12. Project: Who is the real author of Hamlet?

- Process
  - Step 1: Please implement a Text Classifier
    - Test the Text Classifier to predict who the real author of Hamlet is.

	Doc	Words	Author
Training	1	W1 W2 W3 W4 W5	C (Christopher Marlowe)
	2	W1 W1 W4 W3	C (Christopher Marlowe)
	3	W1 W2 W5	C (Christopher Marlowe)
	4	W5 W6 W1 W2 W3	W (William Stanley)
	5	W4 W5 W6	W (William Stanley)
	6	W4 W6 W3	F (Francis Bacon)
	7	W2 W2 W4 W3 W5 W5	F (Francis Bacon)
Test	8 (Hamlet)	W1 W4 W6 W5 W3	?

- Please clearly shows the results of
  - P(C)
  - P(W)
  - P(F)
  - P(W1|C)
  - P(W1|W)
  - P(W1|F)
  - P(W3|C)
  - P(W3|W)
  - P(W3|F)
  - P(W4|C)
  - P(W4|W)
  - P(W4|F)
  - P(W5|C)
  - P(W5|W)
  - P(W5|F)
  - P(W6|C)
  - P(W6|W)
  - P(W6|F)
  - P(C|d8)
  - P(W|d8)
  - P(F|d8)
  - Does d8 belong to C or W or F?
- Step 2: Update your portfolio by linking the Google Slides presentation to the GitHub.
  - Use the following structure

```
Machine Learning
     Text Classification
```

- Step 3: Please submit the URL of your GitHub postings as the homework answer.

Solution:

We have the training data and need to calculate the probabilities of each before testing the Text Classifier to identify the true author of Hamlet.

P(C): The probability of class C = 3/7

P(W): The probability of class W = 2/7

P(F): The probability of class F = 2/7

P(W1|C): The probability that the word "W1" appears on the 3 class C documents

= (count (W1, C) + 1) / (count(C)+|V|) = (4+1) / (12+6) = 5/18

4: how many times the word "W1" appear on the 3 class C documents.

12: how many words in the 3 class C documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W1|W) : The probability that the word "W1" appears on the 3 class W documents

= (count (W1, W) + 1) / (count(W)+|V|) = (1+1) / (8+6) = 2/14 = 1/7

1: how many times the word "W1" appear on the 2 class W documents.

8: how many words in the 3 class W documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W1|F): The probability that the word "W1" appears on the 2 class F documents

= (count(W1, F) + 1) / (count(F)+|V|) = (0+1) / (9+6) = 1/15

0: how many times the word "W1" appear on the 2 class F documents.

9: how many words in the 3 class W documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W3|C) : The probability that the word "W3" appears on the 3 class C documents

= (count(W3, C) + 1) / (count(C)+|V|) = (2+1) / (12+6) = 3/18 = 1/6

2: how many times the word "W3" appear on the 3 class C documents.

12: how many words in the 3 class C documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W3|W) : The probability that the word "W3" appears on the 3 class W documents

= (count(W3, W) + 1) / (count(W)+|V|) = (1+1) / (8+6) = 2/14 = 1/7

1: how many times the word "W3" appear on the 2 class W documents.

8: how many words in the 3 class W documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W3|F) : The probability that the word "W3" appears on the 2 class F documents

= (count(W3, F) + 1) / (count(F)+|V|) = (2+1) / (9+6) = 3/15 = 1/5

2: how many times the word "W3" appear on the 2 class F documents.

9: how many words in the 3 class F documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W4|C) : The probability that the word "W4" appears on the 3 class C documents

= (count(W4, C) + 1) / (count(C)+|V|) = (2+1) / (12+6) = 3/18 = 1/6

2: how many times the word "W4" appear on the 3 class C documents.

12: how many words in the 3 class C documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W4|W) : The probability that the word "W4" appears on the 3 class W documents

= (count(W4, W) + 1) / (count(W)+|V|) = (1+1) / (8+6) = 2/14 = 1/7

1: how many times the word "W4" appear on the 2 class W documents.

8: how many words in the 3 class W documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W4|F) : The probability that the word "W4" appears on the 2 class F documents

= (count(W4, F) + 1) / (count(F)+|V|) = (2+1) / (9+6) = 3/15

2: how many times the word "W4" appear on the 2 class F documents.

9: how many words in the 3 class F documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W5|C): The probability that the word "W5" appears on the 3 class C documents

= (count(W5, C) + 1) / (count(C)+|V|) = (2+1) / (12+6) = 3/18 = 1/6

2: how many times the word "W5" appear on the 3 class C documents.

12: how many words in the 3 class C documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W5|W): The probability that the word "W5" appears on the 3 class W documents

= (count(W5, W) + 1) / (count(W)+|V|) = (2+1) / (8+6) = 3/14

2: how many times the word "W5" appear on the 2 class W documents.

8: how many words in the 3 class W documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W5|F): The probability that the word "W5" appears on the 2 class F documents

= (count(W5, F) + 1) / (count(F)+|V|) = (2+1) / (9+6) = 3/15

2: how many times the word "W5" appear on the 2 class F documents.

9: how many words in the 3 class F documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W6|C): The probability that the word "W6" appears on the 3 class C documents

= (count(W6, C) + 1) / (count(C)+|V|) = (0+1) / (12+6) = 1/18

0: how many times the word "W6" appear on the 3 class C documents.

12: how many words in the 3 class C documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W6|W): The probability that the word "W6" appears on the 2 class W documents

= (count(W6, W) + 1) / (count(W)+|V|) = (2+1) / (8+6) = 3/14

2: how many times the word "W6" appear on the 2 class W documents.

8: how many words in the 3 class W documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(W6|F) : The probability that the word "W6" appears on the 2 class F documents

= (count(W6, F) + 1) / (count(F)+|V|) = (1+1) / (9+6) = 2/15

1: how many times the word "W6" appear on the 2 class F documents.

9: how many words in the 3 class F documents.

6: number of vocabularies: (W1 W2 W3 W4 W5 W6)

P(C|d8) : P(C) * P(W1|C) * P(W4|C)* P(W6|C) * P(W5|C) * P(W3|C)

= ((3/7) * (5/18) * (1/6) * (1/18) * (1/6) * (1/6)

= 0.00003061924 , approx. 0.00004

= 3/7: prior: P(C)

= There are 5 words in d8: W1 W4 W6 W5 W3

Each word "W1" has P(W1|C) = 5/18

The word "W4" has P(W4|C) =3/18 = 1/6

The word "W6" has P(W6|C) = 1/18

The word "W5" has P(W5|C) = 3/18 = 1/6

The word "W3" has P(W3|C) = 3/18 = 1/6

P(W|d8) = P(W) * P(W1|W) * P(W4|W)* P(W6|W) * P(W5|W) * P(W3|W)

= (2/7* 2/14 * 2/14 * 3/14 * 3/14 * 2/14)

= 0.00002824936, approx. 0.00003

= 2/7: prior: P(W)

= There are 5 words in d8: W1 W4 W6 W5 W3

Each word "W1" has P(W1|W) = 2/14

The word "W4" has P(W4|W) = 2/14

The word "W6" has P(W6|W) = 3/14

The word "W5" has P(W5|W) = 3/14

The word "W3" has P(W3|W) = 2/14

P(F|d8) = P(F) * P(W1|F) * P(W4|F)* P(W6|F) * P(W5|F) * P(W3|F)

= ((2/7) * (1/15) * (3/15) * (2/15) * (3/15) * (3/15))

= 0.00002031746, approx. 0.00002

= 2/7: prior: P(F)

= There are 5 words in d8: W1 W4 W6 W5 W3

Each word "W1" has P(W1|F) = 1/15

The word "W4" has P(W4|F) = 3/15

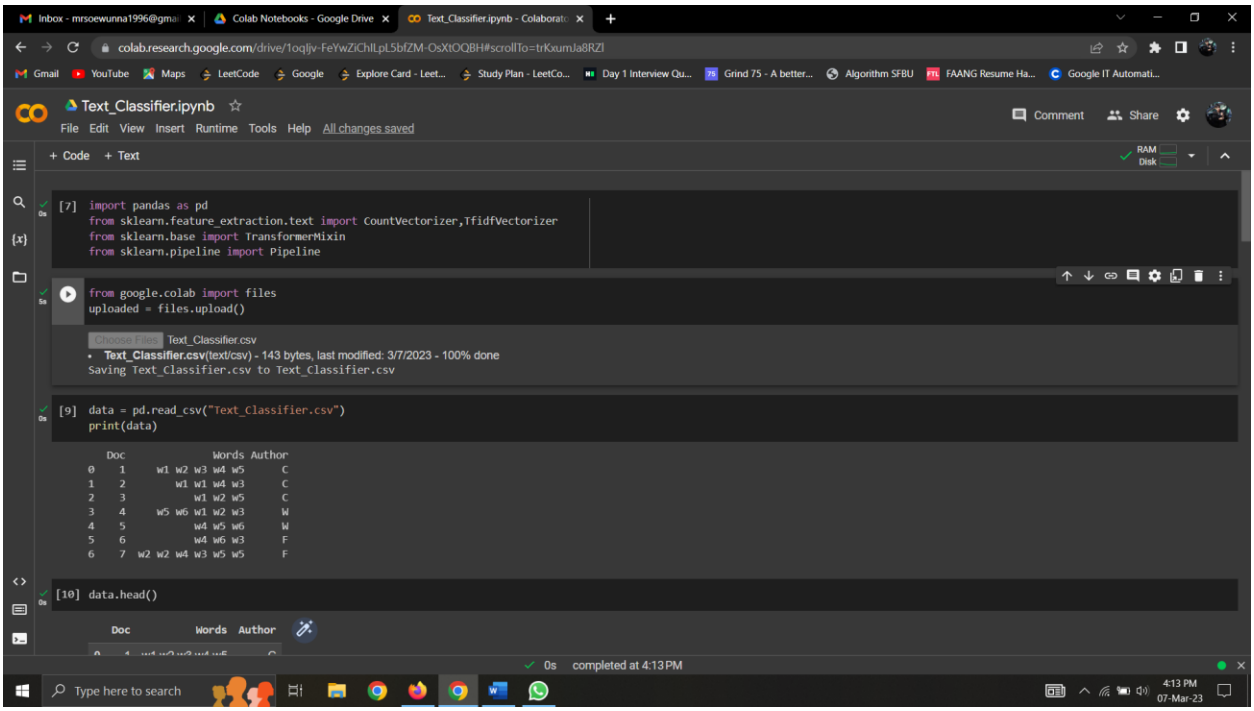
The word "W6" has P(W6|F) = 2/15

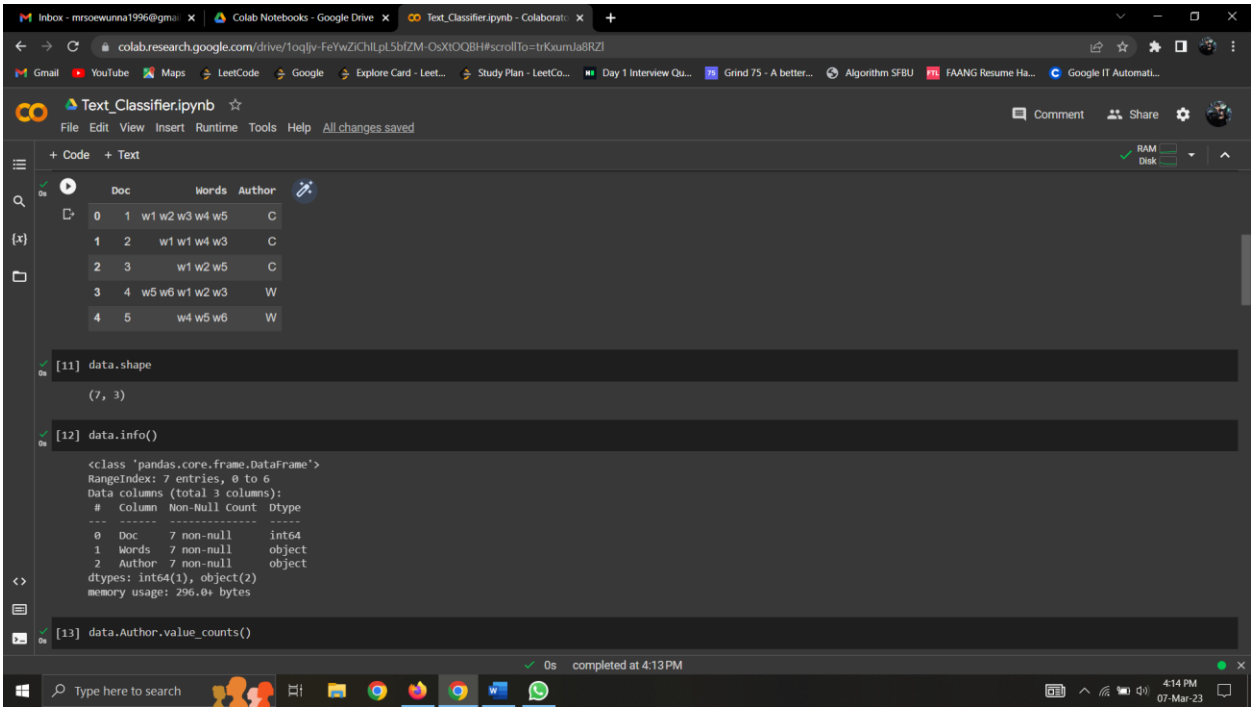
The word "W5" has P(W5|F) = 3/15

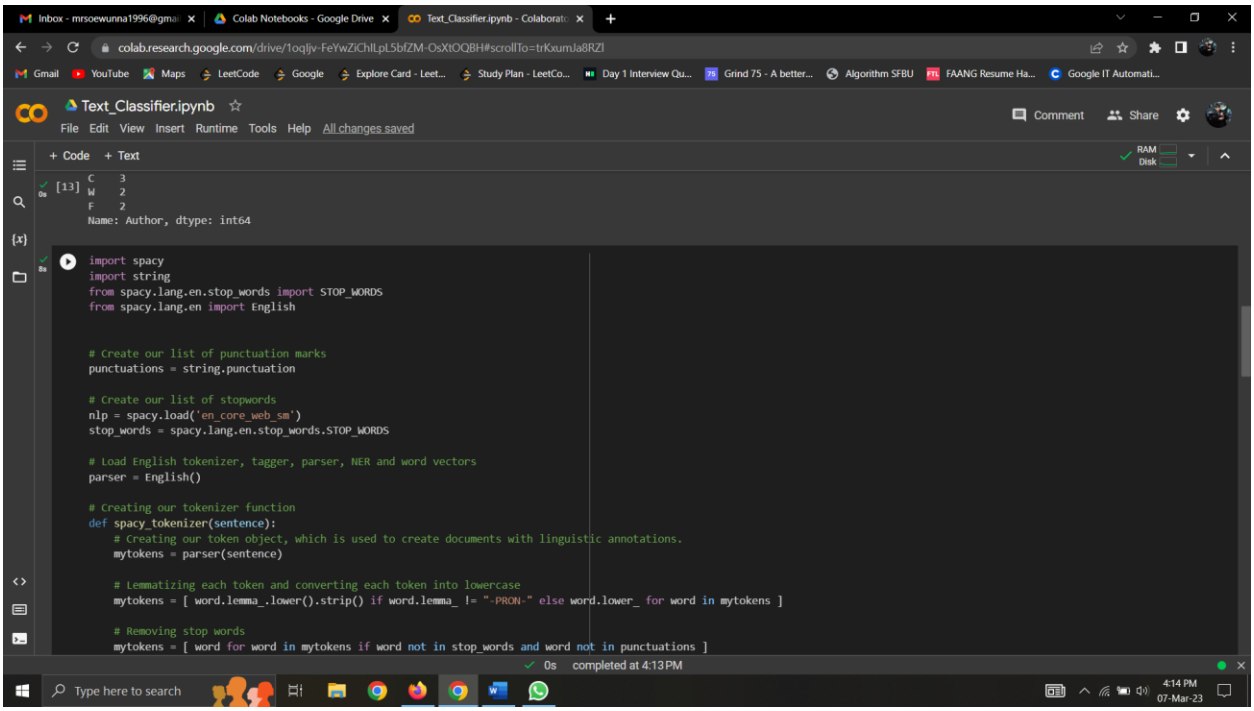
The word "W3" has P(W3|F) = 3/15

Does d8 belong to C or W or F?

The probability calculations show that Document 8 should be in Class C because it has the highest probability calculation.


Text_Classifier.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[7] 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) - 143 bytes, last modified: 3/7/2023 - 100% done
Saving Text_Classifier.csv to Text_Classifier.csv
[9] 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 F
[10] data.head()
Doc Words Author
0s completed at 4:13 PM
4:13 PM
07-Mar-23

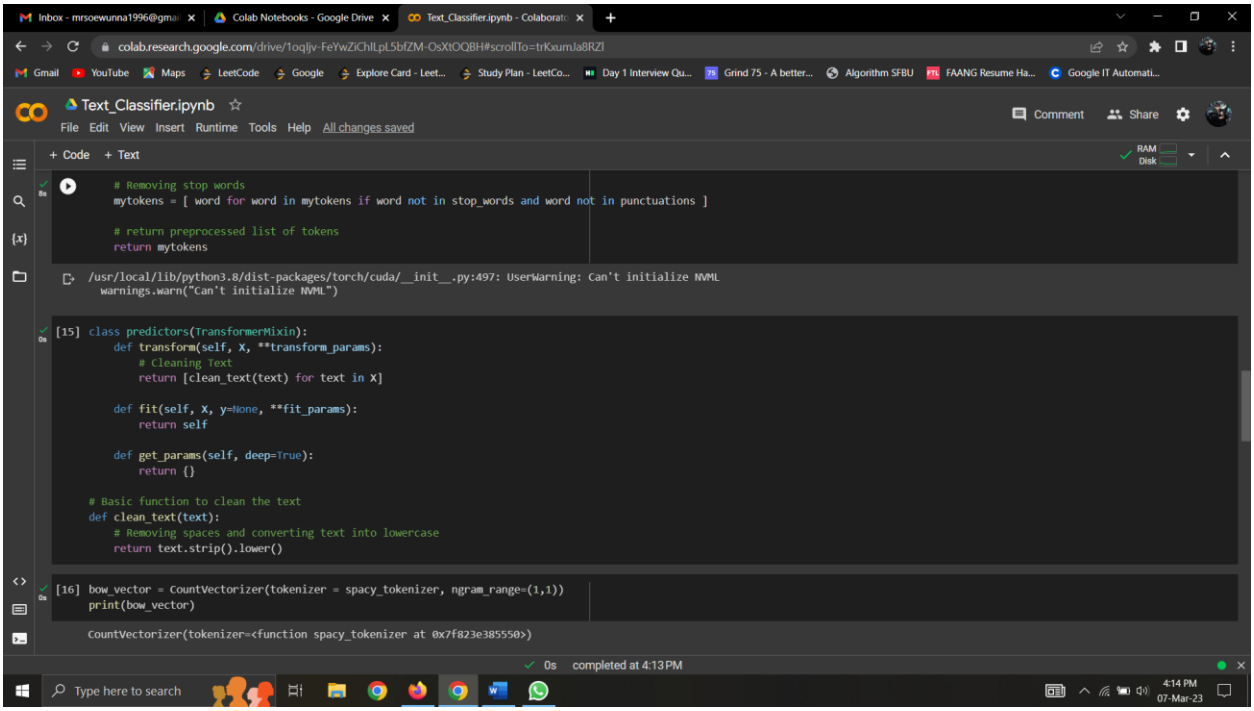

Text_Classifier.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Doc Words Author
0 1 w1 w2 w3 w4 w5 C
1 2 w1 w1 w4 w3 C
2 3 w1 w2 w5 C
3 4 w5 w6 w1 w2 w3 W
4 5 w4 w5 w6 W
[11] data.shape
(7, 3)
[12] data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 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
[13] data.Author.value_counts()
0s completed at 4:13 PM
4:14 PM
07-Mar-23



```
[13] 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 ]
```

0s completed at 4:13 PM



```
    # 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
```

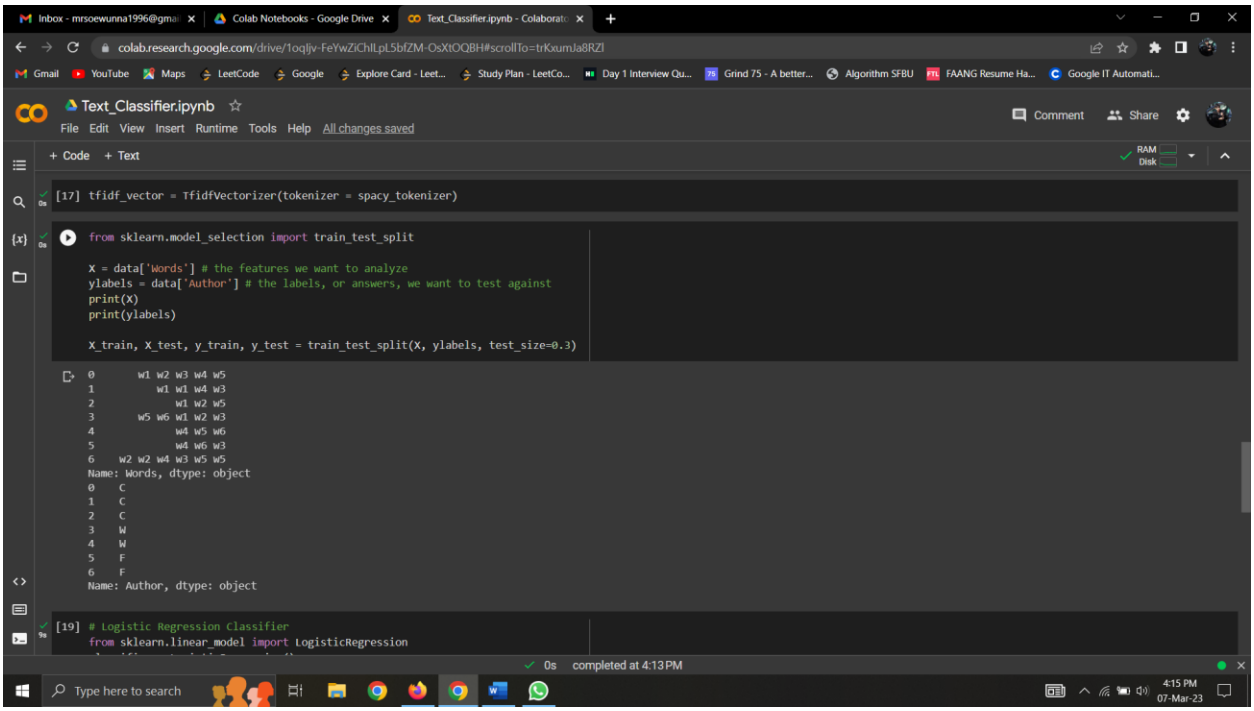
```
/usr/local/lib/python3.8/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
```

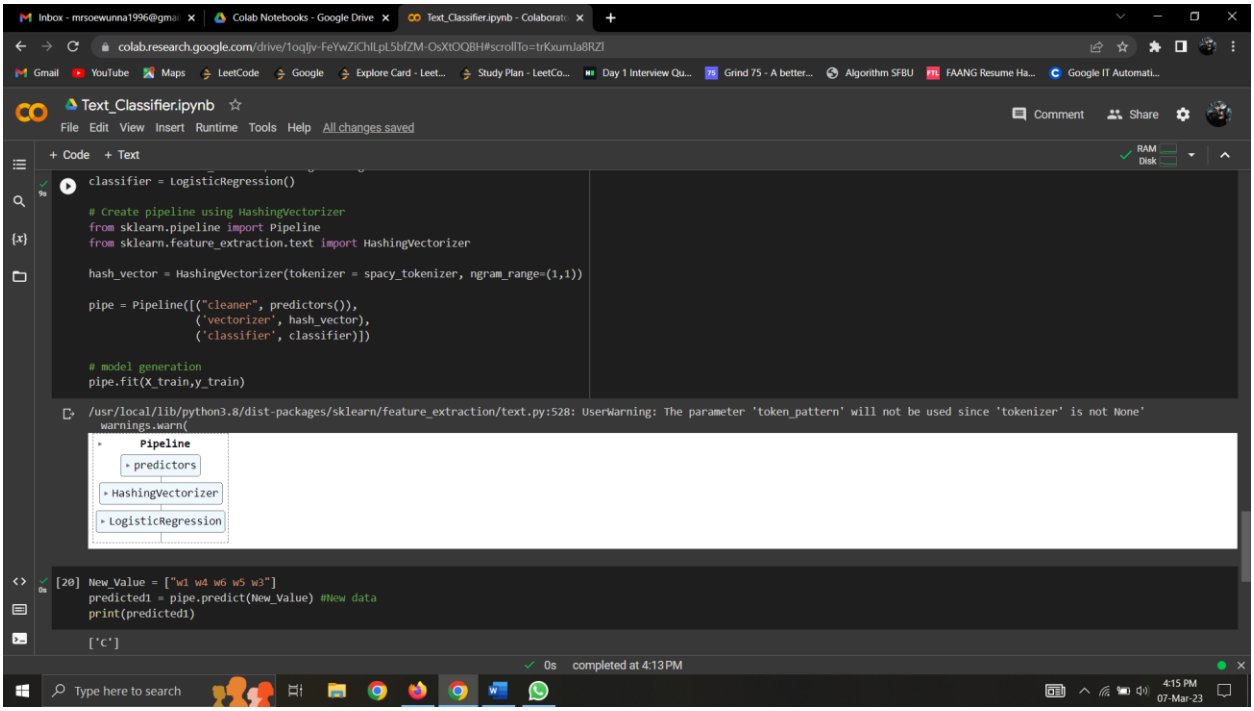
```
[15] class predictors(TransformerMixin):
        def transform(self, X, **transform_params):
            # Cleaning Text
            return [clean_text(text) for text in X]

        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()
```

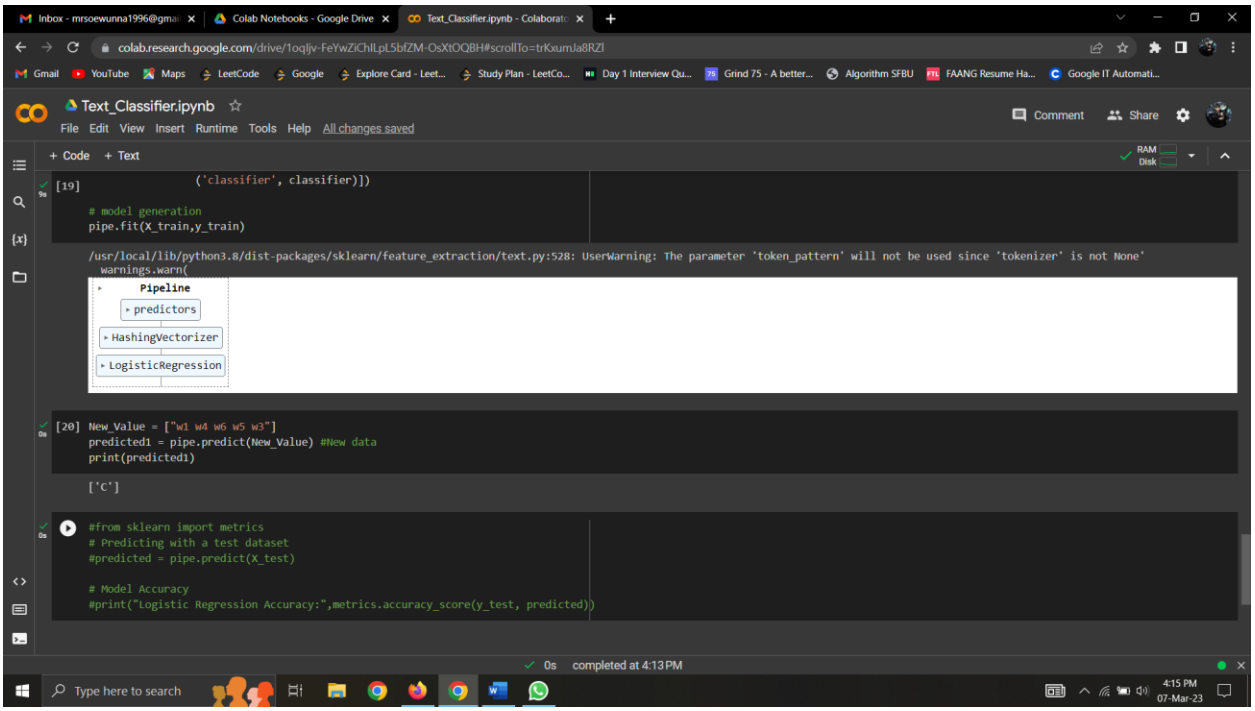
```
[16] bow_vector = CountVectorizer(tokenizer = spacy_tokenizer, ngram_range=(1,1))
     print(bow_vector)
```

```
CountVectorizer(tokenizer=<function spacy_tokenizer at 0x7f823e385550>)
```

0s completed at 4:13 PM


Text_Classifier.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[17] 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
Name: Words, dtype: object
0 C
1 C
2 C
3 W
4 W
5 F
6 F
Name: Author, dtype: object
[19] # Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
0s completed at 4:13 PM
Type here to search
4:15 PM
07-Mar-23


Text_Classifier.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
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 'token_pattern' will not be used since 'tokenizer' is not None'
warnings.warn(
Pipeline
predictors
HashingVectorizer
LogisticRegression
[20] New_Value = ["w1 w4 w6 w5 w3"]
predicted1 = pipe.predict(New_Value) #New data
print(predicted1)
['C']
0s completed at 4:13 PM
Type here to search
4:15 PM
07-Mar-23


Text_Classifier.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
[19] ('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 'token_pattern' will not be used since 'tokenizer' is not None'
warnings.warn(
Pipeline
predictors
HashingVectorizer
LogisticRegression
[20] New_Value = ["w1 w4 w6 w5 w3"]
predicted1 = pipe.predict(New_Value) #New data
print(predicted1)
['c']
#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))
0s completed at 4:13 PM
Type here to search
4:15 PM
07-Mar-23