Importing Necessary Libraries

```
import spacy
from spacy import displacy
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.svm import LinearSVC
import string
```

Loading SpaCy's small english model

To get more details regarding SpaCy models check here : https://spacy.io/usage/models

```
# Loading Spacy small model as nlp
nlp = spacy.load("en_core_web_sm")
```

Gathering all the Stop words which does not convey much meaning in the Sentiment

+ Code + Text

```
# Gathering all the stopwords
from spacy.lang.en.stop_words import STOP_WORDS
stopwords = list(STOP_WORDS)
print(len(stopwords))
```

```
326
```

```
# Loading yelp dataset
data_yelp = pd.read_csv('amazon_cells_labelled (1).txt',
                        sep='\t', header= None)
data_yelp.head()
```

	0	1
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1

```
# Adding column names to the dataframe
columnName = ['Review','Sentiment']
data_yelp.columns = columnName
data_yelp.head()
```

	Review	Sentiment
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
3	Tied to charger for conversations lasting more...	0
4	The mic is great.	1

deduce that Sentiment 1 is Positive and 0 is negative

```
print(data_yelp.shape)
```

```
(1000, 2)
```

```
# Adding Amazon dataset and adding its column name
data_amz = pd.read_csv("amazon_cells_labelled (1).txt",

                       sep='\t', header= None)
data_amz.columns = columnName
data_amz.head()
```

BLACKBOX AI

	Review	Sentiment

```
print(data_amz.shape)
```

```
(1000, 2)
```

```
# Adding IMdB dataset and adding its column name
data_imdb = pd.read_csv("amazon_cells_labelled (1).txt",
                        sep='\t', header= None)
data_imdb.columns = columnName
data_imdb.head()
```

	Review	Sentiment
0	So there is no way for me to plug it in here i...	0
1	Good case, Excellent value.	1
2	Great for the jawbone.	1
3	Tied to charger for conversations lasting more...	0
4	The mic is great.	1

```
print(data_imdb.shape)
```

```
(1000, 2)
```

Appending all the Datasets

```
# Merging all the three dataframes
data = data_yelp.append([data_amz, data_imdb], ignore_index=True)
print(data.shape)
```

```
(3000, 2)
<ipython-input-48-8727f89e72a4>:2: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a future version. Use pandas.concat inst
  data = data_yelp.append([data_amz, data_imdb], ignore_index=True)
```

```
# Sentiment ditribution in the dataset
data.Sentiment.value_counts()
```

```
0    1500
1    1500
Name: Sentiment, dtype: int64
```

```
# Getting information regarding the null entries in the dataset
data.isnull().sum()
```

```
Review       0
Sentiment    0
dtype: int64
```

```
punct = string.punctuation
print(punct)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

Here in the reviews we will find many stop words which do not add any meaning to the review.
Also punctuations will be encountered in the review which which will be considered as a seperate token by our model
So removing all the stop words and punctuation so that our model can train efficiently

```
def dataCleaning(sentence):
  doc = nlp(sentence)
  tokens = []
  for token in doc:
    if token.lemma_ != '-PRON-':
      temp = token.lemma_.lower().strip()
    else:
      temp = token.lower_
    tokens.append(temp)
  clean_tokens = []
  for token in tokens:
    if token not in punct and token not in stopwords:
      clean_tokens.append(token)
  return clean_tokens
```

Here after passing a particular sentence in dataCleaning method we are returned with relevant words which contribute to the sentiments

BLACKBOX AI

```
dataCleaning("Today we are having heavy rainfall, We recommend you to stay at your home and be safe, Do not start running here and there")
# All the useful words are returned, no punctuations no stop words and in the lemmatized form
```

```
['today',
 'heavy',
 'rainfall',
 'recommend',
 'stay',
 'home',
 'safe',
 'start',
 'run']
```

```
# Spillting the train and test data
X = data['Review']
y = data['Sentiment']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
print(X_train.shape,y_test.shape)
```

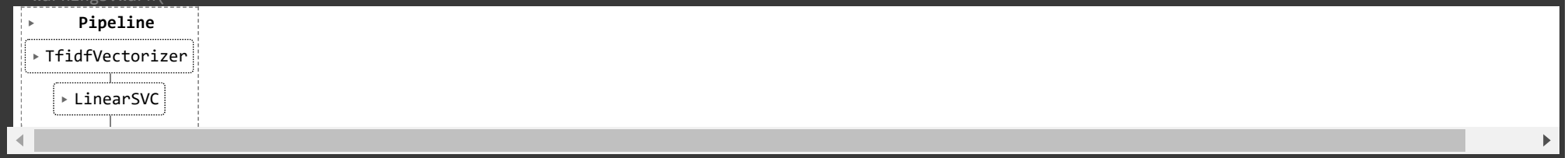
```
(2400,) (600,)
```

Preparing Model

```
# Creating the model and pipeline
tfidf = TfidfVectorizer(tokenizer = dataCleaning)
svm = LinearSVC()
steps = [('tfidf',tfidf),('svm',svm)]
pipe = Pipeline(steps)
```

```
# Training the model
pipe.fit(X_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is nc
  warnings.warn(
```




```
# Testing on the test dataset
y_pred = pipe.predict(X_test)
```

```
# Printing the classification report and the confusion matrix
print(classification_report(y_test,y_pred))
print("\n\n")
print(confusion_matrix(y_test,y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.96      0.94      0.95       277
               1       0.95      0.96      0.96       323

        accuracy                           0.95       600
       macro avg       0.95      0.95      0.95       600
    weighted avg       0.95      0.95      0.95       600




    [[260  17]
     [ 12 311]]
```

Testing on the Random Manual Examples

Here '1' represent that the input is positive sentiment

```
# Testing on random inputs
pipe.predict(["Wow you are an amazing person"])
```

```
    array([1])
```

Here '0' represent that input is negative sentiment

```
pipe.predict(["I am a boy."])
```

```
    array([1])
```

BLACKBOX AI

BLACKBOX AI