Sentiment analysis is a form of exploratory data analysis based on natural language processing. It seeks to understand the underlying subjective tone and quantify the emotional intensity of words and phrases within a text.

The specifications of this implementation are as follows.

- Programming Language: Python 3.8.8 (in Anaconda3)
- Data set: IMDB Movie Reviews
    - including 25000 rows, 2 columns (Reviews, Sentiment)
    - This dataset is taken from https://ai.stanford.edu/~amaas/data/sentiment
- Machine Learning Tools and Libraries: SKLearn, NLTK, Numpy, Panda

Before we can do anything with the data, we should take some preprocessing steps to clean the data, reduce the noise and improve the accuracy of our classifier's results. Regarding the dataset used in this program, these steps include the following.

1. Removing punctuation
2. Removing stop words
3. Transforming words into lower case
4. Lemmatizing words: (to remove inflectional endings only and to return the base or dictionary form of a word)

Besides, in order to show changes at every preprocessing step, a new column named "new_reviews" was intentionally added to the dataset.

In the next step, we need to turn the text into a numerical representation for consumption by the classifier, which is called the Vectorization step. To this end, we used the TF-IDF technique to quantify words. In this step, we set the attribution *max_features to 5000* to consider only the top 5000 words ordered by term frequency across the corpus.

Afterward, we set the classifying parameters as follows:

- Allocating 20 percent of the dataset as the test data
- Using the Linear SVC method as a classifier

In the end, Table (1) indicates a concise classification report for this analysis.

	precision	recall	f1-score	support
neg	0.87	0.87	0.87	2480
pos	0.88	0.88	0.88	2520
accuracy			0.87	5000
macro avg	0.87	0.87	0.87	5000
weighted avg	0.87	0.87	0.87	5000

Table (1) : Classification Report

Now, the trained model for sentiment analysis is prepared. We can test it by feeding some inputs. For instance, if we pass the sentence "What an awesome movie! I recommend it.", it will return "pos" as a positive statement.

Instruction to use codes:

1. Make sure you have installed the following packages on your Python interpreter. Otherwise install them through your Python terminal (powershell.exe).
    - >>> pip3 install nltk
    - >>> import nltk
    - >>> nltk.download('all')
    - >>> nltk.download('wordnet')
    - >>> nltk.download('stopwords')
2. Open the file program.py in your python IDE and click on the Run icon.

A view of the program:

```
# A Basic Semantic Analysis
# Published by: Ehsan Bojnordi -- 11/11/2021
import pandas as pd
import numpy as np
import nltk
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report

############## Importing Dataset ###################
df = pd.read_excel('IMDB-Movie-Reviews-Large-Dataset-50k-master/train.xlsx')
df.head()

############## Removing Punctuation ###################
df['new_reviews'] = df['Reviews'].str.replace('[^\w\s]','', regex=True)
df.head()

############## Removing Stop Words ###################
stop = stopwords.words('english')
df['new_reviews'].apply(lambda x: [item for item in str(x).split() if item not in stop])
df.head(20)

############## Transforming words into lower case ###################
df['new_reviews'] = df['new_reviews'].str.lower().replace('\\', '').replace('_', ' ')
df.head()

############## Lemmatizing Words ###################
lmtzr = WordNetLemmatizer()
df['new_reviews'].apply(lambda lst:[lmtzr.lemmatize(word) for word in str(lst).split()])
df.head()
```