

CS550 - Machine Learning and Business Intelligence

Machine Learning - Text Classification

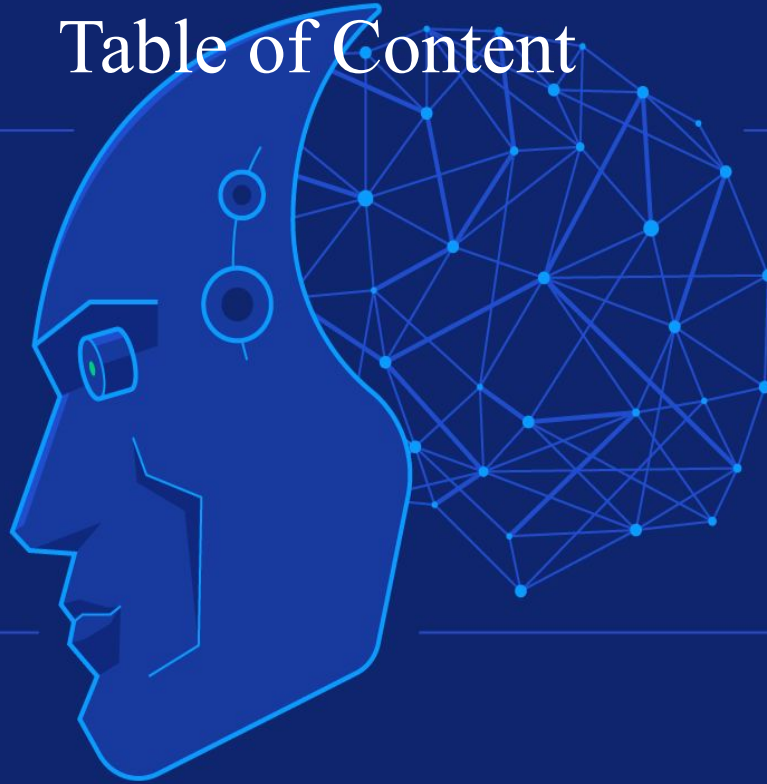
By Soe Wunna (19651)

Instructor: Dr. Chang, Henry



Table of Content

1. Introduction
2. Types of Text Classifiers
3. Manual Calculation
4. Implementation of Code
5. Conclusion
6. References



1. Introduction

A text classifier is a type of machine learning model that learns to classify text data into different categories or classes. The goal of a text classifier is to automatically analyze and understand the content of text data and classify it into predefined categories based on its content.

Text classifiers are widely used in many applications such as sentiment analysis, spam detection, topic categorization, and language identification. They work by using algorithms that analyze the features of the text data, such as the frequency of certain words, the presence of specific patterns or keywords, and the context in which the text appears. The model then uses this information to make predictions about the category or class that the text belongs to.

2. Types of Text Classifiers

There are various types of text classifiers, including rule-based classifiers, Naive Bayes classifiers, decision tree classifiers, support vector machines (SVM), and neural network classifiers. Each type has its own strengths and weaknesses, and the choice of classifier depends on the specific task and the nature of the text data being analyzed.

Overall, text classifiers are a powerful tool for automatically analyzing and categorizing large volumes of text data, which can help businesses and organizations make better decisions and improve their operations.

3. Manual Calculation

	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	?

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.

3. Manual Calculation

$P(C)$: 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
= $(\text{count}(W1, C) + 1) / (\text{count}(C) + |V|) = (4+1) / (12+6) = 5/18$

$P(W1|W)$: The probability that the word "W1" appears on the 3 class W documents
= $(\text{count}(W1, W) + 1) / (\text{count}(W) + |V|) = (1+1) / (8+6) = 2/14 = 1/7$

$P(W1|F)$: The probability that the word "W1" appears on the 2 class F documents
= $(\text{count}(W1, F) + 1) / (\text{count}(F) + |V|) = (0+1) / (9+6) = 1/15$

$P(W3|C)$: The probability that the word "W3" appears on the 3 class C documents
= $(\text{count}(W3, C) + 1) / (\text{count}(C) + |V|) = (2+1) / (12+6) = 3/18 = 1/6$

3. Manual Calculation

$P(W3|W)$: The probability that the word "W3" appears on the 3 class W documents
= $(\text{count}(W3, W) + 1) / (\text{count}(W) + |V|) = (1+1) / (8+6) = 2/14 = 1/7$

$P(W3|F)$: The probability that the word "W3" appears on the 2 class F documents
= $(\text{count}(W3, F) + 1) / (\text{count}(F) + |V|) = (2+1) / (9+6) = 3/15 = 1/5$

$P(W4|C)$: The probability that the word "W4" appears on the 3 class C documents
= $(\text{count}(W4, C) + 1) / (\text{count}(C) + |V|) = (2+1) / (12+6) = 3/18 = 1/6$

$P(W4|W)$: The probability that the word "W4" appears on the 3 class W documents
= $(\text{count}(W4, W) + 1) / (\text{count}(W) + |V|) = (1+1) / (8+6) = 2/14 = 1/7$

$P(W4|F)$: The probability that the word "W4" appears on the 2 class F documents
= $(\text{count}(W4, F) + 1) / (\text{count}(F) + |V|) = (2+1) / (9+6) = 3/15$

$P(W5|C)$: The probability that the word "W5" appears on the 3 class C documents
= $(\text{count}(W5, C) + 1) / (\text{count}(C) + |V|) = (2+1) / (12+6) = 3/18 = 1/6$

3. Manual Calculation

$P(W5|F)$: The probability that the word "W5" appears on the 2 class F documents
= $(\text{count}(W5, F) + 1) / (\text{count}(F) + |V|) = (2+1) / (9+6) = 3/15$

$P(W6|C)$: The probability that the word "W6" appears on the 3 class C documents
= $(\text{count}(W6, C) + 1) / (\text{count}(C) + |V|) = (0+1) / (12+6) = 1/18$

$P(W6|W)$: The probability that the word "W6" appears on the 2 class W documents
= $(\text{count}(W6, W) + 1) / (\text{count}(W) + |V|) = (2+1) / (8+6) = 3/14$

$P(W6|F)$: The probability that the word "W6" appears on the 2 class F documents
= $(\text{count}(W6, F) + 1) / (\text{count}(F) + |V|) = (1+1) / (9+6) = 2/15$

3. Manual Calculation

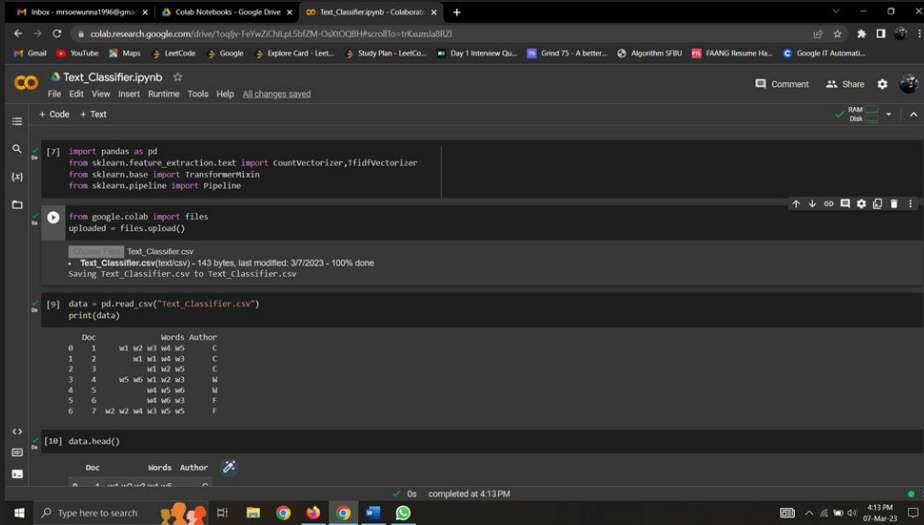
$$\begin{aligned} P(C|d8) &: P(C) * P(W1|C) * P(W4|C) * P(W6|C) * P(W5|C) * P(W3|C) \\ &= ((3/7) * (5/18) * (1/6) * (1/18) * (1/6) * (1/6)) \\ &= 0.00003061924, \text{ approx. } 0.00004 \end{aligned}$$

$$\begin{aligned} P(W|d8) &= P(W) * P(W1|W) * P(W4|W) * P(W6|W) * P(W5|W) * P(W3|W) \\ &= (2/7 * 2/14 * 2/14 * 3/14 * 3/14 * 2/14) \\ &= 0.00002824936, \text{ approx. } 0.00003 \end{aligned}$$

$$\begin{aligned} P(F|d8) &= P(F) * P(W1|F) * P(W4|F) * P(W6|F) * P(W5|F) * P(W3|F) \\ &= ((2/7) * (1/15) * (3/15) * (2/15) * (3/15) * (3/15)) \\ &= 0.00002031746, \text{ approx. } 0.00002 \end{aligned}$$

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

4. Implementation of Code



```
[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()

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

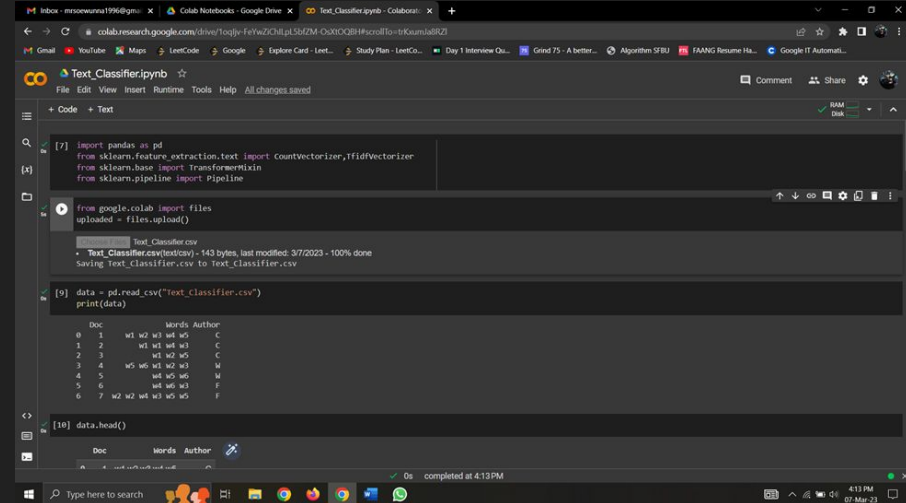
[9] data = pd.read_csv("Text_Classifier.csv")
print(data)
```

	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 w1 w3 w5	F

```
[10] 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
5	6	w4 w6 w3	F
6	7	w2 w2 w1 w3 w5	F

completed at 4:13 PM



```
[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()

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)
```

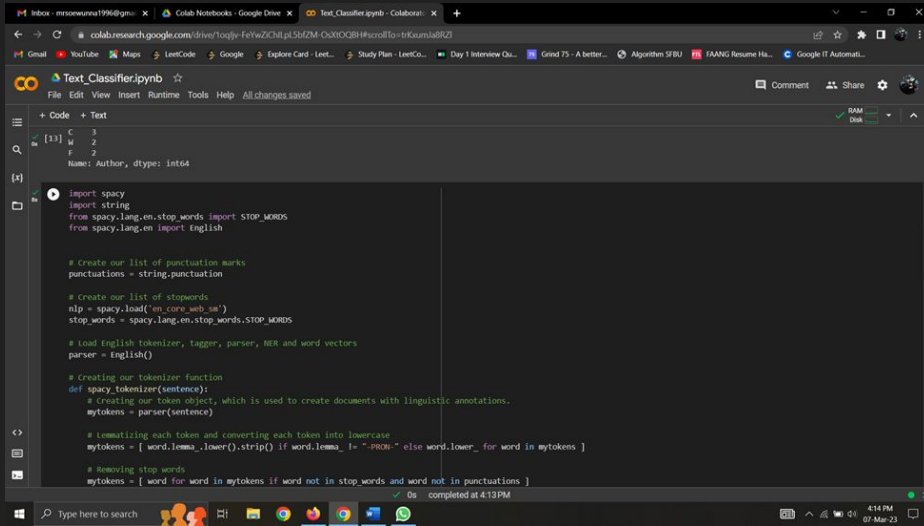
	Doc	Words	Author
0	1	w1 w2 w3 w4 w5	C
1	2	w1 w1 w4 w3	C
2	3	w1 w2 w5	C
3	4	w5 w6 w1 w2 w3	W
4	5	w4 w5 w6	W
5	6	w4 w6 w3	F
6	7	w2 w2 w1 w3 w5	F

```
[10] 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
5	6	w4 w6 w3	F
6	7	w2 w2 w1 w3 w5	F

completed at 4:13 PM

4. Implementation of Code



```
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

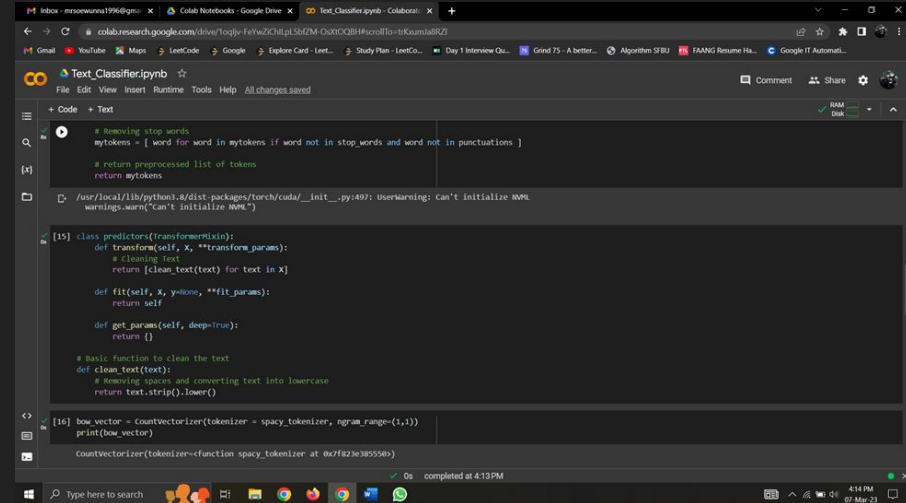
# Load English tokenizer, tagger, parser, NER and word vectors
parser = English()

# 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 ]

    # 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 NLP
warnings.warn("Can't initialize NLP")

[15] class predictors(LinearSVC):
    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 {}

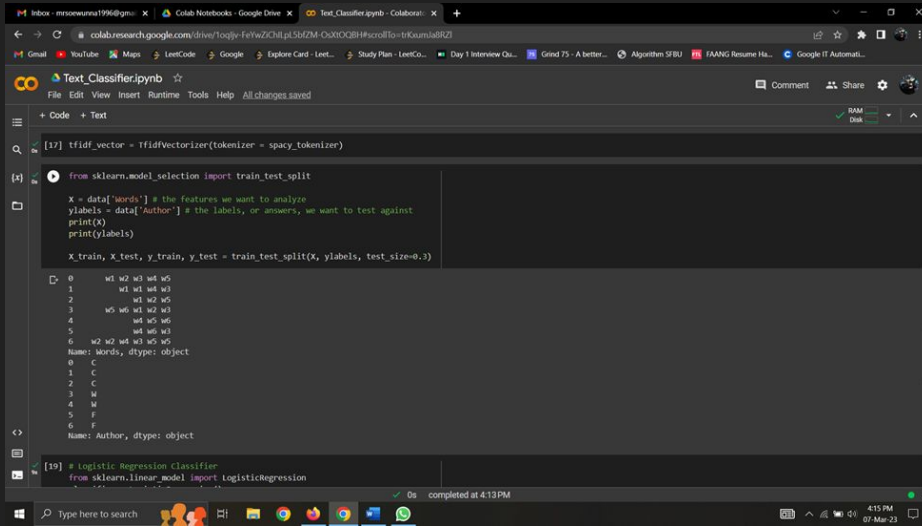
    # Basic function to clean the text
    def clean_text(text):
        # Removing spaces and converting text into lowercase
        return text.strip().lower()

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

CountVecorizer(tokenizer=function spacy_tokenizer at 0x7f823e38559b)

# completed at 4:13 PM
```

4. Implementation of Code



```
Inbox - mrsowunna1996@gmail.com x Colab Notebooks - Google Drive x Text_Classifier.ipynb - Colaboratory x +
colab.research.google.com/drive/1ogpy-felwZCHdpl56ZM-OxIXOBH#scrollTo=tKumjaBRZl
Gmail YouTube Maps LeetCode Google Explore Card - Leet... Study Plan - LeetCo... Day 1 Interview Ques... Grind 75 - A better... Algorithms SFBU FAANG Resume Ha... Google IT Automati...

Text_Classifier.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[17] tfidf_vector = TfidfVectorizer(tokenizer = spacy_tokenizer)

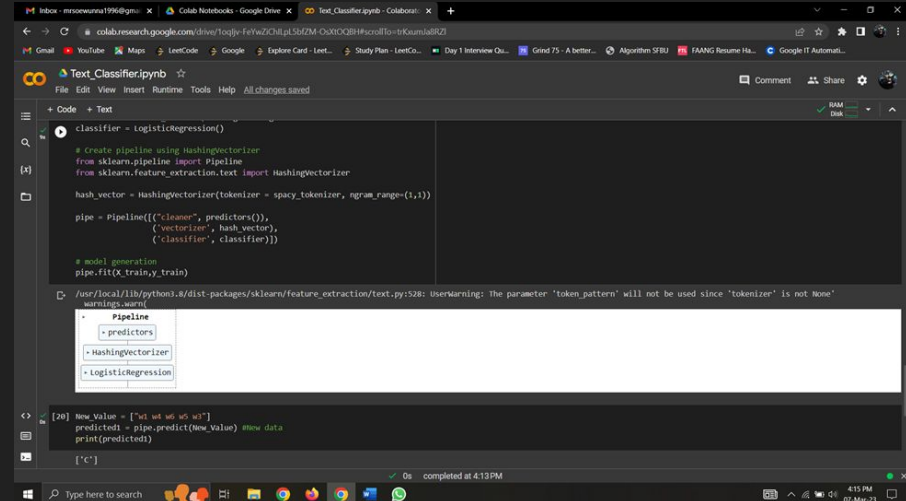
[18] from sklearn.model_selection import train_test_split

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

X_train, X_test, y_train, y_test = train_test_split(X, y_labels, 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 w1 w5 w6
5 w1 w6 w3
6 w2 w2 w1 w3 w5
Name: words, dtype: object
0 C
1 C
2 C
3 M
4 M
5 F
6 F
Name: Author, dtype: object

[19] # Logistic Regression Classifier
from sklearn.linear_model import LogisticRegression
```



```
Inbox - mrsowunna1996@gmail.com x Colab Notebooks - Google Drive x Text_Classifier.ipynb - Colaboratory x +
colab.research.google.com/drive/1ogpy-felwZCHdpl56ZM-OxIXOBH#scrollTo=tKumjaBRZl
Gmail YouTube Maps LeetCode Google Explore Card - Leet... Study Plan - LeetCo... Day 1 Interview Ques... Grind 75 - A better... Algorithms SFBU FAANG Resume Ha... Google IT Automati...

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)

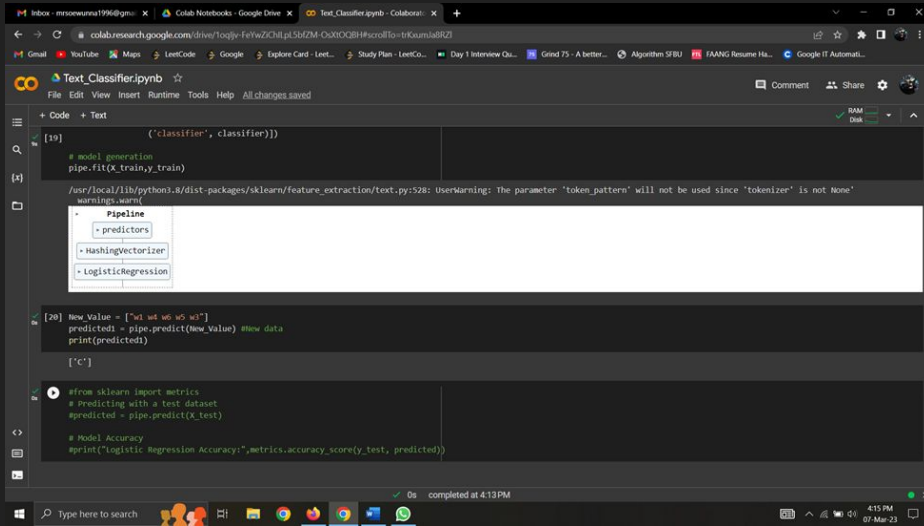
/usr/local/lib/python3.8/dist-packages/sklearn/feature_extraction/text.py:528: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is not None
warnings.warn(

Pipeline
- predictors
- HashingVectorizer
- LogisticRegression

[20] New Value = ["w1 w2 w3 w4 w5"]
predicted1 = pipe.predict(New_Value) #show data
print(predicted1)

['C']
```

4. Implementation of Code



```
[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 w2 w3 w4 w5"]
predicted = pipe.predict(new_value) #new data
print(predicted)

['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))
```

completed at 4:13 PM

Final Result is **Class 'C'**.

5. Conclusion

In conclusion, text classifiers are an essential component of modern machine learning systems. They enable the automatic analysis and categorization of large volumes of text data, which can be used for various applications such as sentiment analysis, spam detection, topic categorization, and language identification.

Text classifiers have become increasingly important in today's data-driven world, where organizations generate and collect vast amounts of text data. With the help of text classifiers, businesses can extract valuable insights from this data and make better-informed decisions.



6. References

1. https://hc.labnet.sfbu.edu/~henry/sfbu/course/mllib/naive_bayes/slide/text_classifier.html
2. <https://levity.ai/blog/text-classifiers-in-machine-learning-a-practical-guide>
3. <https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>