

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from sklearn.base import TransformerMixin
from sklearn.pipeline import Pipeline
```

```
from google.colab import files
uploaded = files.upload()
```

Choose Files

Text_Classifier.csv

- **Text_Classifier.csv**(text/csv) - 138 bytes, last modified: 3/7/2023 - 100% done

Saving Text_Classifier.csv to Text_Classifier.csv

```
data = pd.read_csv("Text_Classifier.csv")
print(data)
```

	Doc	Words					Author	
0	1	w1	w2	w3	w4	w5	C	
1	2		w1	w1	w4	w3	C	
2	3			w1	w2	w5	C	
3	4		w5	w6	w1	w2	w3	W
4	5				w4	w5	w6	W
5	6				w4	w6	w3	F
6	7	w2	w2	w4	w3	w5	w5	F

```
data.head()
```

	Doc	Words					Author
0	1	w1	w2	w3	w4	w5	C
1	2		w1	w1	w4	w3	C
2	3		w1	w2	w5		C
3	4	w5	w6	w1	w2	w3	W
4	5		w4	w5	w6		W

```
data.shape
```

(7, 3)

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 3 columns):
#   Column  Non-Null Count  Dtype
---  -
0    Doc      7 non-null        int64
1   Words     7 non-null        object
2   Author   7 non-null        object
dtypes: int64(1), object(2)
memory usage: 296.0+ bytes
```

```
data.Author.value_counts()
```

```
C    3
W    2
F    2
Name: Author, dtype: int64
```

```
import spacy
import string
from spacy.lang.en.stop_words import STOP_WORDS
from spacy.lang.en import English
```

```
# Create our list of punctuation marks
punctuations = string.punctuation
```

```
# Create our list of stopwords
nlp = spacy.load('en_core_web_sm')
stop_words = spacy.lang.en.stop_words.STOP_WORDS
```

```

# Load English tokenizer, tagger, parser, NER and word vectors
parser = English()

# Creating our tokenizer function
def spacy_tokenizer(sentence):
    # Creating our token object, which is used to create documents with linguistic annotations.
    mytokens = parser(sentence)

    # Lemmatizing each token and converting each token into lowercase
    mytokens = [ word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens ]

    # Removing stop words
    mytokens = [ word for word in mytokens if word not in stop_words and word not in punctuations ]

    # return preprocessed list of tokens
    return mytokens

/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
warnings.warn("Can't initialize NVML")

class predictors(TransformerMixin):
    def transform(self, X, **transform_params):
        # Cleaning Text
        return [clean_text(text) for text in X]

    def fit(self, X, y=None, **fit_params):
        return self

    def get_params(self, deep=True):
        return {}

# Basic function to clean the text
def clean_text(text):
    # Removing spaces and converting text into lowercase
    return text.strip().lower()

bow_vector = CountVectorizer(tokenizer = spacy_tokenizer, ngram_range=(1,1))
print(bow_vector)

CountVectorizer(tokenizer=<function spacy_tokenizer at 0x7f3390dae430>)

tfidf_vector = TfidfVectorizer(tokenizer = spacy_tokenizer)

from sklearn.model_selection import train_test_split

X = data['Words'] # the features we want to analyze
ylabels = data['Author'] # the labels, or answers, we want to test against
print(X)
print(ylabels)

X_train, X_test, y_train, y_test = train_test_split(X, ylabels, test_size=0.3)

0      w1 w2 w3 w4 w5
1      w1 w1 w4 w3
2      w1 w2 w5
3      w5 w6 w1 w2 w3
4      w4 w5 w6
5      w4 w6 w3
6      w2 w2 w4 w3 w5 w5
Name: Words, dtype: object
0      C
1      C
2      C
3      W
4      W
5      F
6      F
Name: Author, dtype: object

# Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()

# Create pipeline using HashingVectorizer
from sklearn.pipeline import Pipeline

```

```
from sklearn.feature_extraction.text import HashingVectorizer
```

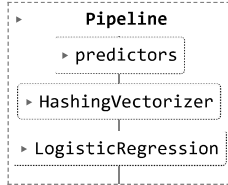
```
hash_vector = HashingVectorizer(tokenizer = spacy_tokenizer, ngram_range=(1,1))
```

```
pipe = Pipeline([("cleaner", predictors()),  
                 ('vectorizer', hash_vector),  
                 ('classifier', classifier)])
```

```
# model generation
```

```
pipe.fit(X_train,y_train)
```

```
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'tokenizers' is deprecated in favor of 'tokenizer'.
```



```
New_Value = ["w1 w4 w6 w5 w3"]
```

```
predicted1 = pipe.predict(New_Value) #New data  
print(predicted1)
```

```
['W']
```

```
#from sklearn import metrics
```

```
# Predicting with a test dataset
```

```
#predicted = pipe.predict(X_test)
```

```
# Model Accuracy
```

```
#print("Logistic Regression Accuracy:",metrics.accuracy_score(y_test, predicted))
```