EEP 596A Advanced Topics in Signal and Image Processing

Mini Project 2 Report

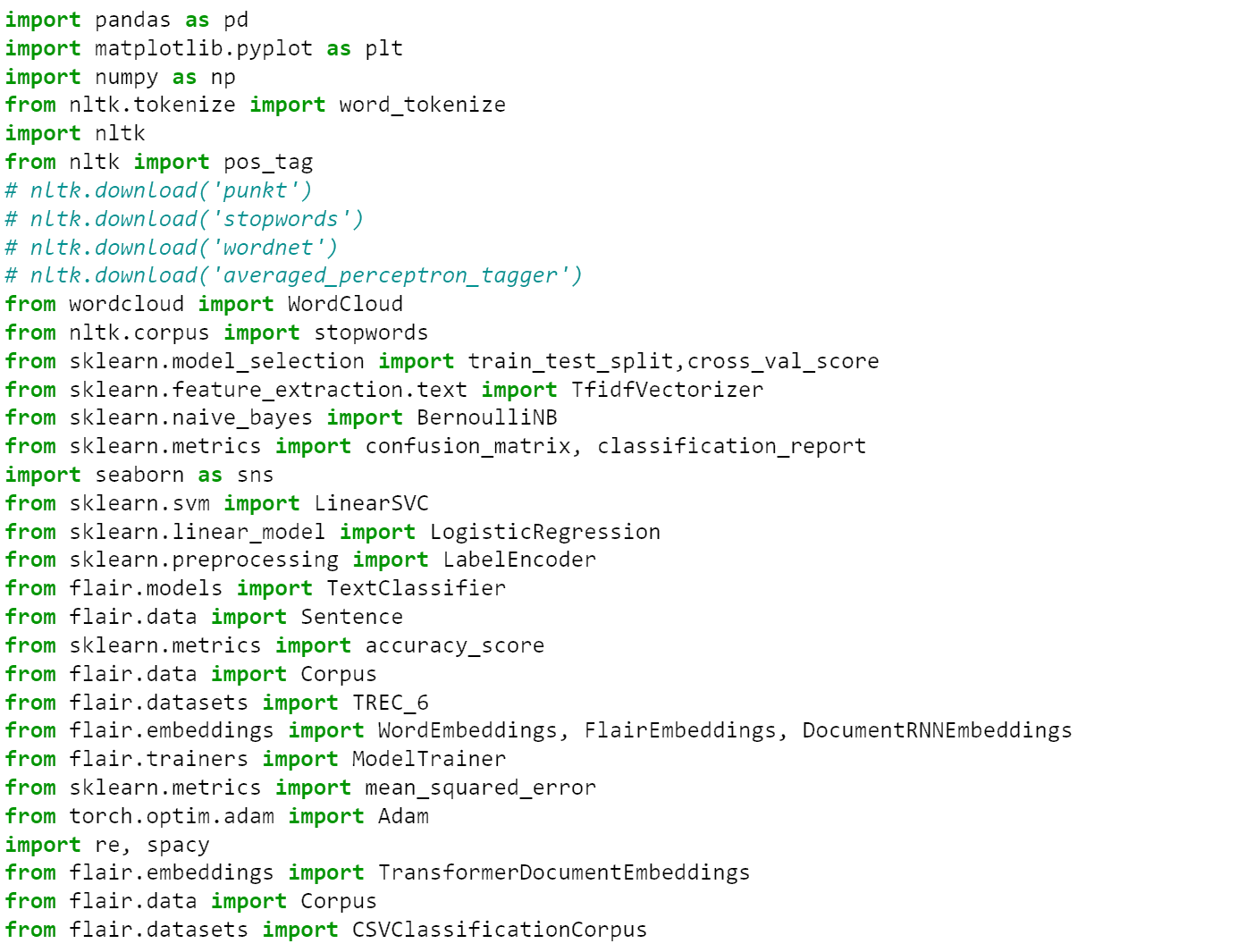
# Introduction and Overview

This assignment is about Twitter Sentiment Analysis. In this project, we are asked to classify the sentiment of tweets from a real dataset by building and training some practical machine learning models. Sentiment analysis is meant to classify the emotion and perspective expressed from original text sources. It is usually that a few sentiment data details appeared by analyzing tweets. This data is contributing to learning about the opinions of people. Therefore, we are about to develop an automatic machine learning model to judge the sentiments of tweets. However, it is difficult to be developed because of some noise, which could be called ‘useless characters’ like punctuations and URLs. So, in this article, we are about to develop a pipeline to analyze the Twitter dataset. It includes three classifiers: Logistic Regression, Bernoulli Naive Bayes, and SVM along with using Term Frequency- Inverse Document Frequency (TF-IDF). The criterion for judging models is accuracy and F1-score.

# Preliminary Work

In this section, we imported various necessary dependent libraries, read and loaded the dataset, visualization, and built a word cloud to show the most common words in both positive and negative datasets. Finally, we preprocessed the dataset, including converting it to lowercase, cleaning the datasets from stopwords, punctuations, URLs, numbers, and common words. We did all the things above to decrease the useless attributes of datasets and make them cleaner than before. We supposed that in this way, the models would be trained in a better condition and finally arrived at better accuracies and F1-scores.

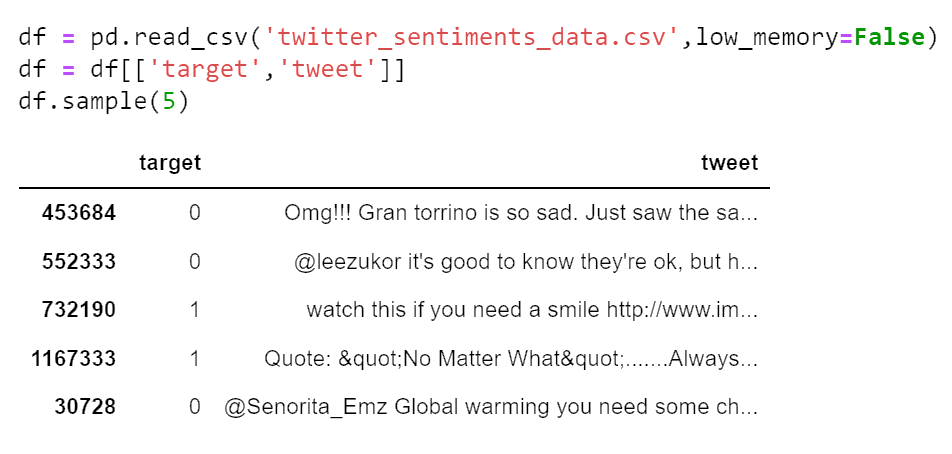
## Importing Necessary Dependencies



Picture 2- 1

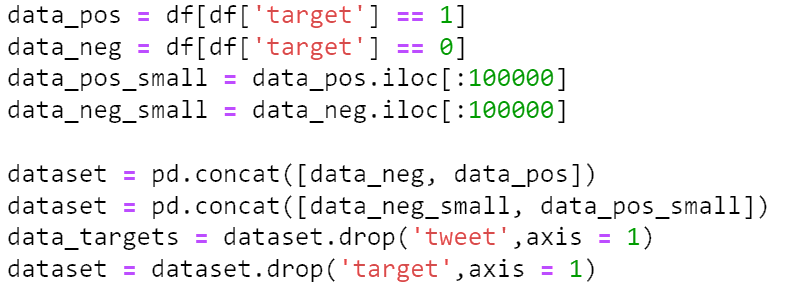
In this assignment, we applied several libraries, including Pandas, NumPy, Sklearn, Torch, NLTK and so on to support our whole work.

## Reading and Loading the Dataset



Picture 2- 2

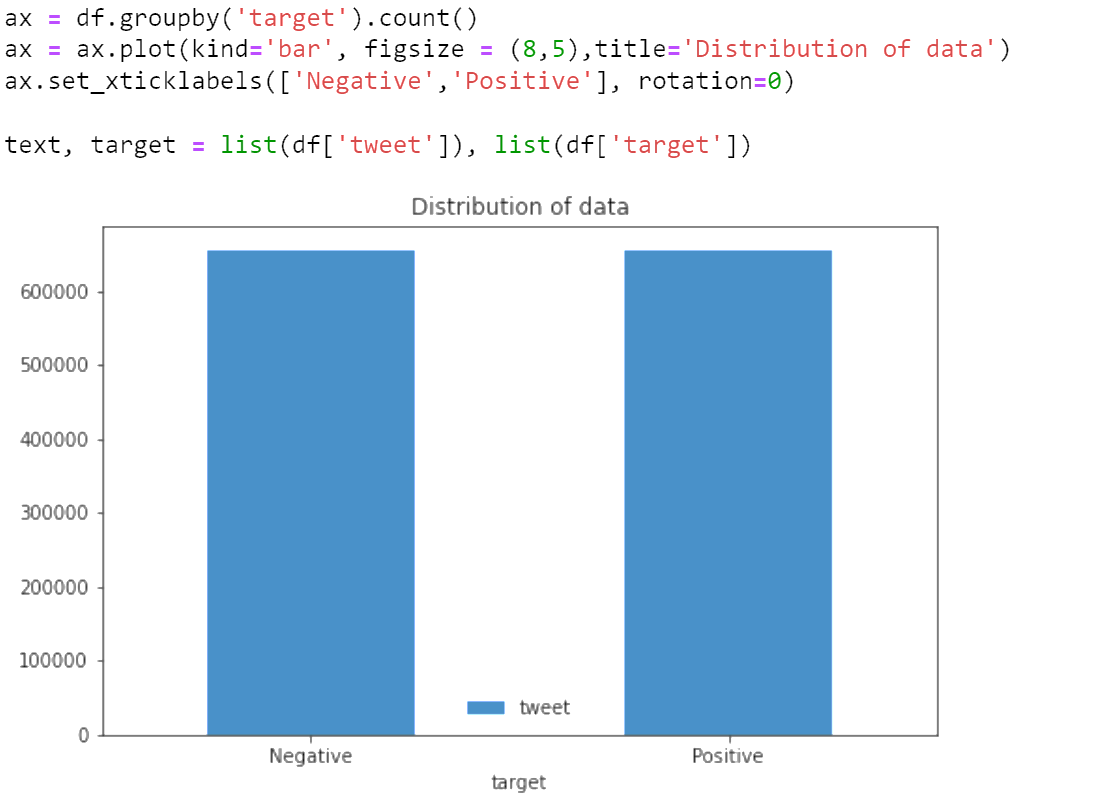
We read the dataset file by the function read\_csv in Pandas and just used two columns of it: target and tweet, to remove other useless information.



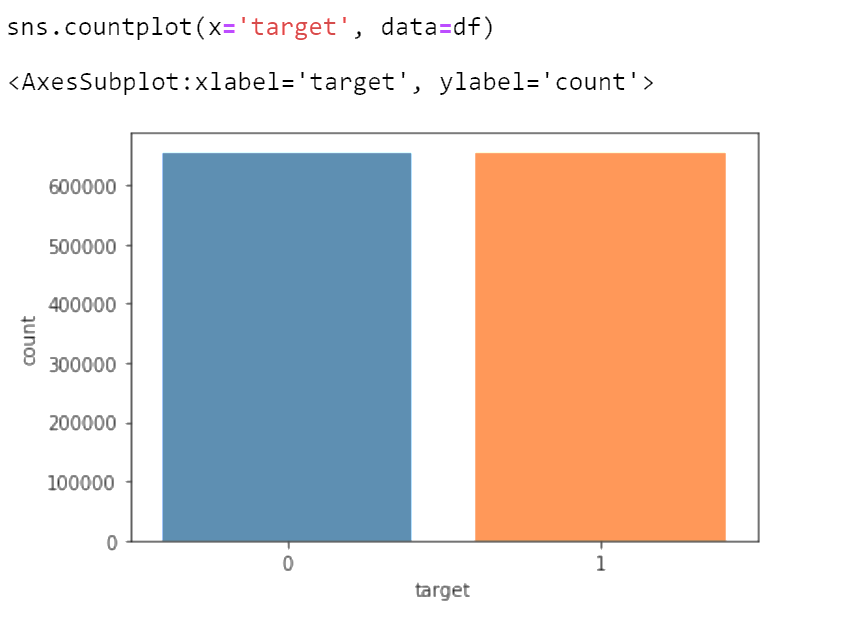
Picture 2- 3

Then divided the dataset into two sections: a positive one and a negative one. Chose ten thousand sentences for each set since the original dataset is too large to be completed quickly.

## Data Visualization of Target Variables



Picture 2- 4



Picture 2- 5

In this part, we visualized the whole dataset and showed the data conditions in it by making bar graphs. We can see that the data on both sides are approximately equal.

## Word cloud



Picture 2- 6

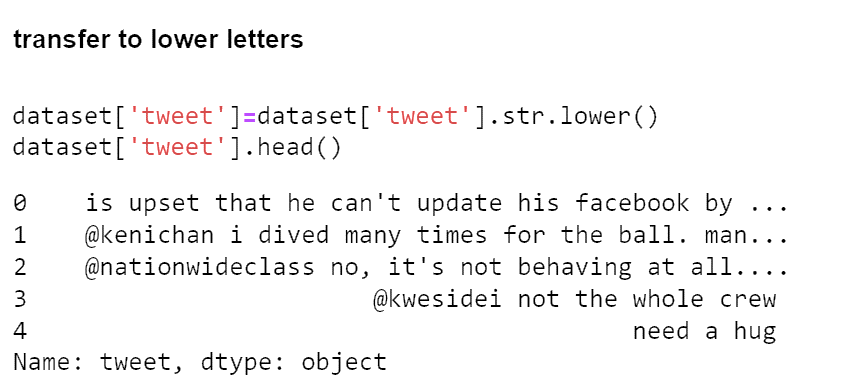


Picture 2- 7

A word cloud can demonstrate the common words in the dataset, so we build two clouds, one is for negative and another is for positive. In these two pictures, we realized that people usually say ‘day’, ‘work’, ‘now’, ‘go’, ’time’, and ‘today’, etc. when they feel not good. By contrast, ‘good’, ‘thank’, ‘love’, ‘day’, ‘lol’, ’now’ are some predictors of positive sentiment. However, we could learn that some words like ‘day’, ‘today’, ‘going’ occurred on both sides, so cleaning the dataset would be what we should do for the next step.

# Data Processing

## Converting to Lowercase



Picture 3- 1

We converted all the letters to lowercase by lower() function.

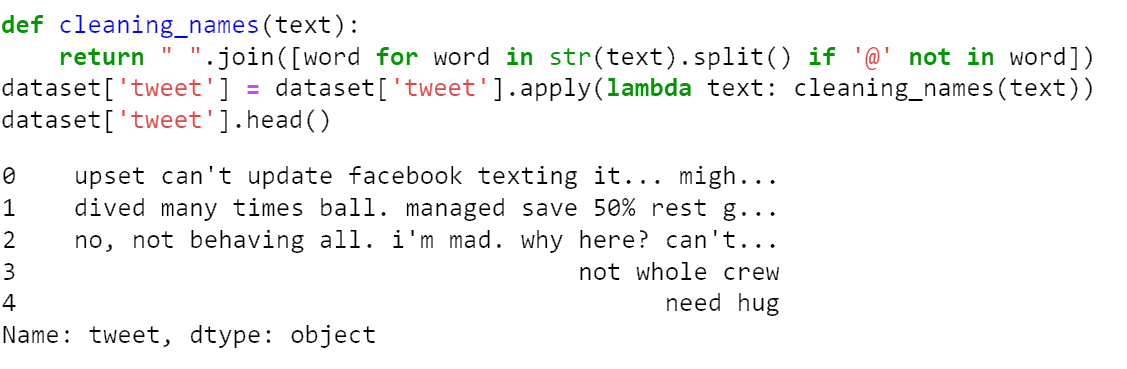
## Defining and Cleaning Stopwords



Picture 3- 2

We defined the stopwords library and removed the words in that library from data to eliminate the influence made by stopwords.

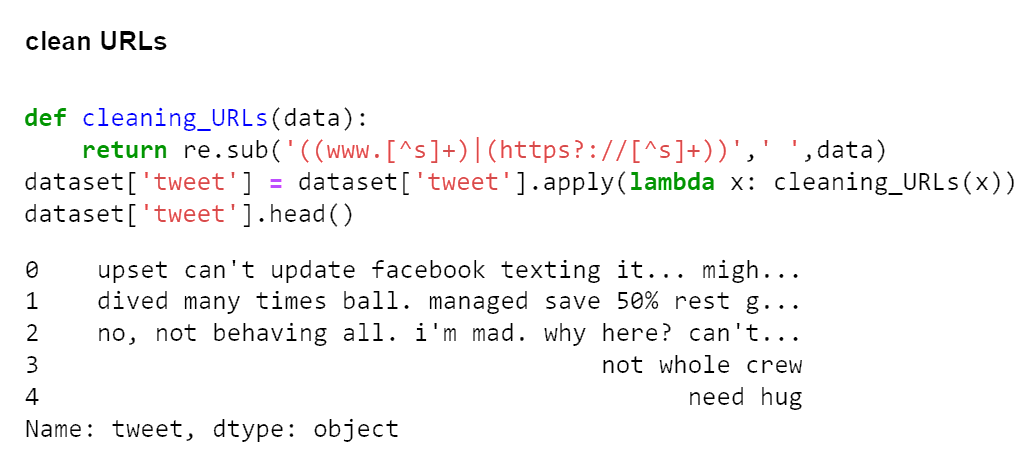
## Removing Names



Picture 3- 3

We removed names that were cued by the label’@’. We supposed that nicknames were not attributed to classification.

## Removing URLs



Picture 3- 4

URLs are non-sentiment parts of sentences, so we removed them.

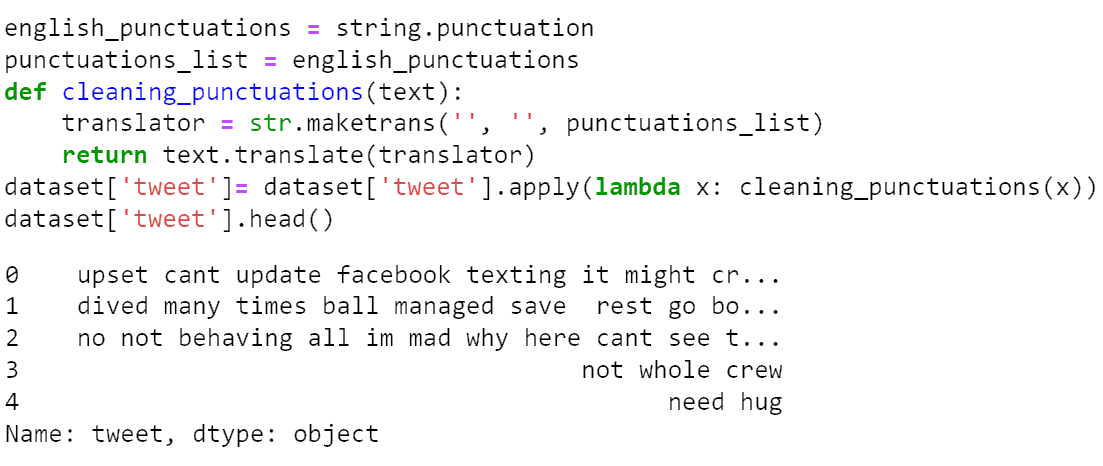
## Removing Numeric Numbers



Picture 3- 5

We removed numbers from 0 to 9.

## Removing Punctuations



Picture 3- 6

Removing punctuations is beneficial for training and obtaining a higher accuracy.

## Tokenizing Sentences



Picture 3- 7

We perform a word tokenizer on all sentences using custom functions and let every single word be a string in the list.

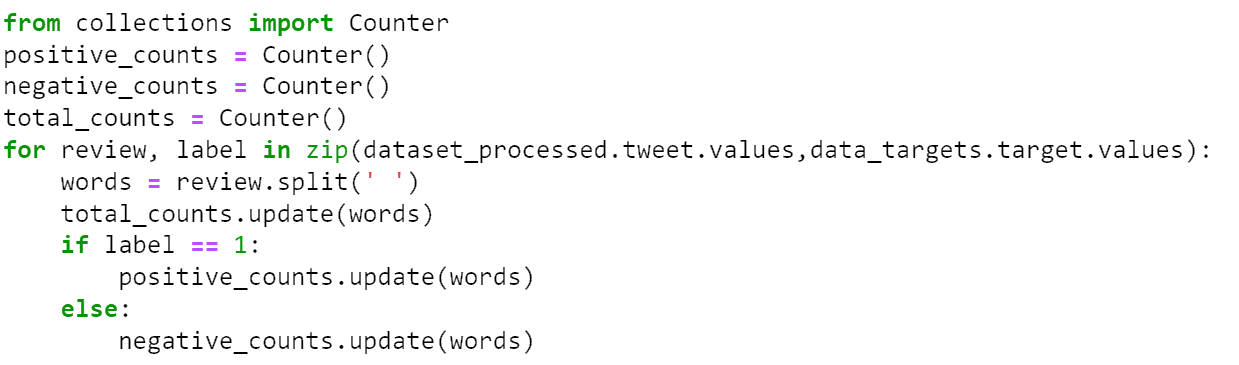
## Lemmatizing words



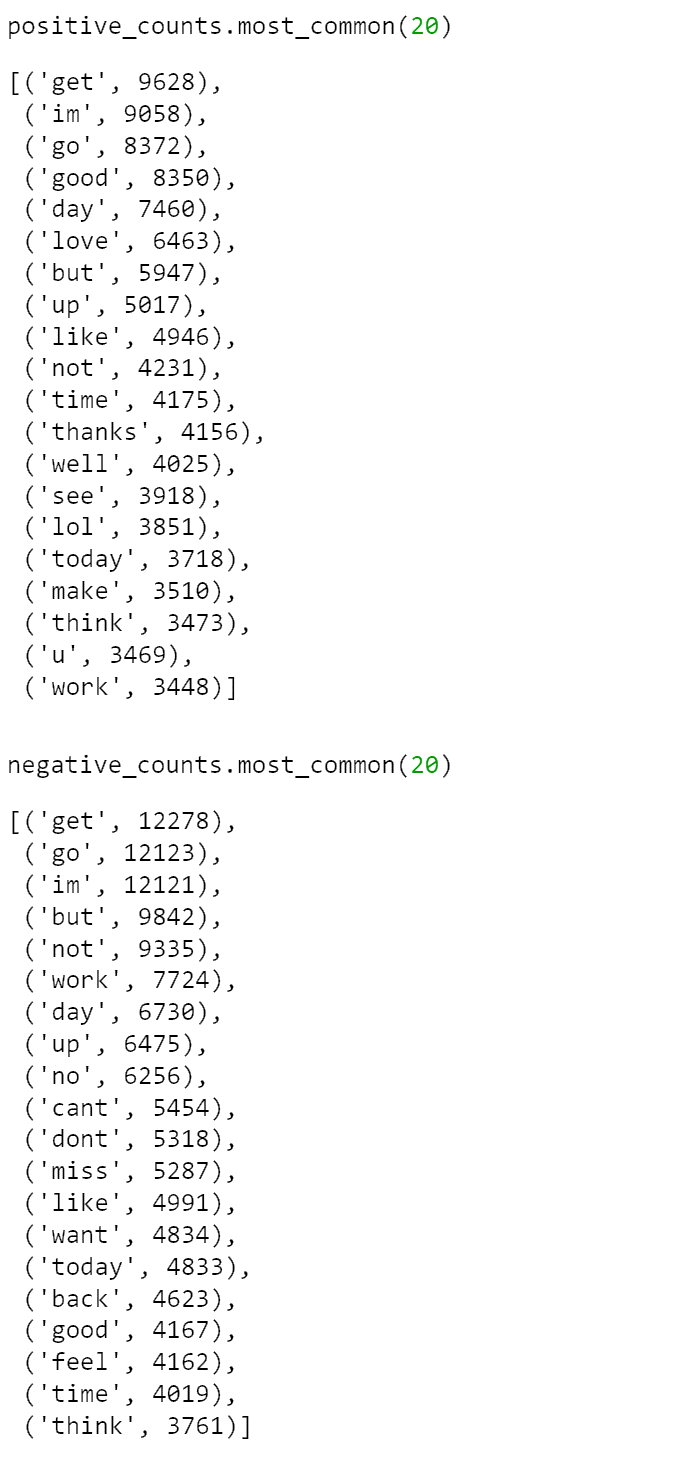
Picture 3- 8

We recognized the tag of each word like ‘NN’ for none, and ‘JJ’ for adjective words by pos\_tag () function, and lemmatized the words depending on the result of pos\_tag, through WordNetLemmatizer () function in the nltk library.

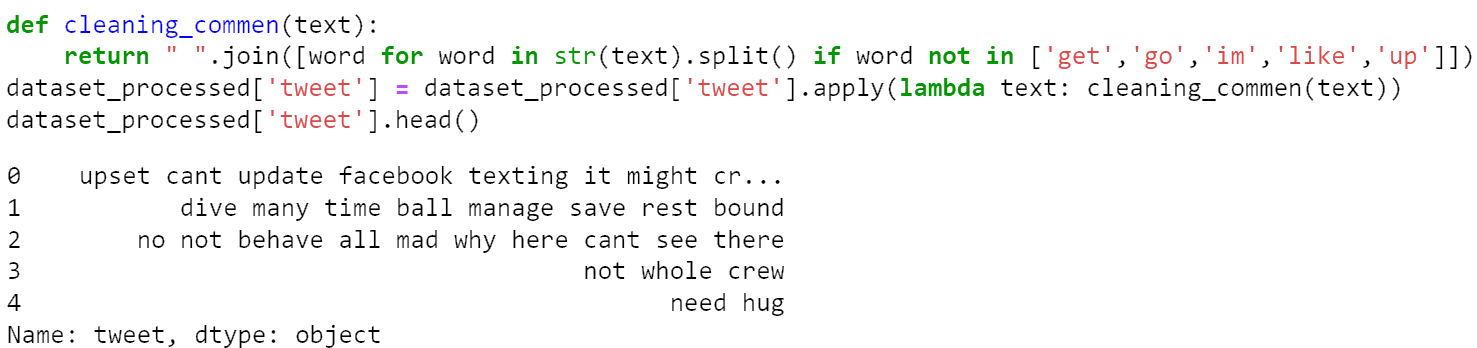
## Removing the most common words



Picture 3- 9



Picture 3- 10



Picture 3- 11

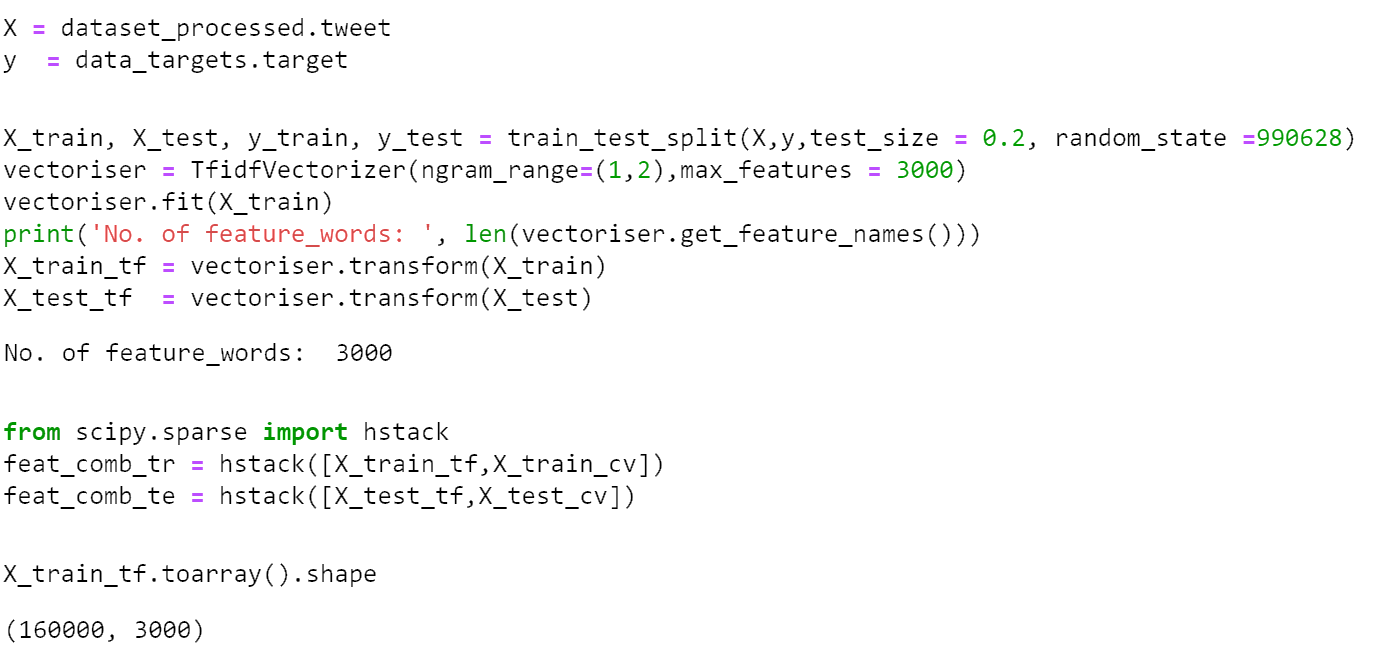
In this part, we realized that some high-frequency words, like ‘get’, ‘go’, on both sides will affect the model's judgment. Thus, we checked them and removed them.

# Supervising learning models

First of all, split the dataset into two parts, for train and test. Then, we built three models: Bernoulli Naïve Bayes, SVM, and Logistic Regression to train the dataset. The idea behind choosing these models is that we want to try all the classifiers on the dataset ranging from simple ones to complex models and then try to find out the one which gives the best performance among them. After training the model we then applied the evaluation measures to check how the model is performing. Accordingly, we used the following evaluation parameters to check the performance of the models respectively: Accuracy Score, Confusion Matrix with Plot, and ROC-AUC Curve.

## Splitting Datasets and Vectorizing

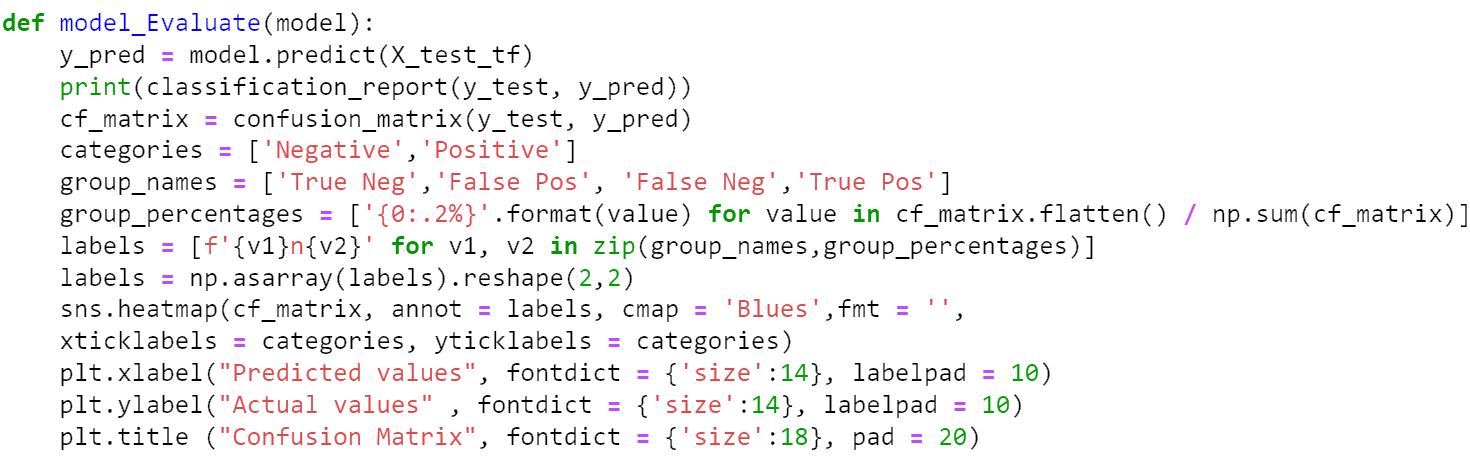
We split the dataset into two parts: X\_train and X\_test by train\_test\_split function, which accounts for 80% and 20% separately. And then vectorized all the words to 3000 features by TfidfVectorizer() function.



Picture 4- 1

## Building evaluation function

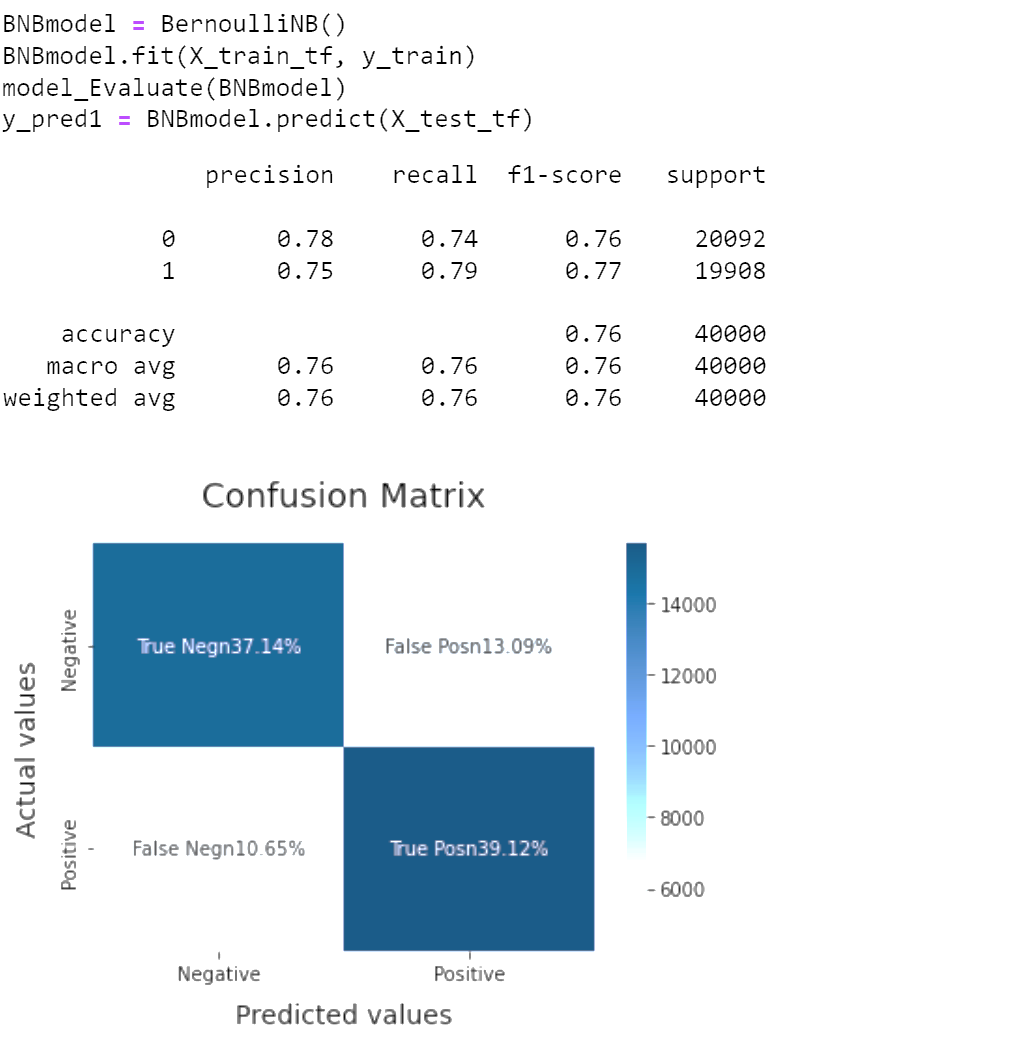
The evaluation model function was defined by us. We made the predictions of the test module Besides, in this function, we integrated some other functions like confusion\_matrix () to show the confusion matrix and show the classification report by classification\_report ().



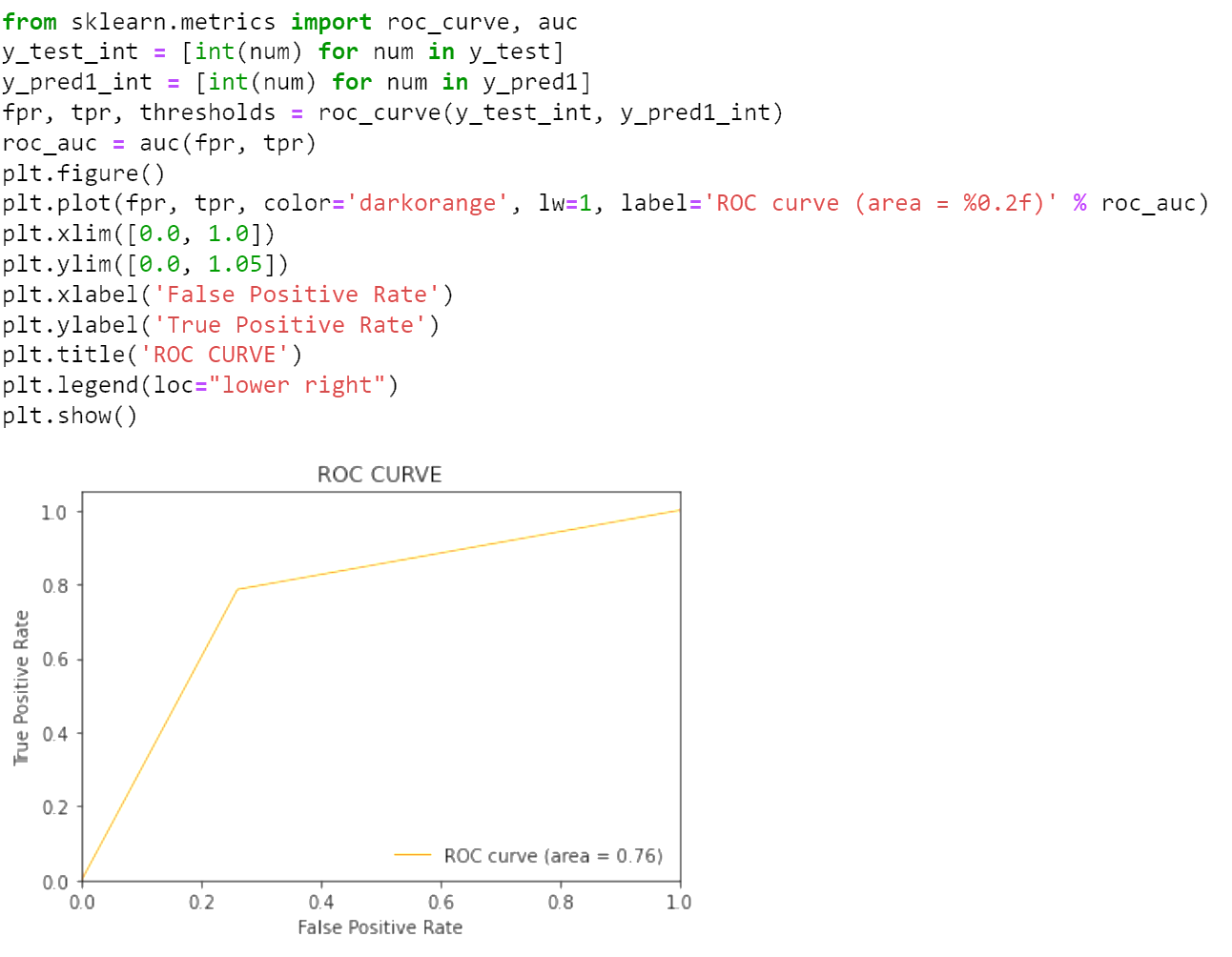
Picture 4- 2

## Bernoulli Naïve Bayes

We used the Bernoulli Naïve Bayes model and show the confusion matrix and ROC curve graph below. We could see that the Bayes model did not perform well enough, which just had 0.76 precision and F1-score.



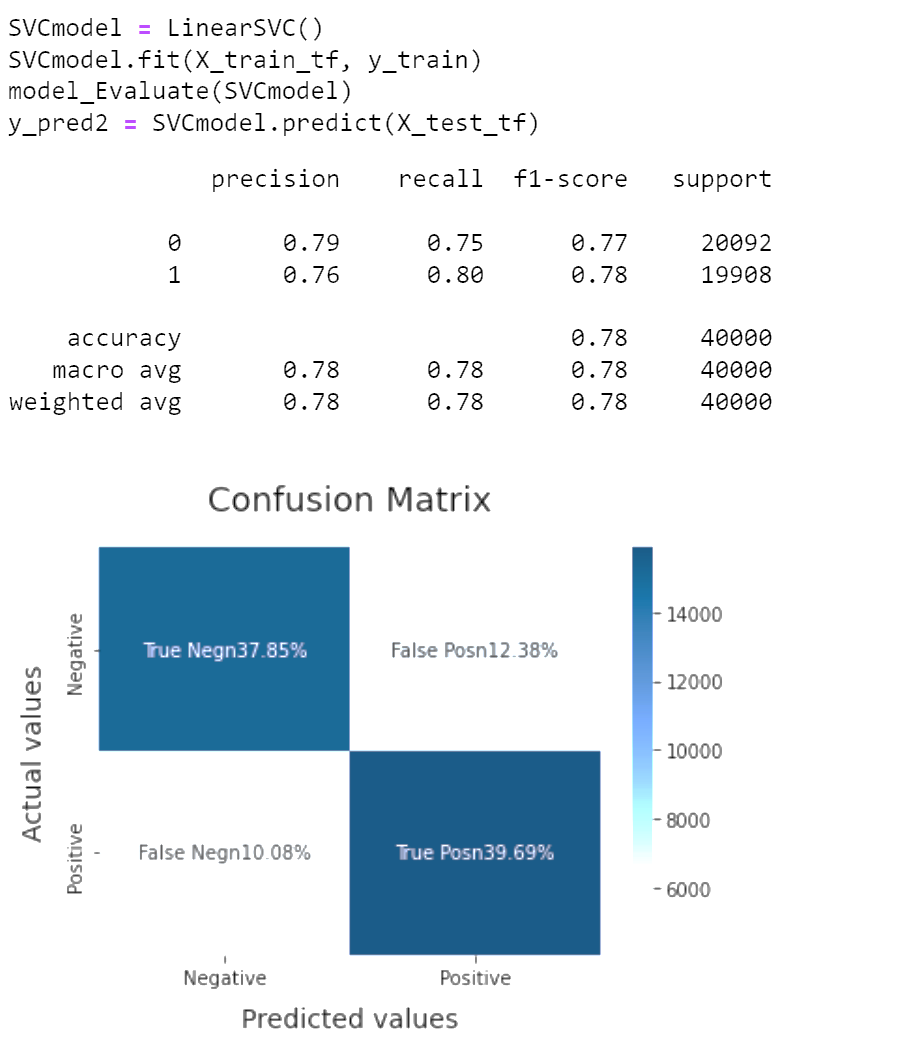
Picture 4- 3



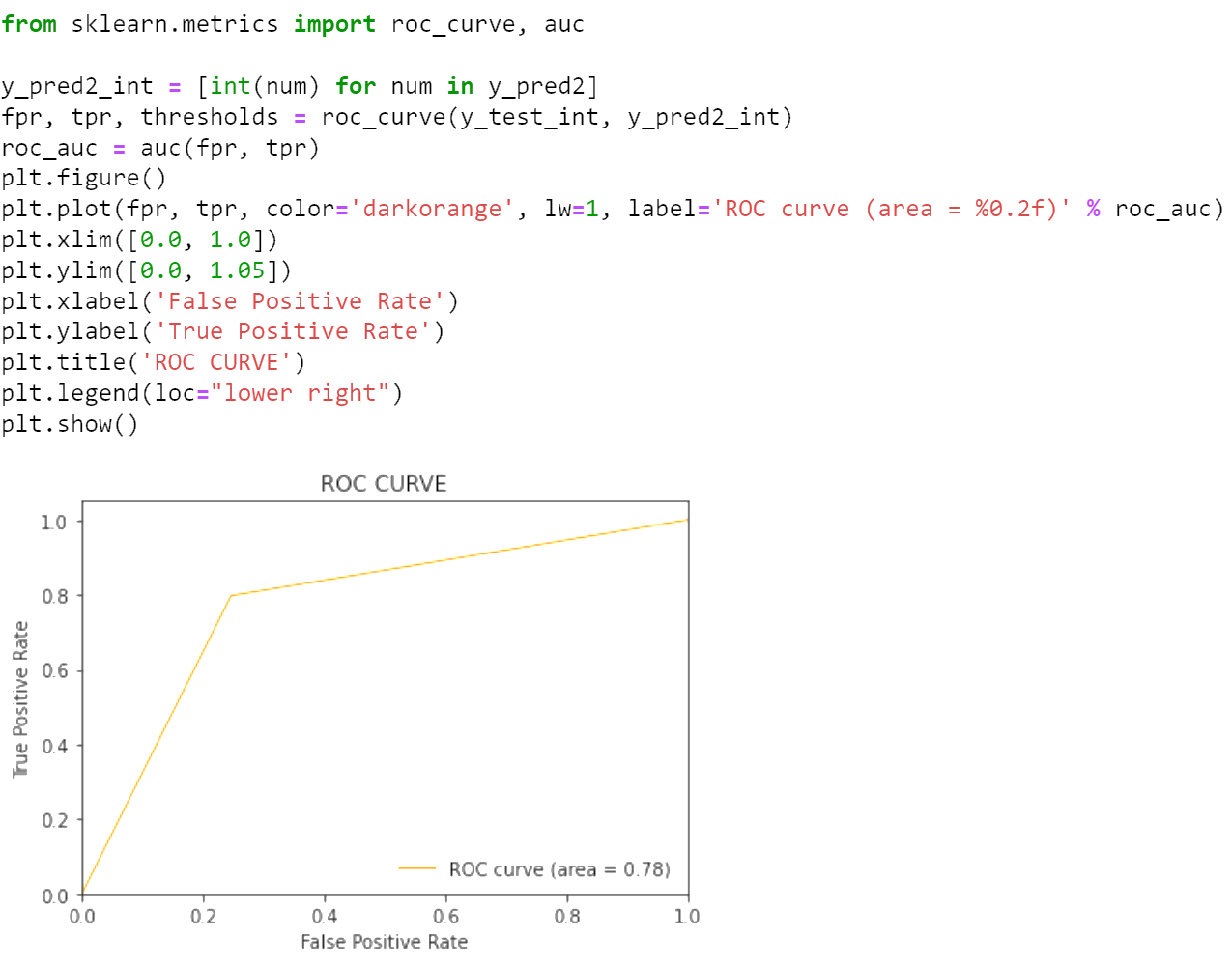
Picture 4- 4

## Support Vector Machine

Then, we chose the SVM model. Similarly, we showed the confusion matrix and ROC. It worked better than Bayes that it had 0.78 precision. However, its accuracy was still not high enough.



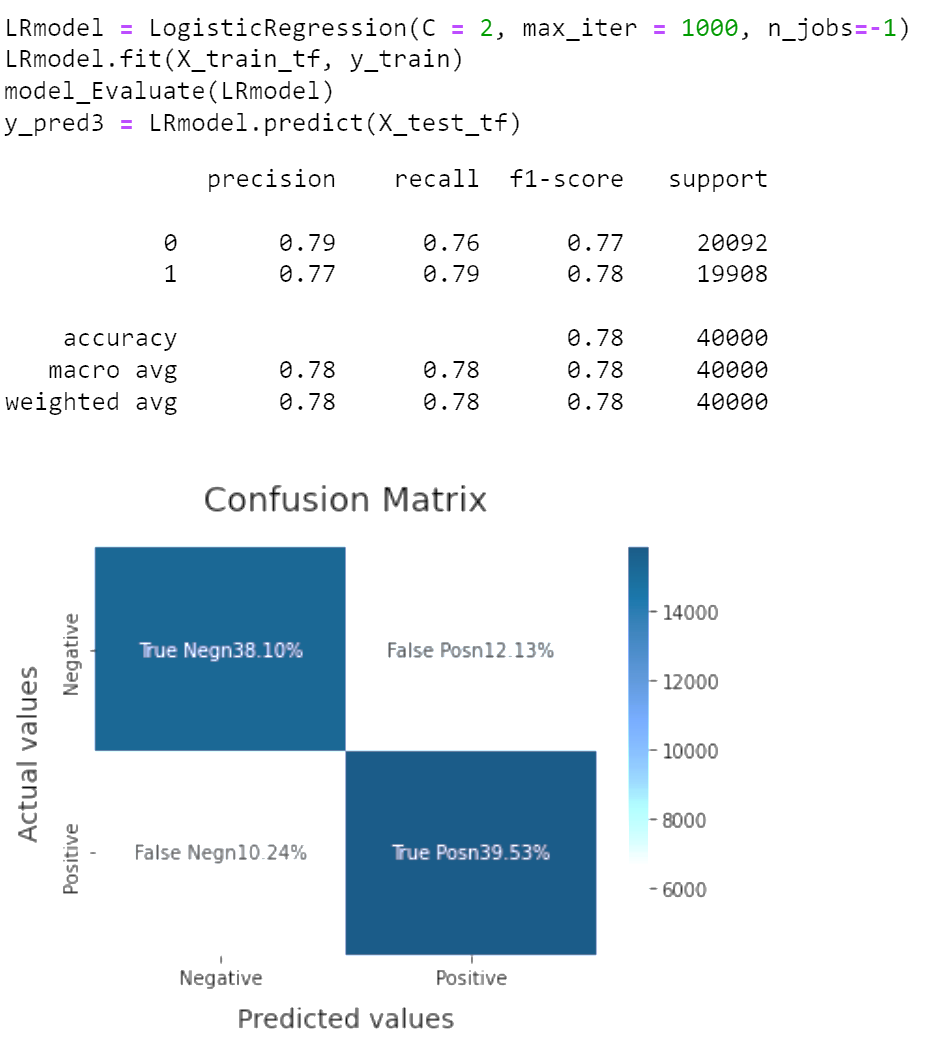
Picture 4- 5



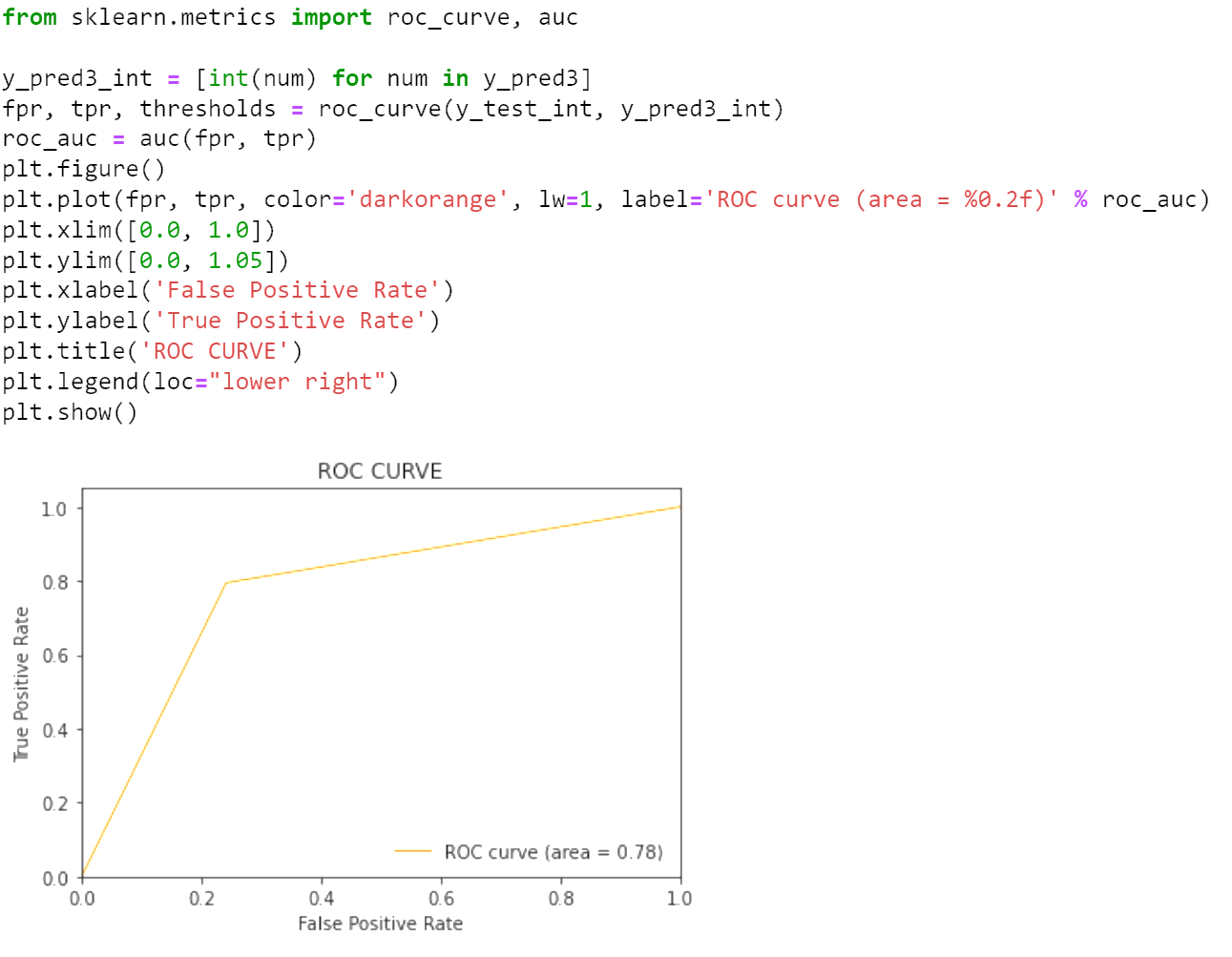
Picture 4- 6

## Logistic Regression

Finally, we tried the logistic regression model and showed the matrix and ROC.



Picture 4- 7



Picture 4- 8

## Comparison of Supervising Models

Chart 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| Bernoulli NB | 0.76 | 0.76 | 0.76 | 40000 |
| SVC | 0.78 | 0.78 | 0.78 | 40000 |
| Logistic | 0.78 | 0.78 | 0.78 | 40000 |

In this chart, we can see that logistic regression and SVC share the value of matrices (0.78), which are higher than BNB—only 0.76 for all. However, 0.78 is not high enough in this assignment, so we decided to build deep learning models.

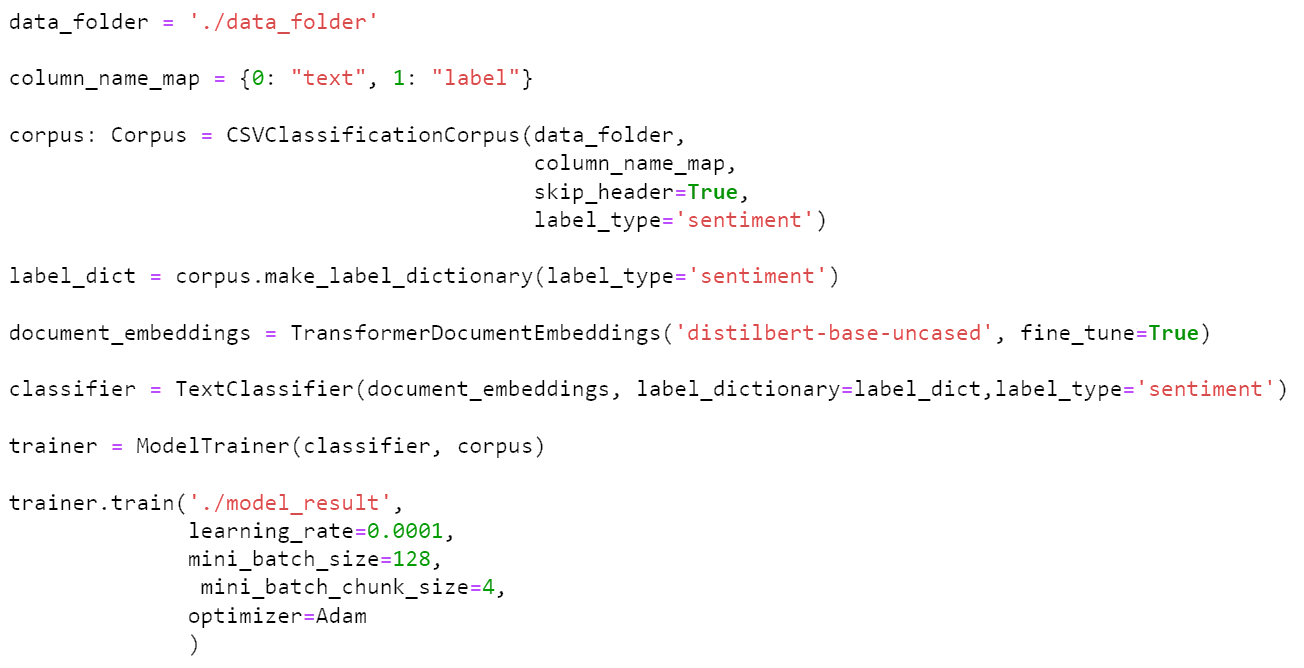
# Neural Network Model

We built two deep learning models: Flair and a two-layer shallow neural network.

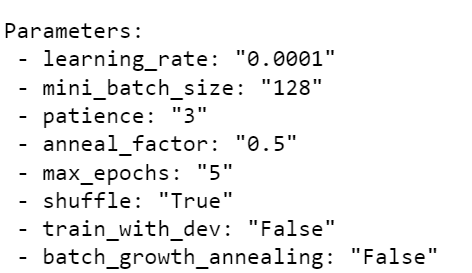
## Flair

### Structure of model

We imported the integrated functions of Flair like TextClassifier and ModelTrainer to train our dataset. Firstly, Loaded the folder where the train, test, and validation datasets are in, built a map to indicate which columns hold the text and labels, loaded the corpus containing training, test, and validation data by CSVClassificationCorpus. Then created the label dictionary by make\_label\_dictionary, then initialized the transformer document embeddings by using the function of TransformerDocumentEmbeddings. Besides, created the text classifier by Textclassifier, initialized the text classifier trainer with Adam optimizer, which is super practical in the classification assignment. Finally, trained the model with some parameters like a very small learning rate like 0.0001, batch size (128), epochs (5), and optimizer (Adam).

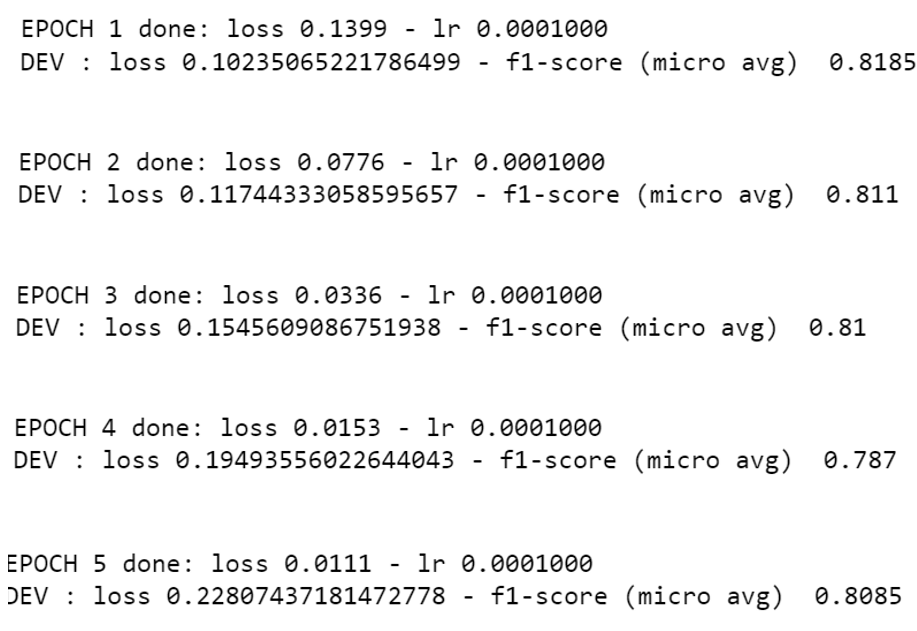


Picture 5- 1

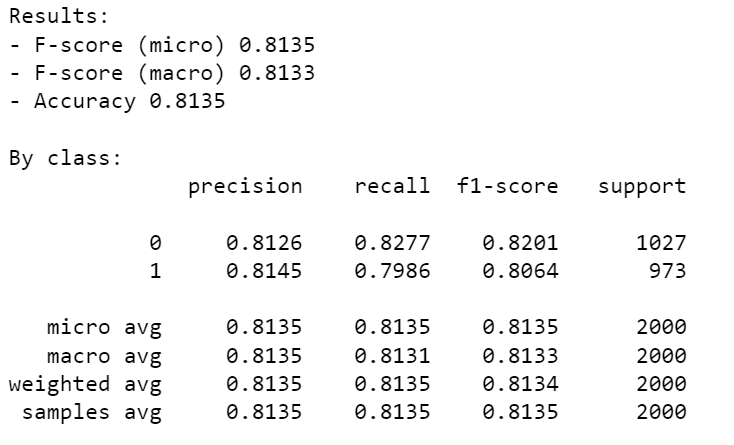


Picture 5- 2

### Training and Result



Picture 5- 3



Picture 5- 4



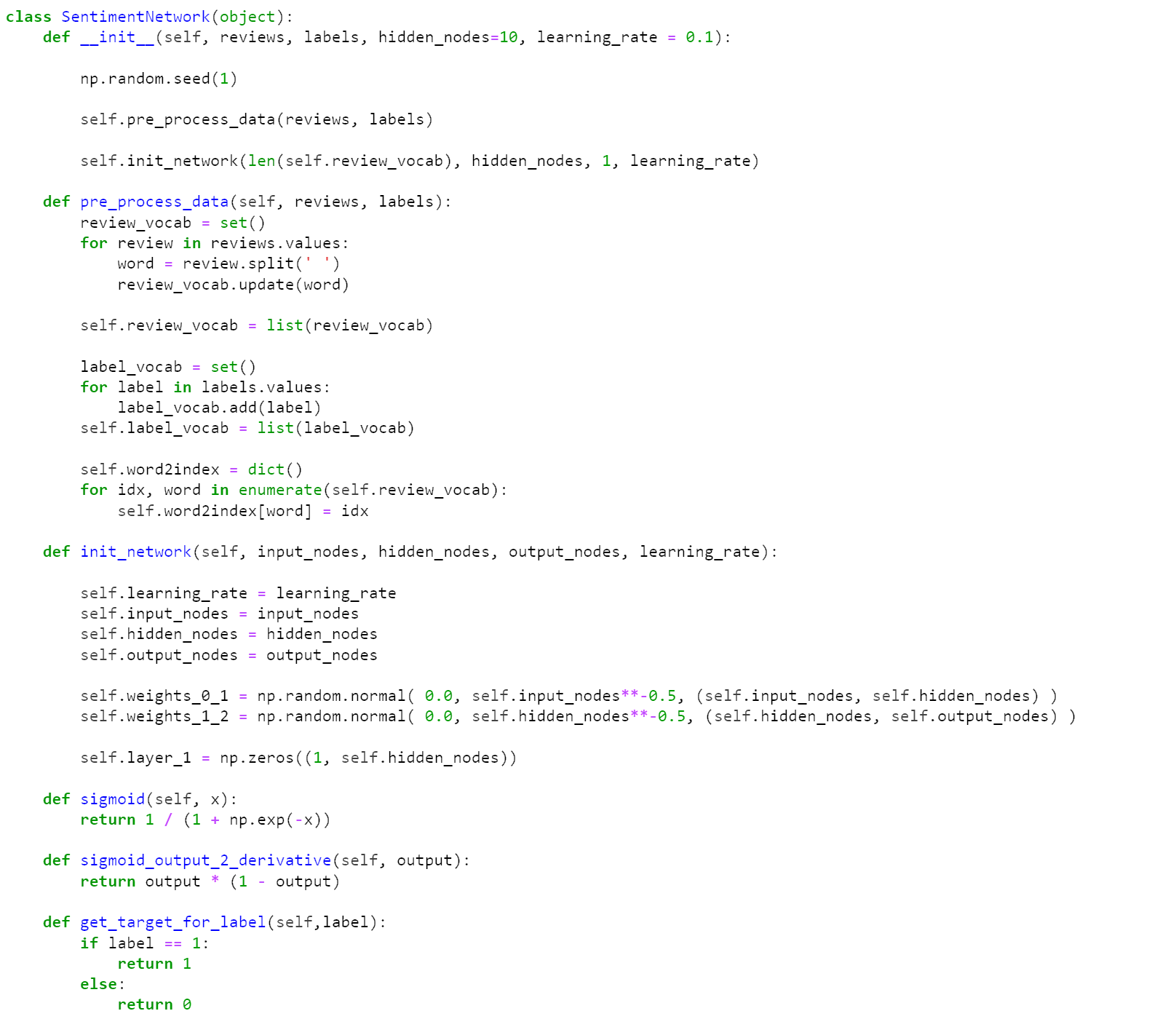
Picture 5- 5

We can see that the precision and F1-score are higher than any other models in this report, so we supposed that the Flair model is the most accurate。

## Two-layer shallow neural network

### Structure of network

We built a two-layer neural network to train the dataset. Firstly, preprocessing dataset by defining the function pre\_process\_function( ). This function is used for preprocessing data, counting all words appearing in reviews, generating word2index, counting all the words that appear in the dataset, counting all the values that appear in the target, building word2idx, and assigning a "house number" to each word. Then, initializing parameters of the network by function init\_network( )， like learning rate, number of input nodes, number of hidden nodes, number of output nodes.



Picture 5- 6



Picture 5- 7



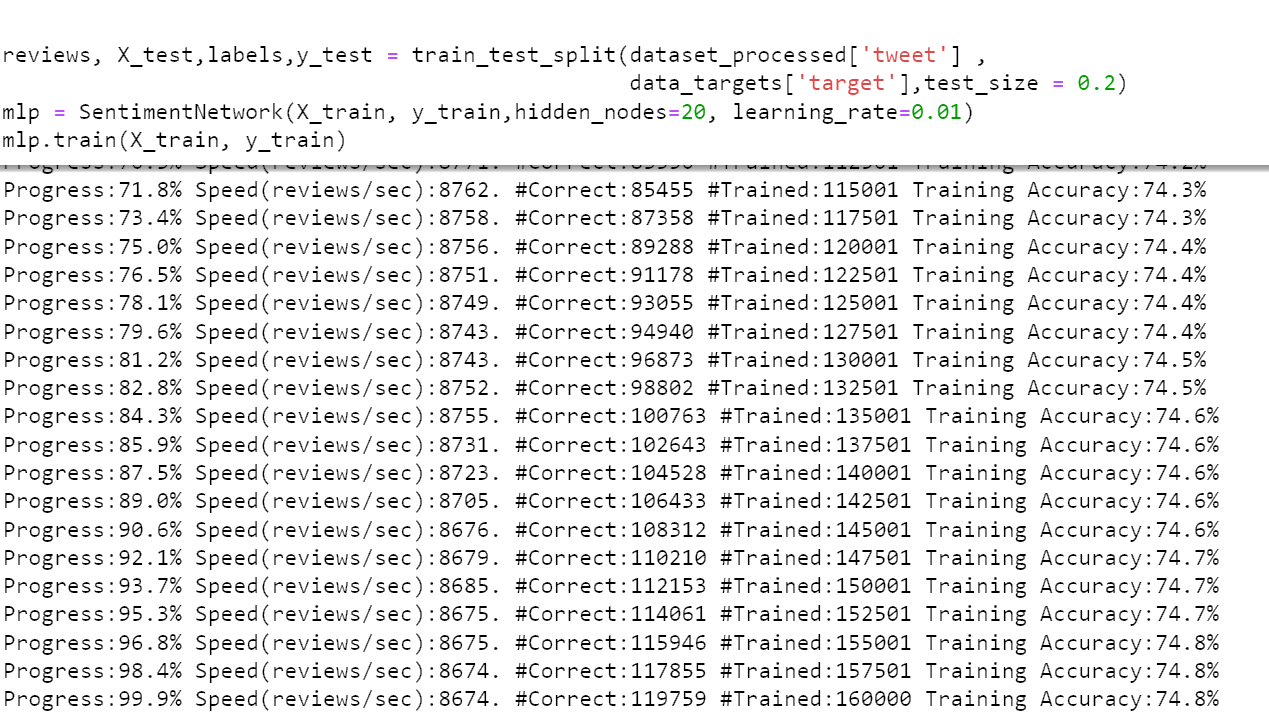
Picture 5- 8



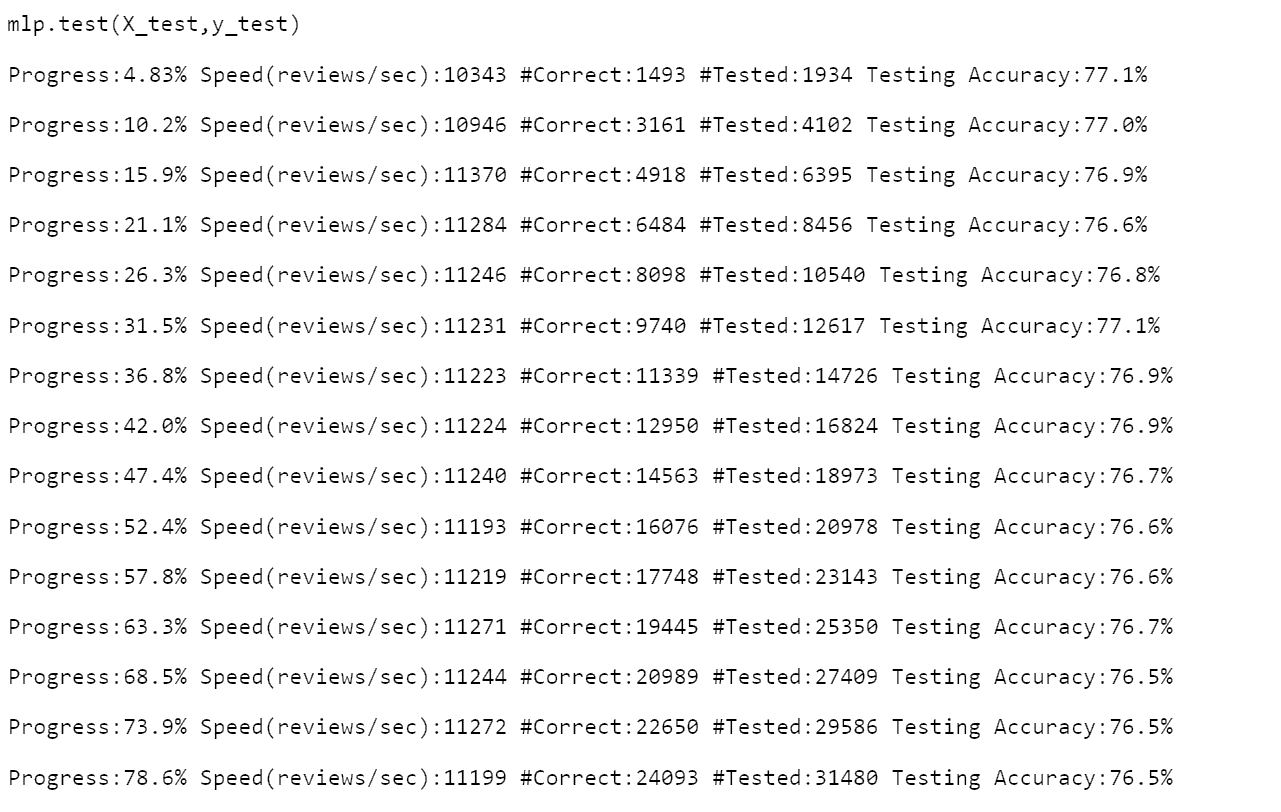
Picture 5- 9

### Training and Result

Then we split the train and test dataset with the proportions of 80% and 20%, trained the model with 20 hidden nodes and a 0.01 learning rate.



Picture 5- 10



Picture 5- 11

Finally, we got 74.8% accuracy of training and 76.5% of testing.

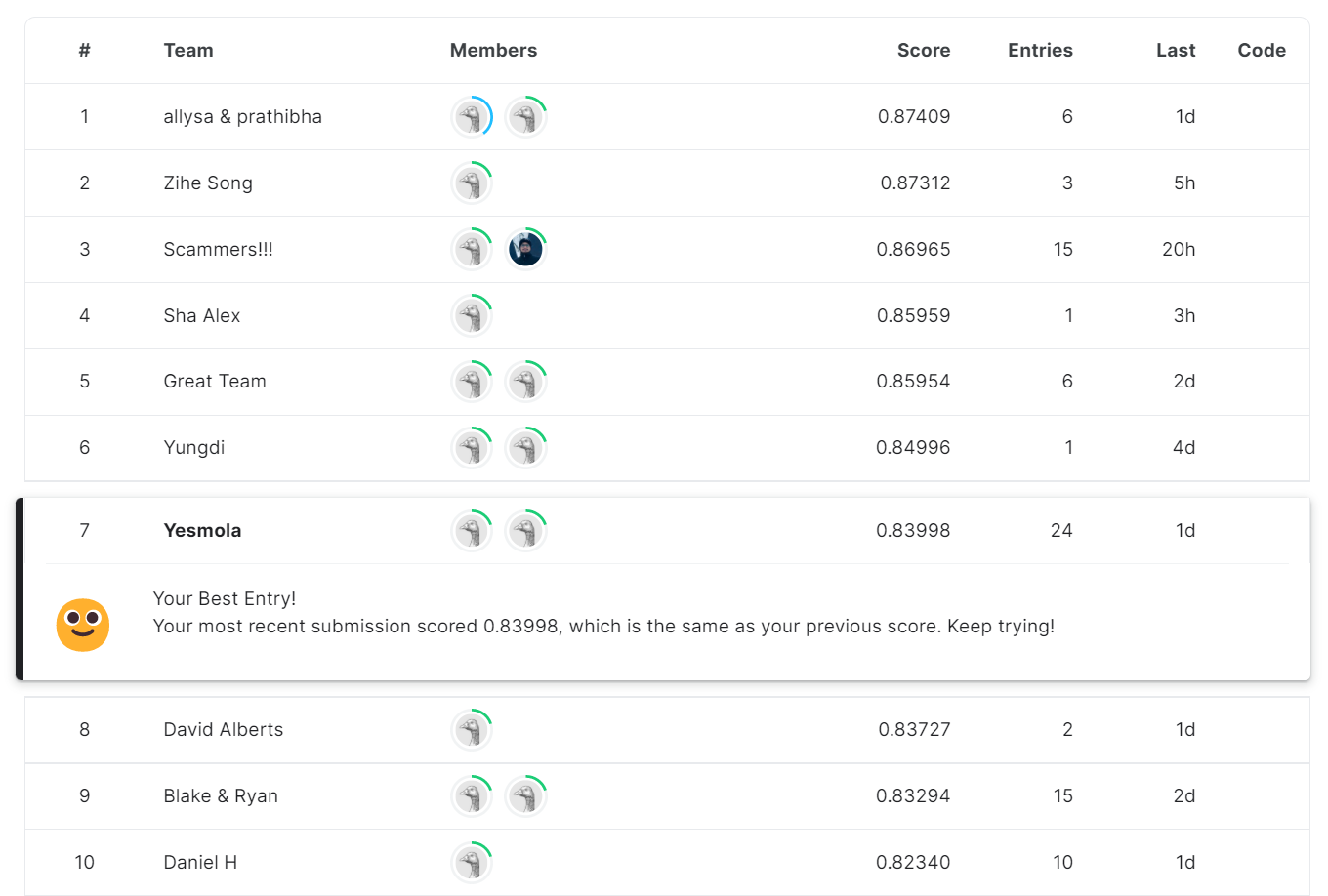
# Comparison of Models and Kaggle Result

Chart 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Precision | Recall | F1-score | Support |
| Bernoulli NB | 0.76 | 0.76 | 0.76 | 40000 |
| SVC | 0.78 | 0.78 | 0.78 | 40000 |
| Logistic | 0.78 | 0.78 | 0.78 | 40000 |
| Flair | 0.8135 | 0.8135 | 0.8135 | 40000 |
| Two-layer | 0.748 | 0.748 | 0.748 | 40000 |

The Flair model has the highest accuracy and F1score—both are 0.8135, followed by logistic regression and support vector machines, then the Bernoulli model, and finally the self-built two-layer neural network.

As for Kaggle, we got 7th place on the leaderboard for F1-score 0.83998.



Picture 5- 12

# Conclusion and contribution of each team member

We discussed how to process the dataset, specifically how to decrease the dimension of the attributes. We then created several functions to pre-process the data. Afterward, we built three supervising learning models including BNB, SVM, and LR models, and compared them to figure out which one is the best. Furthermore, we built two deep learning models like Flair and a two-layer shallow neural network and compared them. Finally, we compared all of the models and find that Flair had the highest accuracy.

In the mini-project, our team has two people: Xiangyu Gao and Rui Li. Xiangyu is mainly responsible for the model establishment, writing main codes, and parameter debugging, while Rui is mainly responsible for finding reference materials, correcting codes, and selecting model methods. Throughout the project, we found each other online and decided to team up together. During this time, we were coding face-to-face three times a week, because face-to-face communication allowed us to program more efficiently. Although we have our division of labor, the overall procedure is the result of both of us. We all feel very happy to work with each other.