

# CS550 - Machine Learning and Business Intelligence

Machine Learning - Text Classification

By : Prince Rajasekaran  
Instructor: Dr. Chang, Henry

# Table of content

- **Introduction**
- **Types of text classifiers**
- **Manual calculation**
- **Implementation of Code**
- **Conclusion**
- **References**



# 1. Introduction

Text classification is a machine learning technique that assigns a set of predefined categories to open ended text. Text classifiers can be used to organize, structure, and categorize pretty much any kind of text – from documents, medical studies and files, and all over the web.



## 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.



## 2. Data collection

- The quantity & quality of your data dictate how accurate our model is
- The outcome of this step is generally a representation of data which we will use for training
- Using pre-collected data



### 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
- $= (\text{count}(W6, F) + 1) / (\text{count}(F) + |V|) = (1+1) / (9+6) = 2/15$

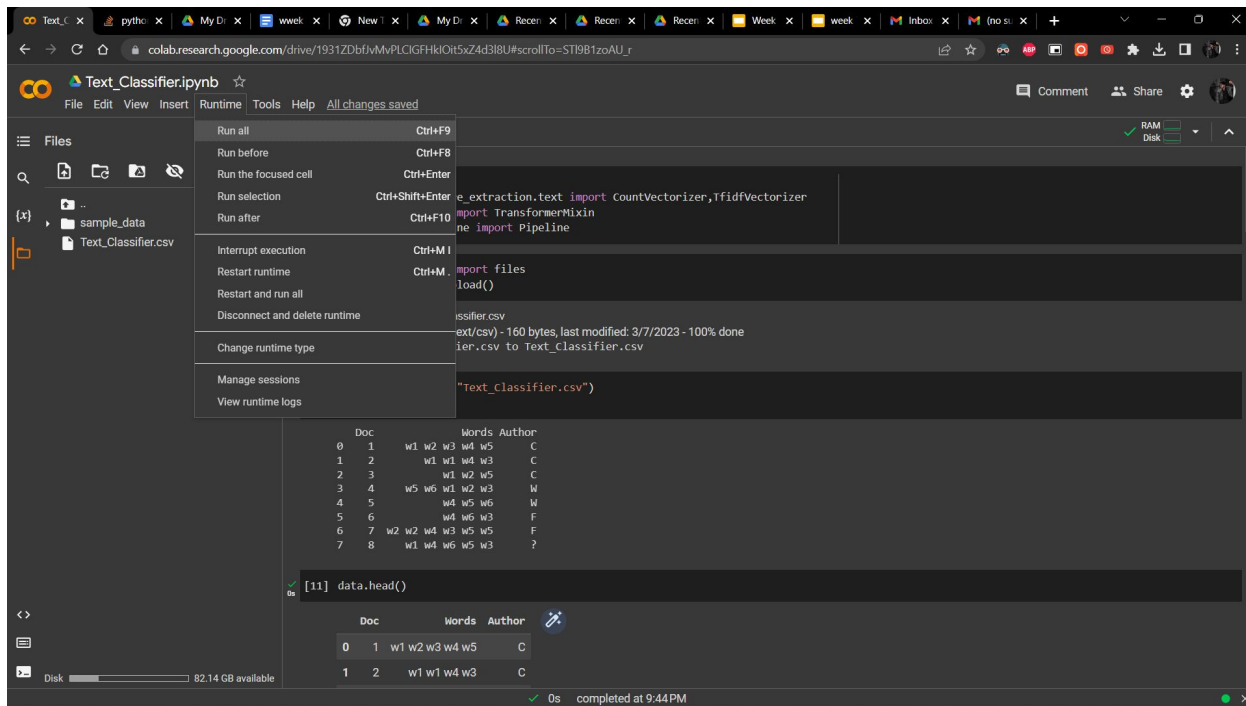




### 3. Manual calculation

- $P(C|d8) : P(C) * P(W1|C) * P(W4|C) * P(W6|C) * P(W5|C) * P(W3|C)$
- $= ((3/7) * (5/18) * (1/6) * (1/18) * (1/6) * (1/6))$
- $= 0.00003061924$ , approx. 0.00004
- $P(W|d8) = P(W) * P(W1|W) * P(W4|W) * P(W6|W) * P(W5|W) * P(W3|W)$
- $= (2/7 * 2/14 * 2/14 * 3/14 * 3/14 * 2/14)$
- $= 0.00002824936$ , approx. 0.00003
- $P(F|d8) = P(F) * P(W1|F) * P(W4|F) * P(W6|F) * P(W5|F) * P(W3|F)$
- $= ((2/7) * (1/15) * (3/15) * (2/15) * (3/15) * (3/15))$
- $= 0.00002031746$ , approx. 0.00002
- The probability calculations show that Document 8 should be in Class C because it has the highest
- probability calculation.

## 4. Implementation of Code



The screenshot shows a Google Colab notebook interface. The notebook is titled "Text\_Classifier.ipynb" and is open in a browser window. The left sidebar shows the file explorer with a folder named "sample\_data" containing a file "Text\_Classifier.csv". The main area displays a code cell with the following Python code:

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
import transformers
from transformers import pipeline

# Load the data
data = pd.read_csv('Text_Classifier.csv')

# Create a vectorizer
vectorizer = CountVectorizer()

# Fit the vectorizer on the data
vectorizer.fit(data['text'])

# Transform the data
data['text'] = vectorizer.transform(data['text']).toarray()

# Create a pipeline
classifier = pipeline('text-classification')

# Train the classifier
classifier.fit(data['text'], data['author'])
```

A runtime menu is open over the code cell, showing options such as "Run all", "Run before", "Run the focused cell", "Run selection", "Run after", "Interrupt execution", "Restart runtime", "Restart and run all", "Disconnect and delete runtime", "Change runtime type", "Manage sessions", and "View runtime logs".

Below the code cell, the output of the code is displayed as a table:

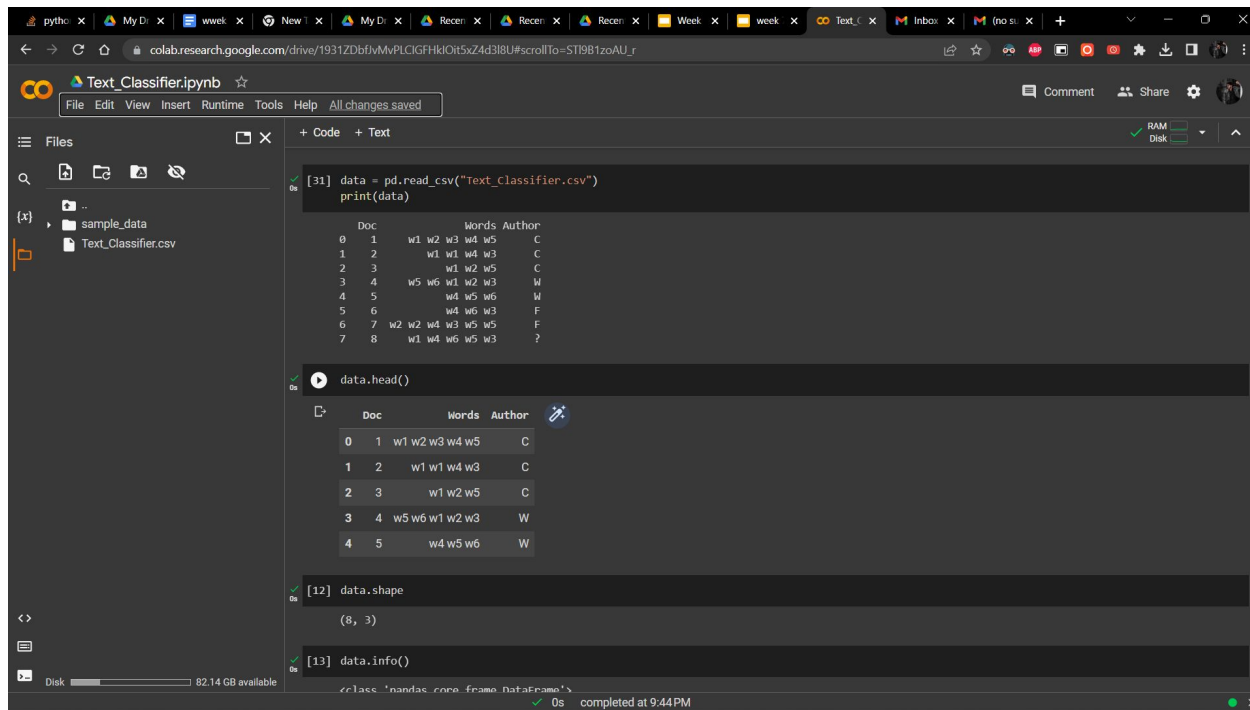
Doc	Words	Author
0	1 w1 w2 w3 w4 w5	C
1	2 w1 w1 w4 w3	C
2	3 w1 w2 w5	C
3	4 w5 w6 w1 w2 w3	W
4	5 w4 w5 w6	W
5	6 w4 w6 w3	F
6	7 w2 w2 w4 w3 w5 w5	F
7	8 w1 w4 w6 w5 w3	?

At the bottom of the notebook, the output of the code is displayed as a table:

Doc	Words	Author
0	1 w1 w2 w3 w4 w5	C
1	2 w1 w1 w4 w3	C

The status bar at the bottom indicates that the code was completed at 9:44 PM.

# Implementation



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `colab.research.google.com/drive/1931ZDbfjVwMvPLOGFhKt5x24d3l8U#scrollTo=ST9B1zoAU_r`. The notebook title is "Text\_Classifier.ipynb". The left sidebar shows a file explorer with a folder named "sample\_data" containing a file named "Text\_Classifier.csv".

The main code cell contains the following Python code:

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

The output of the code is a DataFrame with the following structure:

	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
7	8	w1 w4 w6 w5 w3	?

Below the DataFrame, the code `data.head()` is executed, showing the first 5 rows of the DataFrame.

The next code cell contains:

```
[12] data.shape
```

The output is:

```
(8, 3)
```

The final code cell contains:

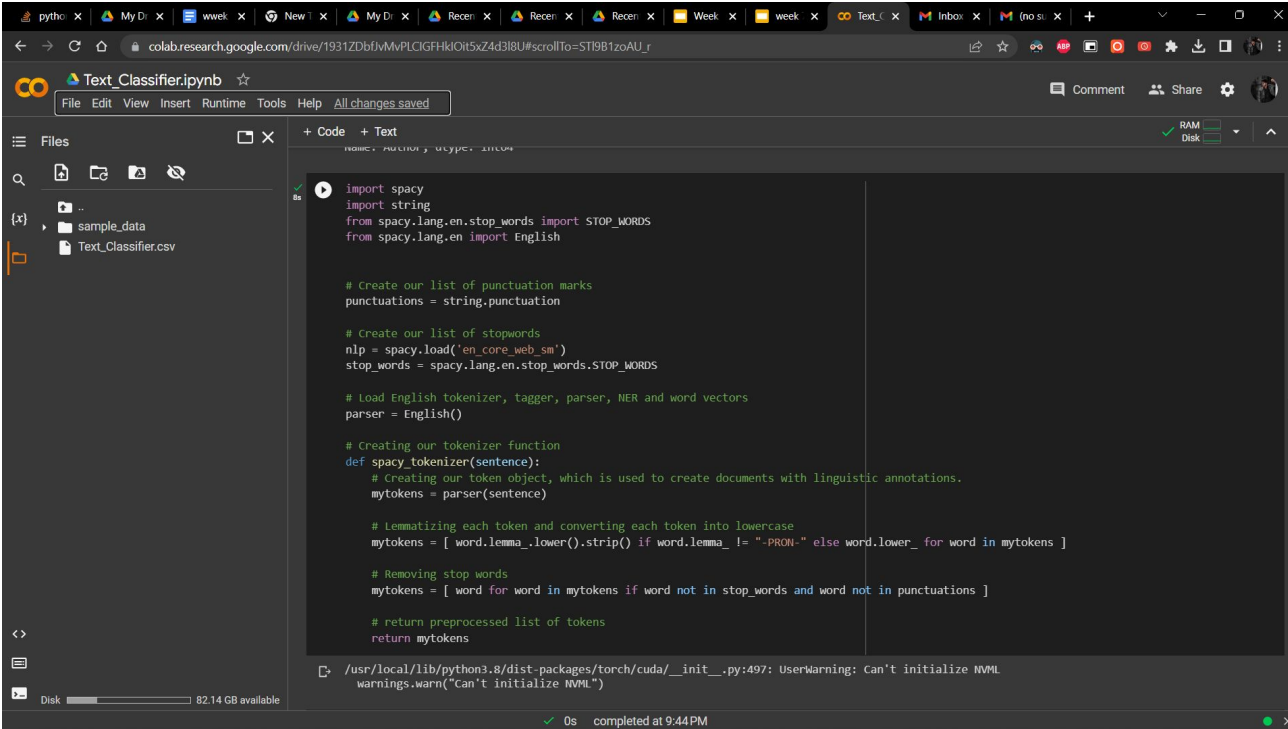
```
[13] data.info()
```

The output is:

```
<class 'pandas.core.frame.DataFrame'>
```

The status bar at the bottom indicates "Disk 82.14 GB available" and "completed at 9:44 PM".

# Implementation



The screenshot shows a Google Colab notebook interface. The top bar includes a menu (File, Edit, View, Insert, Runtime, Tools, Help) and a status bar indicating 'All changes saved'. The left sidebar shows a file explorer with a folder named 'sample\_data' containing a file 'Text\_Classifier.csv'. The main area displays a code cell with the following Python 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)

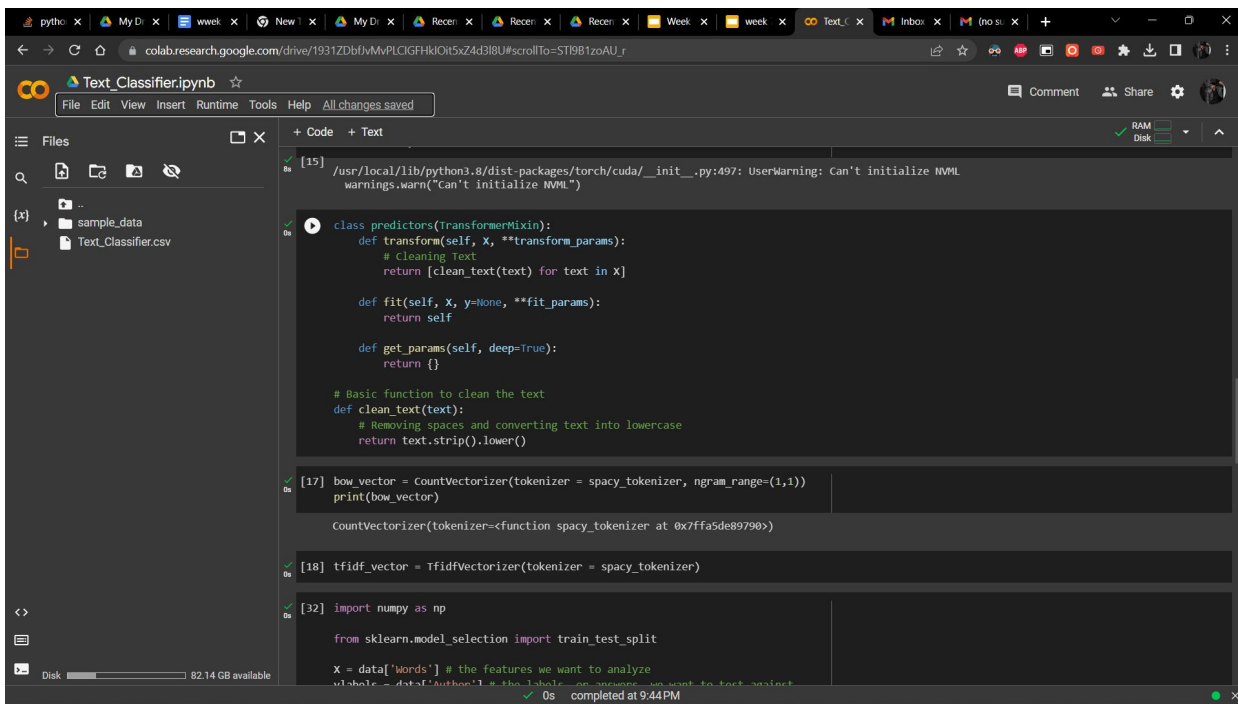
    # Lemmatizing each token and converting each token into lowercase
    mytokens = [ word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens ]

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

    # return preprocessed list of tokens
    return mytokens
```

At the bottom of the code cell, a warning message is displayed: `/usr/local/lib/python3.8/dist-packages/torch/cuda/_init_.py:497: UserWarning: Can't initialize NVML warnings.warn("Can't initialize NVML")`. The status bar at the very bottom indicates '0s completed at 9:44 PM'.

# Implementation



The screenshot displays a Google Colab notebook interface. The top toolbar includes options for File, Edit, View, Insert, Runtime, Tools, and Help. The left sidebar shows a file explorer with a folder named 'sample\_data' containing a file 'Text\_Classifier.csv'. The main workspace contains the following code:

```
[15] /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()

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

      CountVectorizer(tokenizer=<function spacy_tokenizer at 0x7ffa5de89790>)

[18] tfidf_vector = TfidfVectorizer(tokenizer = spacy_tokenizer)

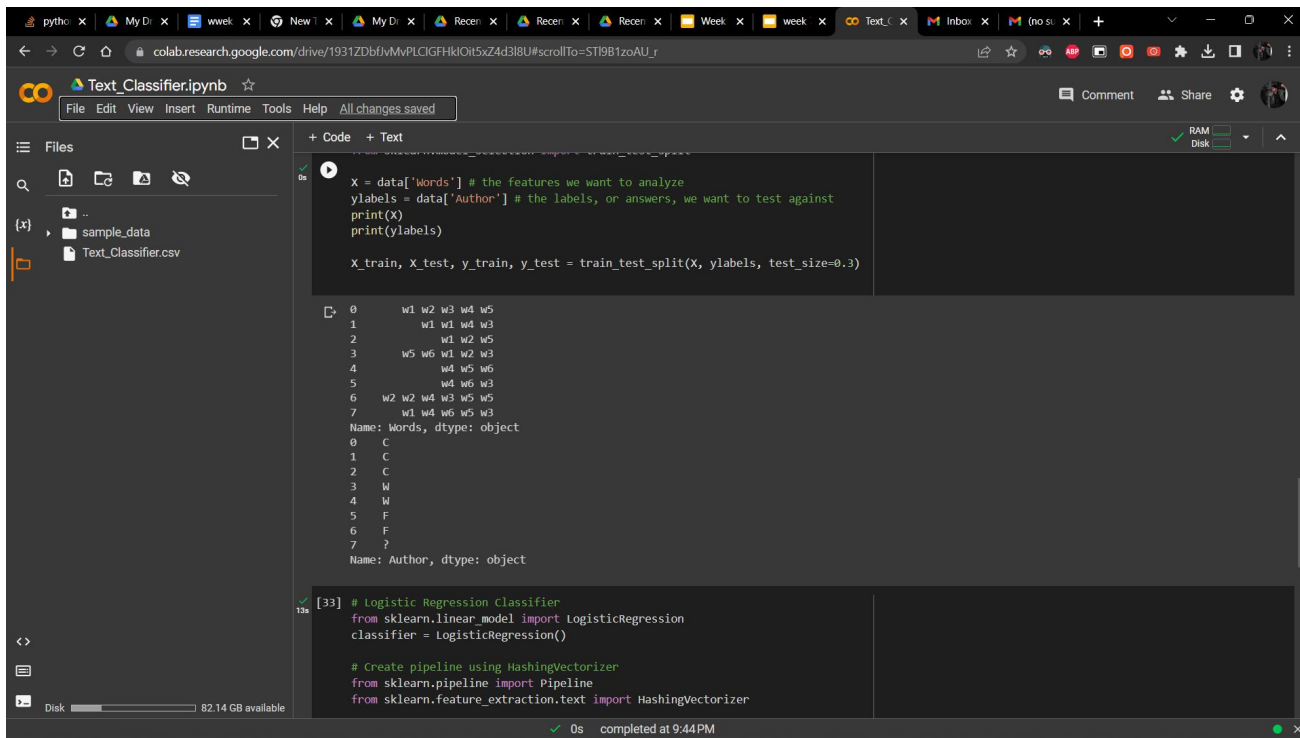
[32] import numpy as np

      from sklearn.model_selection import train_test_split

      X = data['Words'] # the features we want to analyze
      y_label = data['author'] # the labels we want to predict
```

The notebook status bar at the bottom indicates it was completed at 9:44 PM.

# Implementation



The screenshot shows a Google Colab notebook interface. The top bar includes tabs for various applications and a browser window showing the Colab URL. The notebook title is 'Text\_Classifier.ipynb'. The left sidebar shows a file explorer with a folder named 'sample\_data' containing a file 'Text\_Classifier.csv'. The main area displays the following code and its output:

```
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)
```

The output of the code is as follows:

```
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
7      w1 w4 w6 w5 w3
Name: Words, dtype: object
0      C
1      C
2      C
3      W
4      W
5      F
6      F
7      ?
Name: Author, dtype: object
```

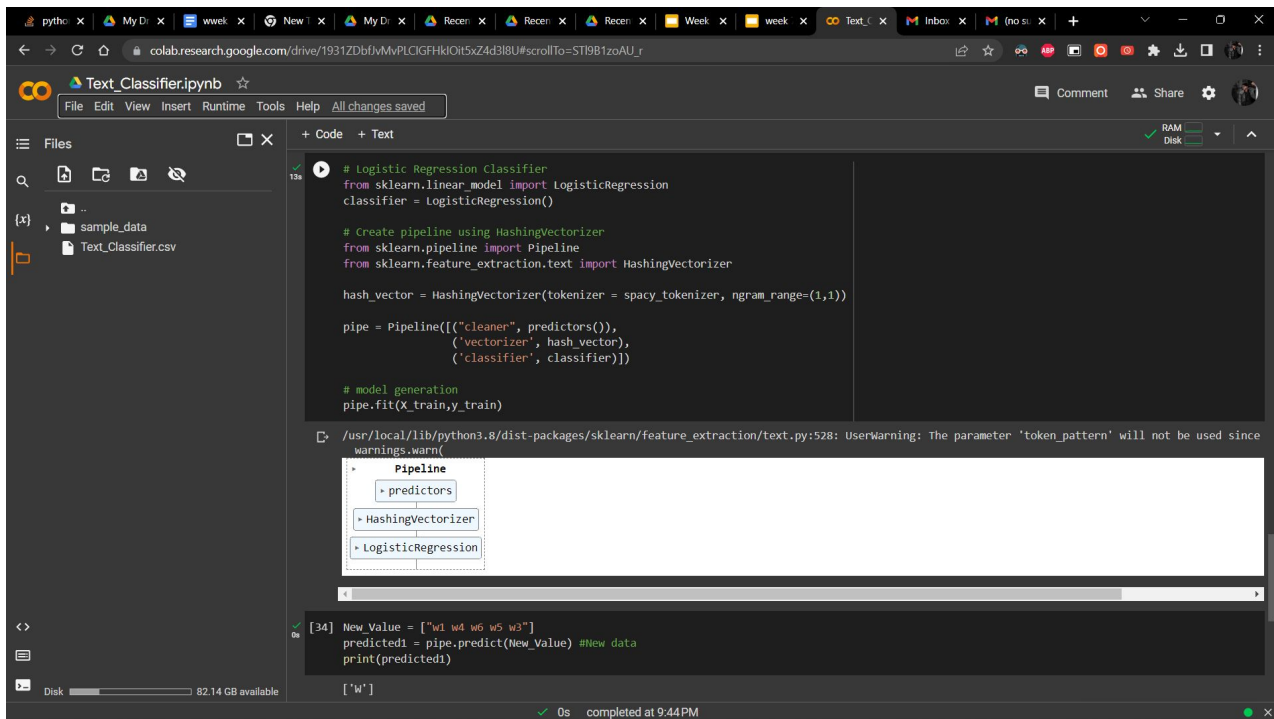
Below the output, the code continues with:

```
[33] # 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
```

The bottom status bar indicates '0s completed at 9:44 PM'.

# Implementation



The screenshot displays a Google Colab notebook interface. The top toolbar includes options for File, Edit, View, Insert, Runtime, Tools, and Help. The left sidebar shows a file explorer with a folder named 'sample\_data' and a file named 'Text\_Classifier.csv'. The main workspace contains a code cell with the following Python code:

```
# 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)
```

Below the code cell, a warning message is displayed: `Warning: The parameter 'token_pattern' will not be used since`. A visual representation of the pipeline is shown, consisting of three steps: 'predictors', 'HashingVectorizer', and 'LogisticRegression', connected sequentially. The bottom of the notebook shows the output of a cell, which is a list containing a single string: `['w']`. The status bar at the bottom indicates that the code was completed at 9:44 PM.

# 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.





# References

<https://levity.ai/blog/text-classifiers-in-machine-learning-a-practical-guide>

<https://monkeylearn.com/text-classification/#:~:text=Tutorial-,What%20is%20Text%20Classification%3F,and%20all%20over%20the%20web.>

