

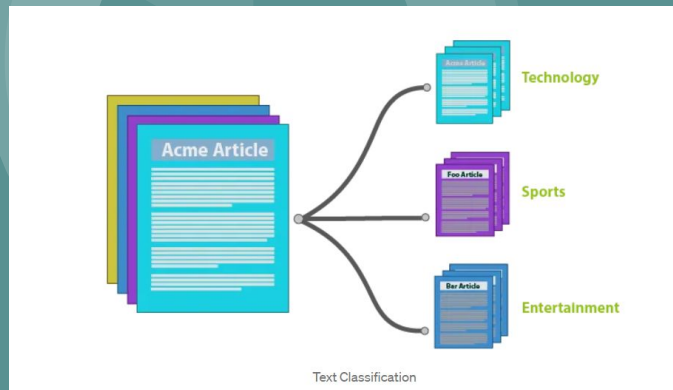
# CS550 - Machine Learning and Business Intelligence

## Text Classification

By: Maranata Muluneh

Instructor: Dr. Chang, Henry

2023





# Table of Content

- Introduction
- Theory
- Implementation
- Conclusion
- References



# Introduction

**Text classification**, also known as text categorization, is a subfield of machine learning that involves classifying text documents into predefined categories based on their content. The goal of text classification is to automatically assign the appropriate category or label to a given text document, based on its topic, sentiment, author, or other relevant characteristics.



# Introduction

**Text classification has many real-world applications**, such as spam filtering, sentiment analysis, news categorization, product review analysis, and content recommendation. By automating the process of categorizing large volumes of text, text classification can save time and resources, enable better decision-making, and improve the user experience.



# Theory

## Text Classification: Definition

- **Input:**

A document  $d$

A fixed set of classes  $C = \{c_1, c_2, \dots, c_j\}$

- **Output:** a predicted class  $c$  is element of  $C$



# Theory

## Classification Methods: Hand coded rules

- Rules based on combinations of words or other features

Spam: black- list address OR (“dollars” and “have been selected”)

- Accuracy can be high

If rules carefully refined by expert

- But building and maintaining these rules is expensive



# Theory

## Classification Methods: Supervised Machine Learning

- **Input:**

A document  $d$

A fixed set of classes  $C = \{c_1, c_2, \dots, c_j\}$

A training set of  $m$  hand-labeled documents  $(d_1, c_1) \dots (d_m, c_m)$

- **Output:** a learned classifier  $y: d \Rightarrow c$



# Theory

## Bayes Rule Applied to Documents and Classes

- For a document  $d$  and a class  $c$

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$



# Implementation

**Project: Who is the real author of Hamlet?**

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





# Implementation

Project: Who is the real author of Hamlet?

	Doc	Words	Author
<b>Training</b>	1	W1 W2 W3 W4 W5	C ( <a href="#">Christopher Marlowe</a> )
	2	W1 W1 W4 W3	C ( <a href="#">Christopher Marlowe</a> )
	3	W1 W2 W5	C ( <a href="#">Christopher Marlowe</a> )
	4	W5 W6 W1 W2 W3	W ( <a href="#">William Stanley</a> )
	5	W4 W5 W6	W ( <a href="#">William Stanley</a> )
	6	W4 W6 W3	F ( <a href="#">Francis Bacon</a> )
	7	W2 W2 W4 W3 W5 W5	F ( <a href="#">Francis Bacon</a> )
<b>Test</b>	8 (Hamlet)	W1 W4 W6 W5 W3	?



# Implementation

- We can do the prediction of the above question in two ways:

Manually applying the compare model

Using Google Colab



## Implementation (Manually)

- **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**

$$= (\text{count}(W1, C) + \underline{1}) / (\text{count}(C) + |V|)$$

$$= (4+1) / (12+6) = 5/18$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W1, W) + \underline{1}) / (\text{count}(W) + |V|)$$

$$= (1+1) / (8+6) = 2/14 = 1/7$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W1, F) + \underline{1}) / (\text{count}(F) + |V|)$$

$$= (0+1) / (9+6) = 1/15$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W3, C) + \underline{1}) / (\text{count}(C) + |V|)$$

$$= (2+1) / (12+6) = 3/18 = 1/6$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W3, W) + \underline{1}) / (\text{count}(W) + |V|)$$

$$= (1+1) / (8+6) = 2/14 = 1/7$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)





## Implementation (Manually)

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

$$= (\text{count}(W3, F) + \underline{1}) / (\text{count}(F) + |V|)$$

$$= (2+1) / (9+6) = 3/15 = 1/5$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W4, C) + \underline{1}) / (\text{count}(C) + |V|)$$

$$= (2+1) / (12+6) = 3/18 = 1/6$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W4, W) + \underline{1}) / (\text{count}(W) + |V|)$$

$$= (1+1) / (8+6) = 2/14 = 1/7$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W4, F) + \underline{1}) / (\text{count}(F) + |V|)$$

$$= (2+1) / (9+6) = 3/15$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(\text{W5}, \text{C}) + \underline{1}) / (\text{count}(\text{C}) + |\text{V}|)$$

$$= (2+1) / (12+6) = 3/18 = 1/6$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W5, W) + \underline{1}) / (\text{count}(W) + |V|)$$

$$= (2+1) / (8+6) = 3/14$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W5, F) + \underline{1}) / (\text{count}(F) + |V|)$$

$$= (2+1) / (9+6) = 3/15$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W6, C) + \underline{1}) / (\text{count}(C) + |V|)$$

$$= (0+1) / (12+6) = 1/18$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)





## Implementation (Manually)

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

$$= (\text{count}(W6, W) + \underline{1}) / (\text{count}(W) + |V|)$$

$$= (2+1) / (8+6) = 3/14$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

$$= (\text{count}(W6, F) + \underline{1}) / (\text{count}(F) + |V|)$$

$$= (1+1) / (9+6) = 2/15$$

- 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 vocabulary: (W1 W2 W3 W4 W5 W6)



## Implementation (Manually)

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

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

$$= 0.00003061924, \text{ approx } 0.00003$$

$$= 3/7: \text{ prior : } P(C)$$

$$= \text{There are 5 words in d8 : } W1 \ W4 \ W6 \ W5 \ W3$$

- 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  $P(W3|C) = 3/18 = 1/6$



## Implementation (Manually)

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

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

$$= 0.00003824936, \text{ approx } 0.00004$$

$$= 2/7: \text{ prior : } P(W)$$

$$= \text{There are 5 words in d8 : } W1 \ W4 \ W6 \ W5 \ W3$$

- 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  $P(W3|W) = 2/14$



## Implementation (Manually)

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

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

$$= 0.00002031746, \text{ approx } 0.00002$$

$$= 2/7 : \text{prior} : P(F)$$

$$= \text{There are 5 words in } d8 : W1 \ W4 \ W6 \ W5 \ W3$$

- 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  $P(W3|F) = 3/15$



# Implementation (Manually)

**Does d8 belong to C or W or F?**

According to the numbers from the probability calculations, **Document 8** should belong to **class W** Since it has the highest probability calculation.

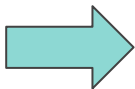


# Implementation (Colab)

Environment : Colab, Tensorflow 2

Programming Language: Python

Libraries

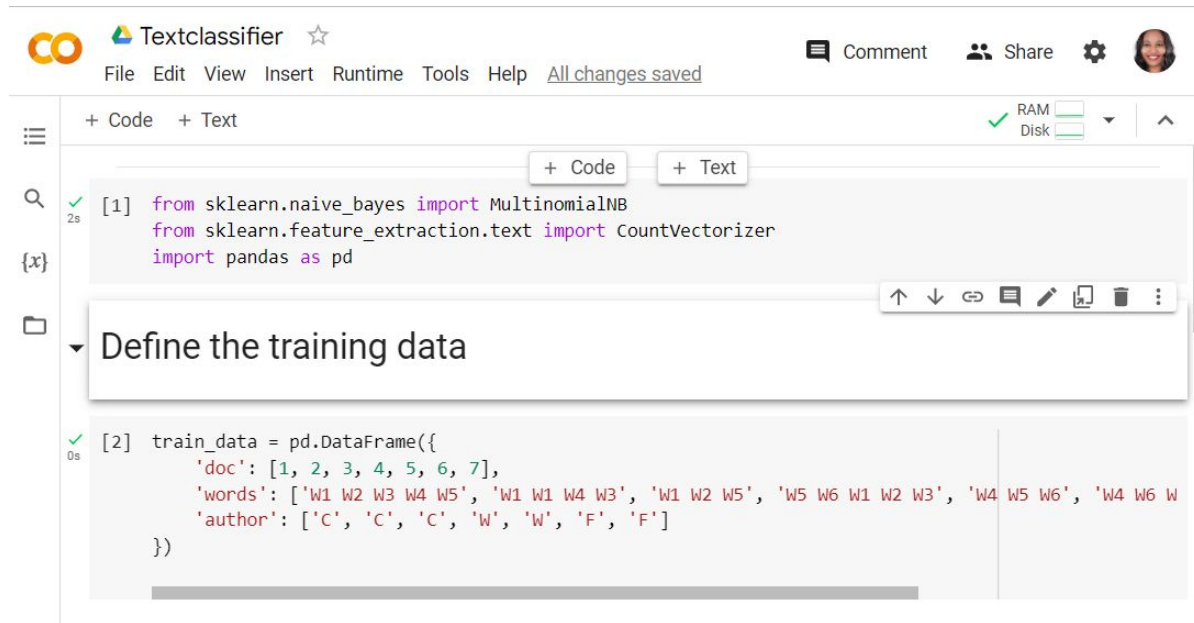


```
from sklearn.naive_bayes import  
MultinomialNB  
from sklearn.feature_extraction.text import  
CountVectorizer  
import pandas as pd
```

# Implementation

- Here's implementation of the text classifier using Colab:

## Step 1: Define the training data



The screenshot shows a Google Colab notebook interface. At the top, the title is "Textclassifier" with a star icon. Below the title is a menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". To the right of the menu bar are icons for "Comment", "Share", and a user profile. Below the menu bar, there are tabs for "+ Code" and "+ Text". The notebook contains two code cells. The first cell, labeled "[1]", imports the following modules: `from sklearn.naive_bayes import MultinomialNB`, `from sklearn.feature_extraction.text import CountVectorizer`, and `import pandas as pd`. The second cell, labeled "[2]", defines a pandas DataFrame named `train_data` with the following data: `train_data = pd.DataFrame({'doc': [1, 2, 3, 4, 5, 6, 7], 'words': ['W1 W2 W3 W4 W5', 'W1 W1 W4 W3', 'W1 W2 W5', 'W5 W6 W1 W2 W3', 'W4 W5 W6', 'W4 W6 W', 'author': ['C', 'C', 'C', 'W', 'W', 'F', 'F']})`. The notebook interface also shows a left sidebar with icons for search, variables, and file explorer. On the right side, there are status indicators for RAM and Disk usage.

```
[1] from sklearn.naive_bayes import MultinomialNB
    from sklearn.feature_extraction.text import CountVectorizer
    import pandas as pd
```

Define the training data

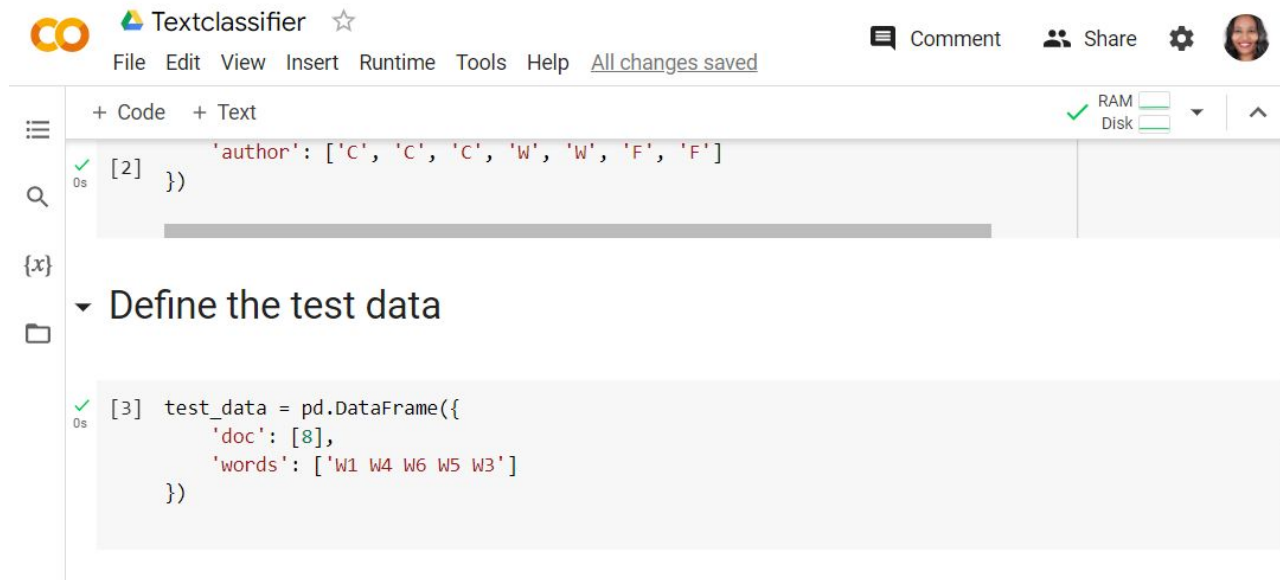
```
[2] train_data = pd.DataFrame({
    'doc': [1, 2, 3, 4, 5, 6, 7],
    'words': ['W1 W2 W3 W4 W5', 'W1 W1 W4 W3', 'W1 W2 W5', 'W5 W6 W1 W2 W3', 'W4 W5 W6', 'W4 W6 W',
    'author': ['C', 'C', 'C', 'W', 'W', 'F', 'F']
})
```



# Implementation

- Here's implementation of the text classifier using Colab:

## Step 2: Define the test data



The screenshot shows a Google Colab notebook interface. At the top, the title bar says "Textclassifier" with a star icon. Below it are menu items: "File", "Edit", "View", "Insert", "Runtime", "Tools", "Help", and a link "All changes saved". On the right side of the title bar are icons for "Comment", "Share", a settings gear, and a user profile picture. Below the title bar, there are tabs for "+ Code" and "+ Text". The main area of the notebook shows two code cells. The first cell, labeled "[2]", contains the following Python code: 

```
'author': ['C', 'C', 'C', 'W', 'W', 'F', 'F']
```

. The second cell, labeled "[3]", contains the following Python code: 

```
test_data = pd.DataFrame({  
    'doc': [8],  
    'words': ['W1 W4 W6 W5 W3']  
})
```

. On the left side of the notebook, there is a sidebar with icons for a menu, search, a variable {x}, and a folder.

```
'author': ['C', 'C', 'C', 'W', 'W', 'F', 'F']
```

### Define the test data

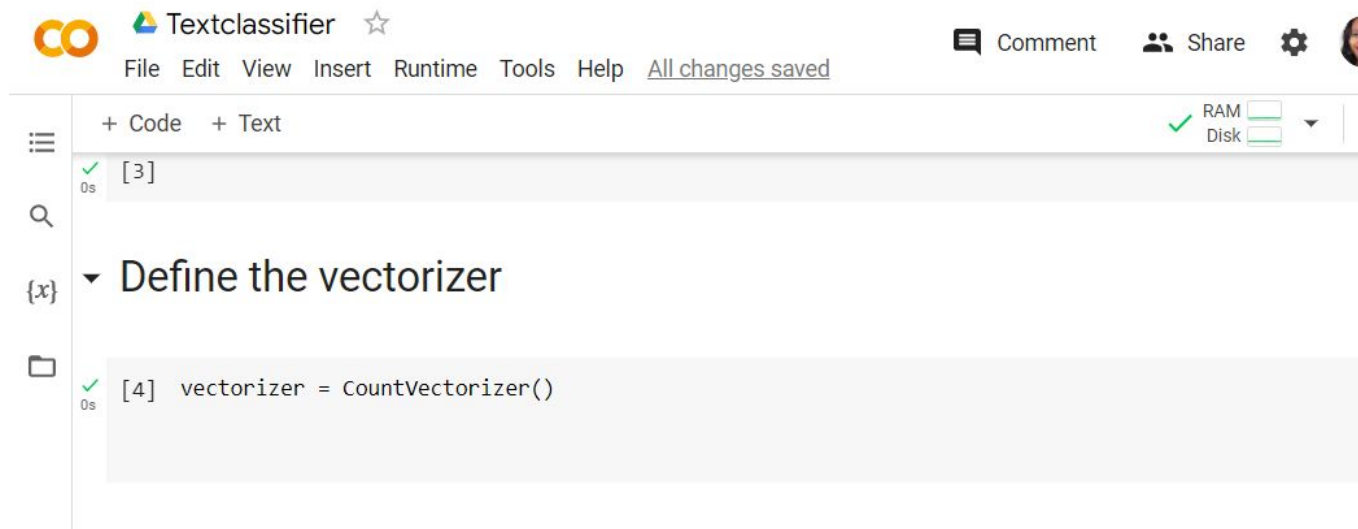
```
test_data = pd.DataFrame({  
    'doc': [8],  
    'words': ['W1 W4 W6 W5 W3']  
})
```



# Implementation

- Here's implementation of the text classifier using Colab:

## Step 3: Define the Vectorizer



# Implementation

- Here's implementation of the text classifier using Colab:

Step 4: Transform the training data

Step 5: Train the classifier

Step 6: Transform the test data

```
Textclassifier ☆
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM ✓ Disk ✓

▼ Transform the training data
[5] X_train = vectorizer.fit_transform(train_data['words'])
    y_train = train_data['author']

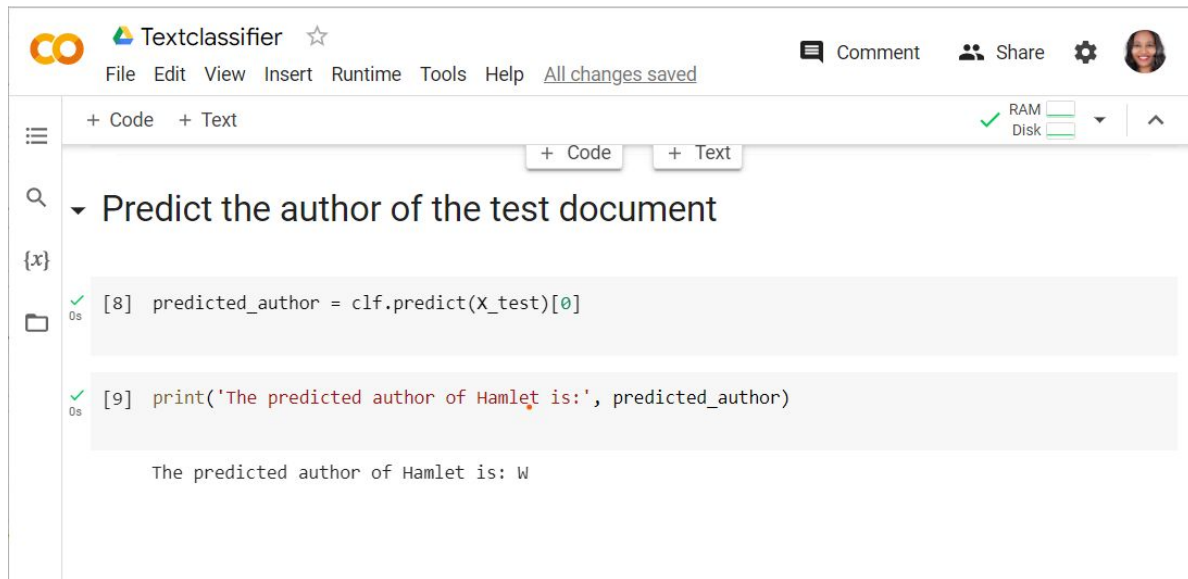
▼ Train the classifier
[6] clf = MultinomialNB()
    clf.fit(X_train, y_train)
    ▼ MultinomialNB
    MultinomialNB()

▼ Transform the test data
[7] X_test = vectorizer.transform(test_data['words'])
```

# Implementation

- Here's implementation of the text classifier using Colab:

Step 7: Predict the author of the test document



Textclassifier ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

+ Code + Text

▼ Predict the author of the test document

```
[8] predicted_author = clf.predict(X_test)[0]
```

```
[9] print('The predicted author of Hamlet is:', predicted_author)
```

The predicted author of Hamlet is: W



## Conclusion

Text classification is a powerful technique that allows us to automatically categorize and classify text data. It has a wide range of applications, including sentiment analysis, spam filtering, and topic modeling.

Overall, text classification is a valuable tool for analyzing and processing large amounts of text data, but it requires careful planning and implementation to be effective.



## References

- Shaikh, J. (2017, October 30). Machine Learning, NLP: Text classification using scikit-learn, python and NLTK. Medium. Retrieved March 7, 2023, from <https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a>
- San Francisco Bay University. SFBU. (n.d.). Retrieved March 7, 2023, from [https://hc.labnet.sfbu.edu/~henry/sfbu/course/mllib/naive\\_bayes/hw/q12/2021\\_summer/Text\\_Classifier.ipynb](https://hc.labnet.sfbu.edu/~henry/sfbu/course/mllib/naive_bayes/hw/q12/2021_summer/Text_Classifier.ipynb)