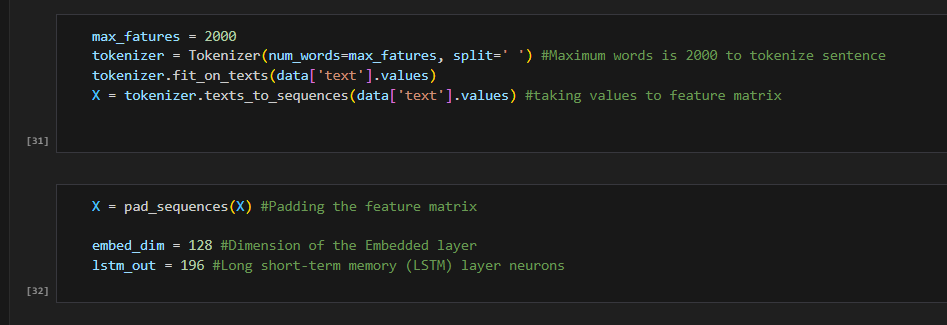
Importing the packages, reading the .csv file, loading the dataset as a Panda DataFrame, selecting only the necessary columns that are ‘text’ and ‘sentiment’.



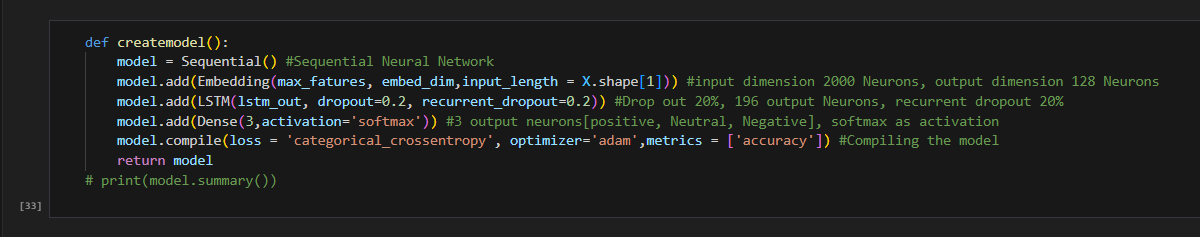
Using lambda function to convert to lowercase.

Using re.sub() regular expression to replace any non-alphanumeric characters and non-whitespace characters with an empty string.

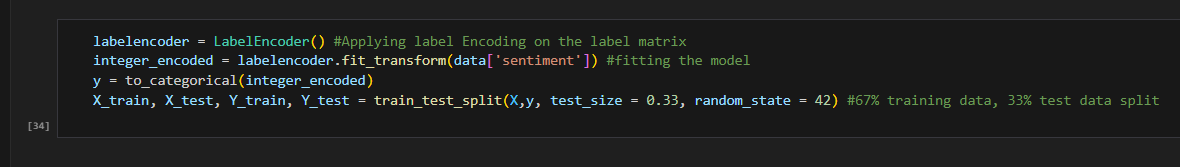
Then, Removing the retweets.



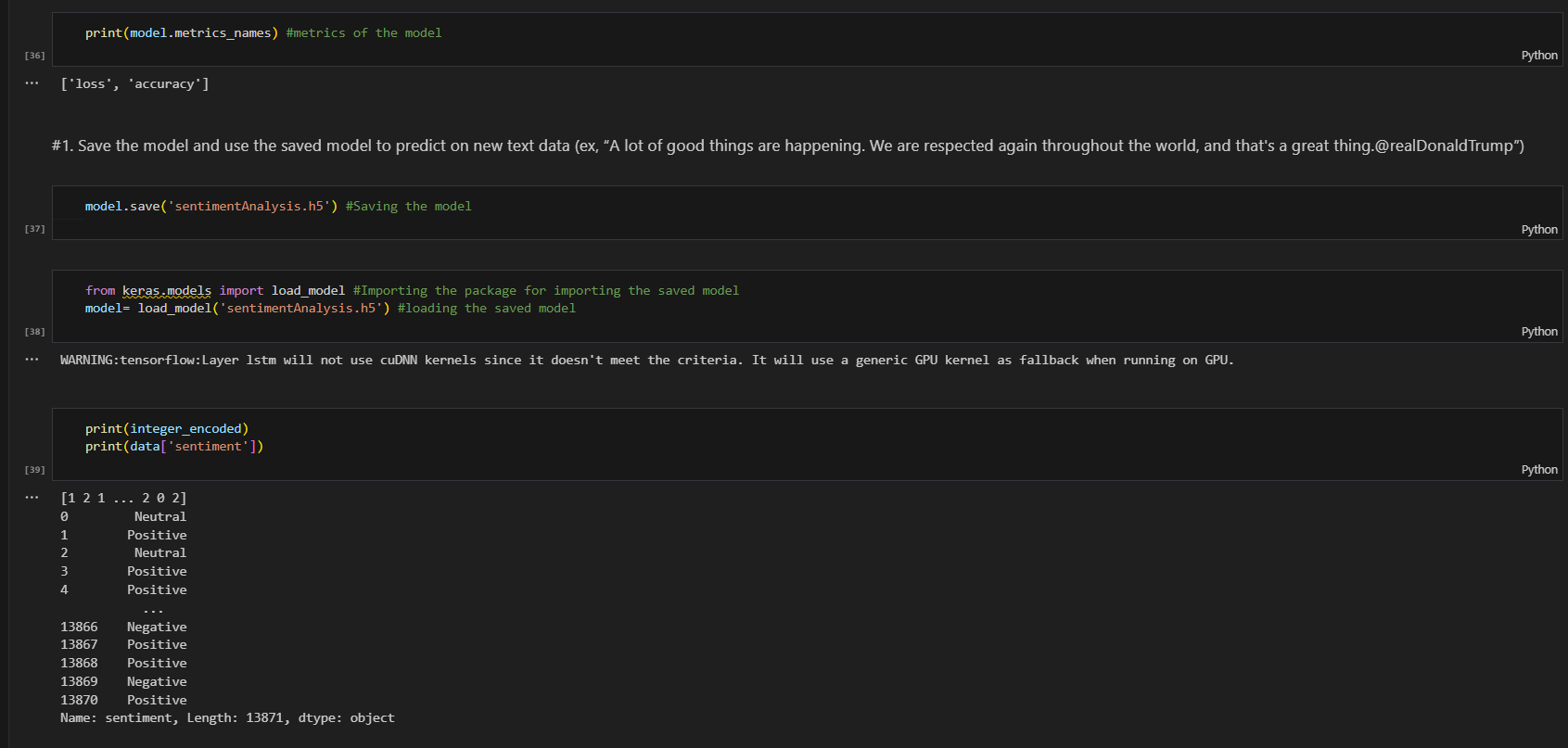
Total number of words to tokenize a sentence is limited to 2000 and takes values to feature matrices. Now padding the feature matrix with Dimension 128 and long short-term memory to 196.



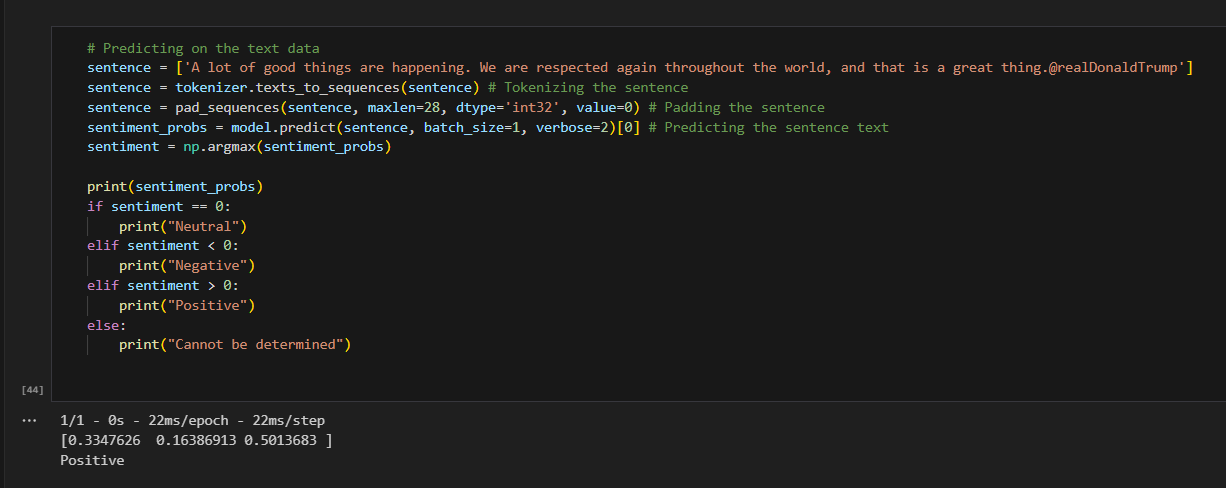
the ‘createmodel()’ function creates a Sequential model with an Embedding layer to handle the input data, an LSTM layer to extract features from the input sequences, and a Dense layer with softmax activation to predict the sentiment class. The model is then compiled and returned.



the ‘fit\_transform()’ method of the ‘LabelEncoder’ instance is to convert the categorical sentiment labels in the data DataFrame to numerical values. The resulting numerical values are stored in the integer\_encoded variable. Then this splits the input data (X) and target data (y) into training and testing sets using the train\_test\_split function from the Scikit-learn model selection module. The testing set is set to be 33% of the total data (test\_size=0.33) and the random seed is set to 42 (random\_state=42). The resulting variables are X\_train (training input data), X\_test (testing input data), Y\_train (training target data), and Y\_test (testing target data).

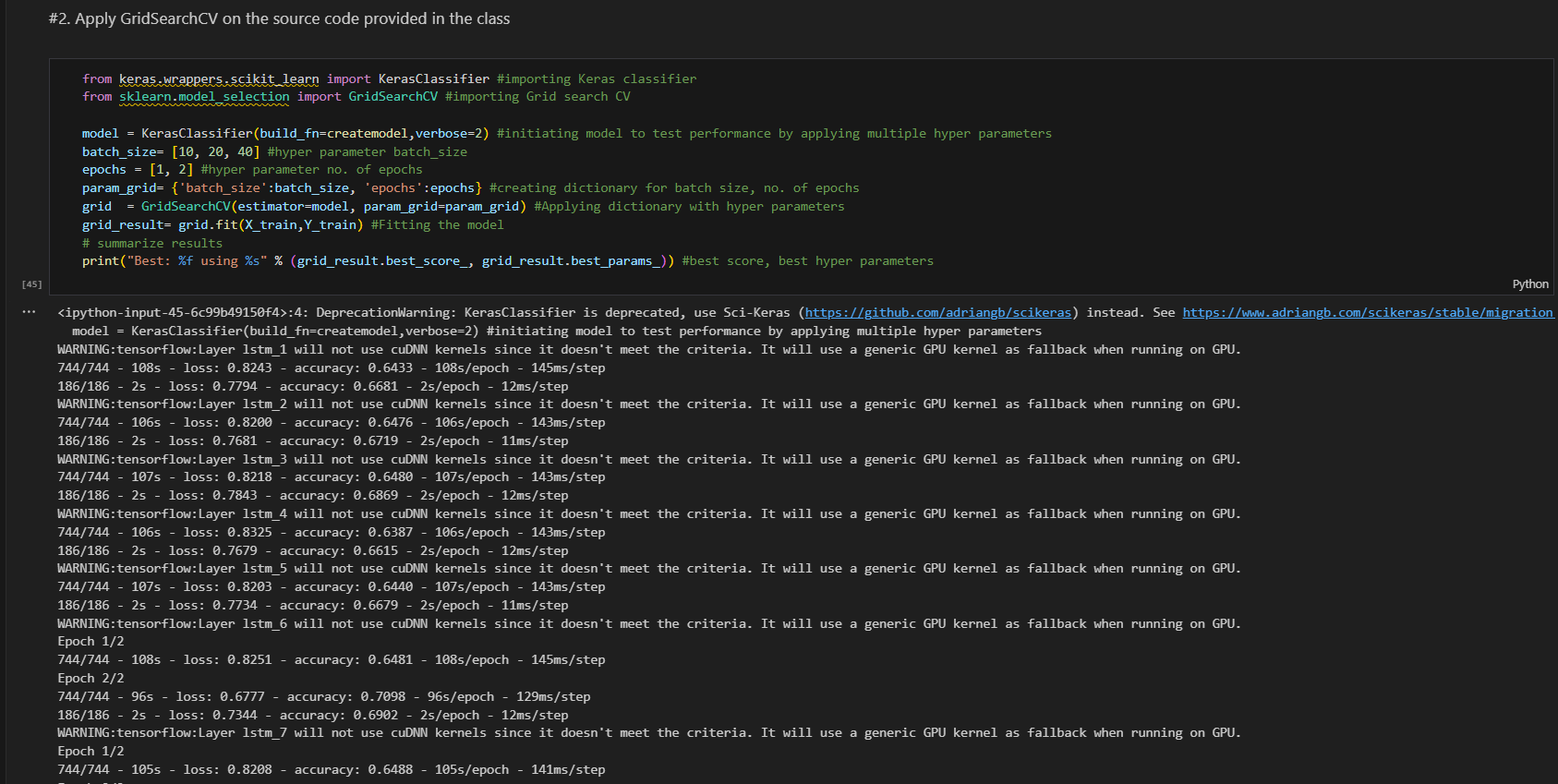


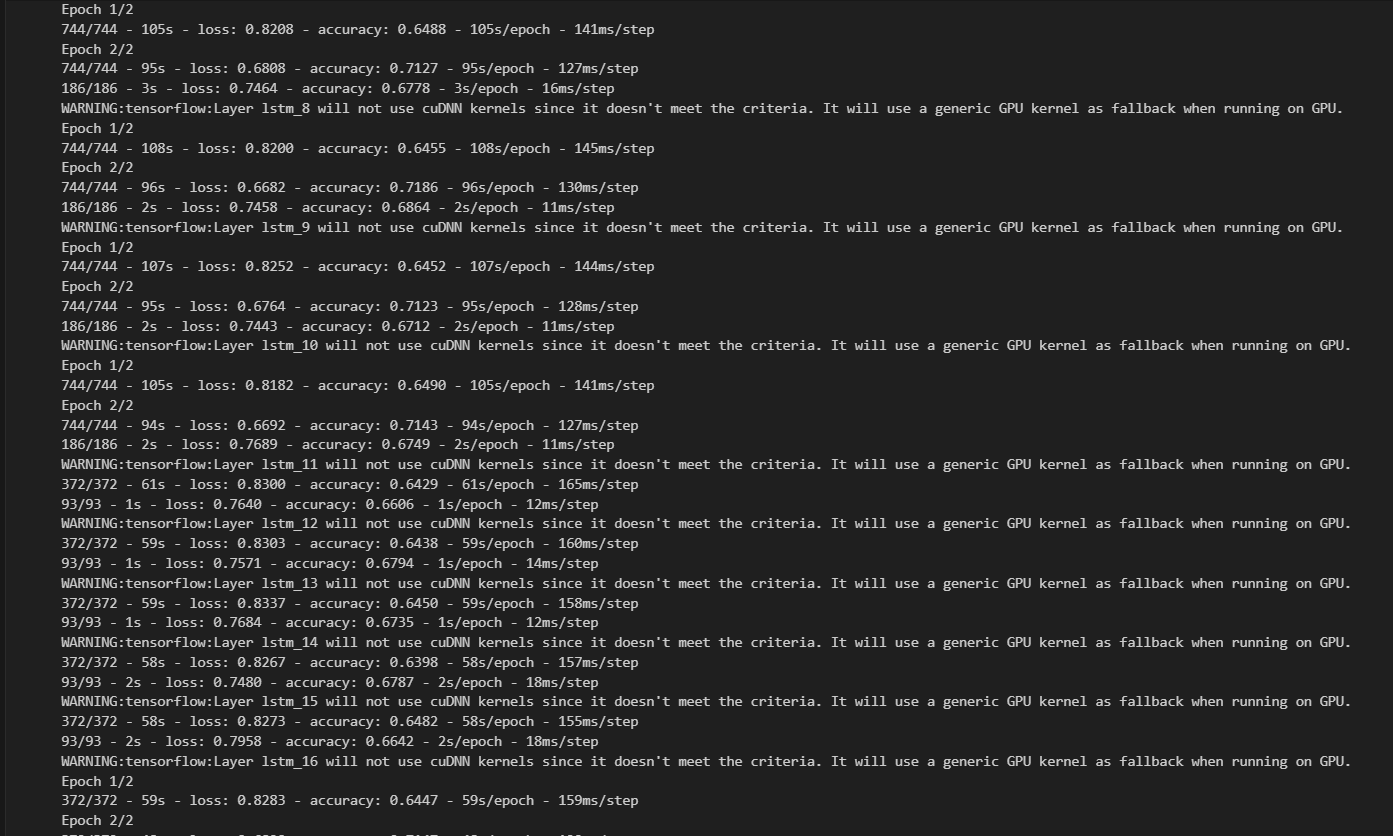
Now printing the metrics, saving the model, then printing the accuracy of each data.

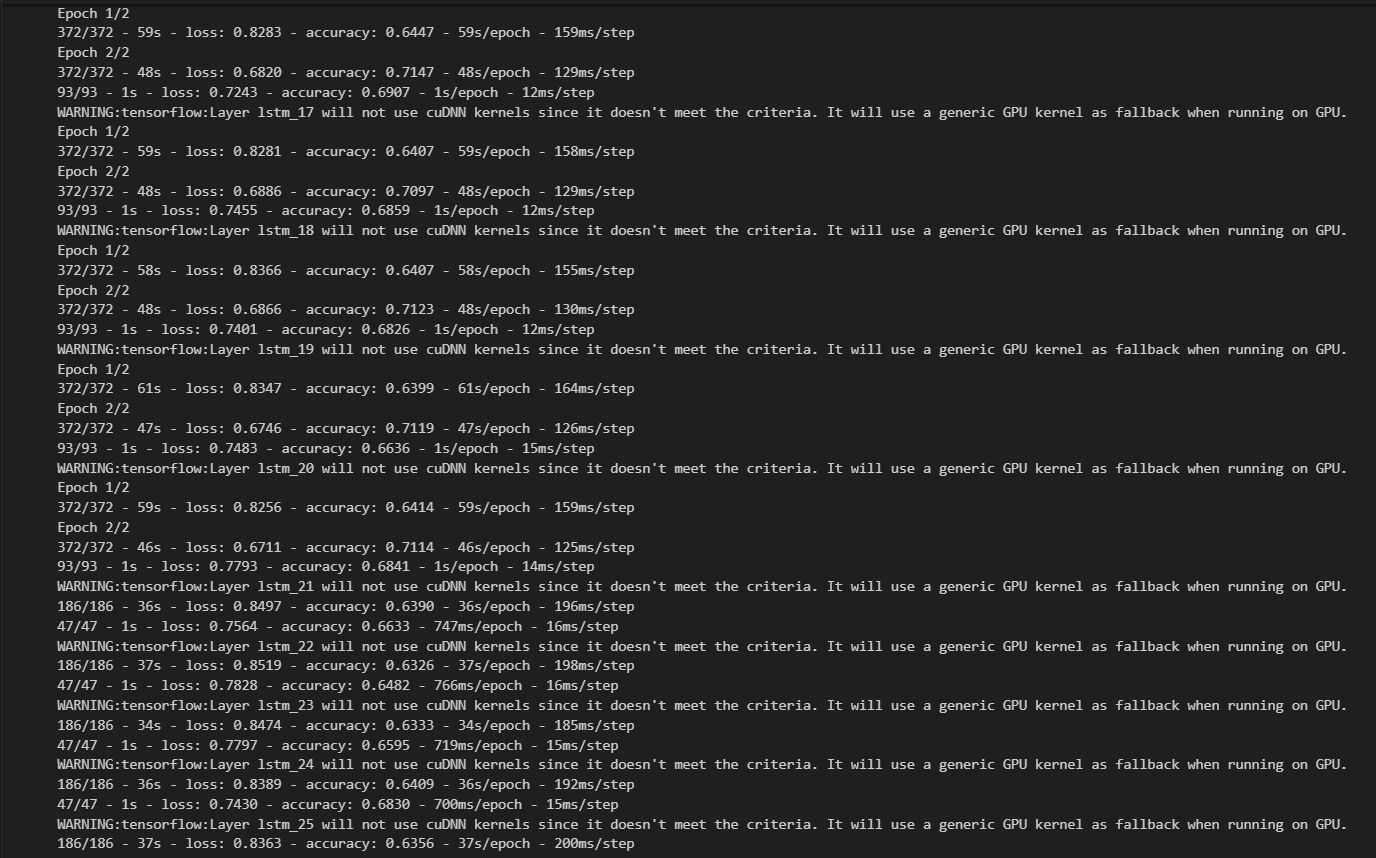


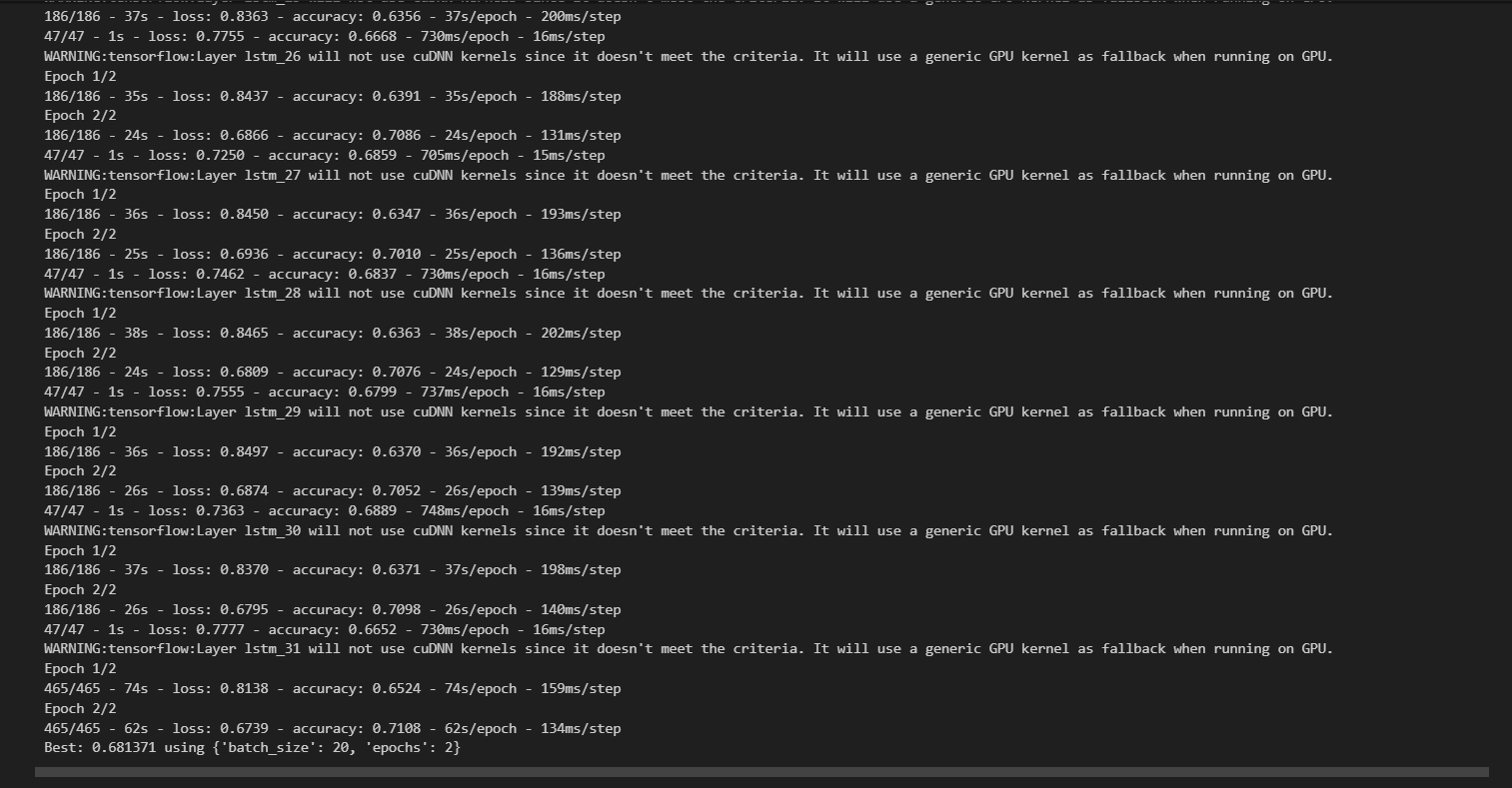
Now predicting the model by testing on a sentence. And the result is positive.

**2. Apply GridSearchCV on the source code provided in the class**









Initiating model to test performance by applying multiple hyper parameters. Defining the batch\_size, no. of epochs and then creating a dictionary for batch size & no, of epochs. Then fitting the model and summarizing the results. The accuracy is 0.7108 with batch size 20 and epochs as 2.