AI VIETNAM
All-in-One Course
(TA Session)

Text Classification

Project

Dinh-Thang Duong – TA

Anh-Khoi Nguyen – STA

Year

Getting Started

❖ Objectives

$$p(Y|X) = \frac{p(X|Y) \times p(Y)}{p(X)}$$

Spam	Not spam
	
"SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info"	"Nah I don't think he goes to usf, he lives around here though"

Our objectives:

- Discuss about Naïve Bayes algorithm.
- Delve into one of popular NLP tasks: Text Classification.
- Apply Naïve Bayes and its variants to solve a text classification task.
- Investigate an improved baseline over Naïve Bayes and conduct experiments.

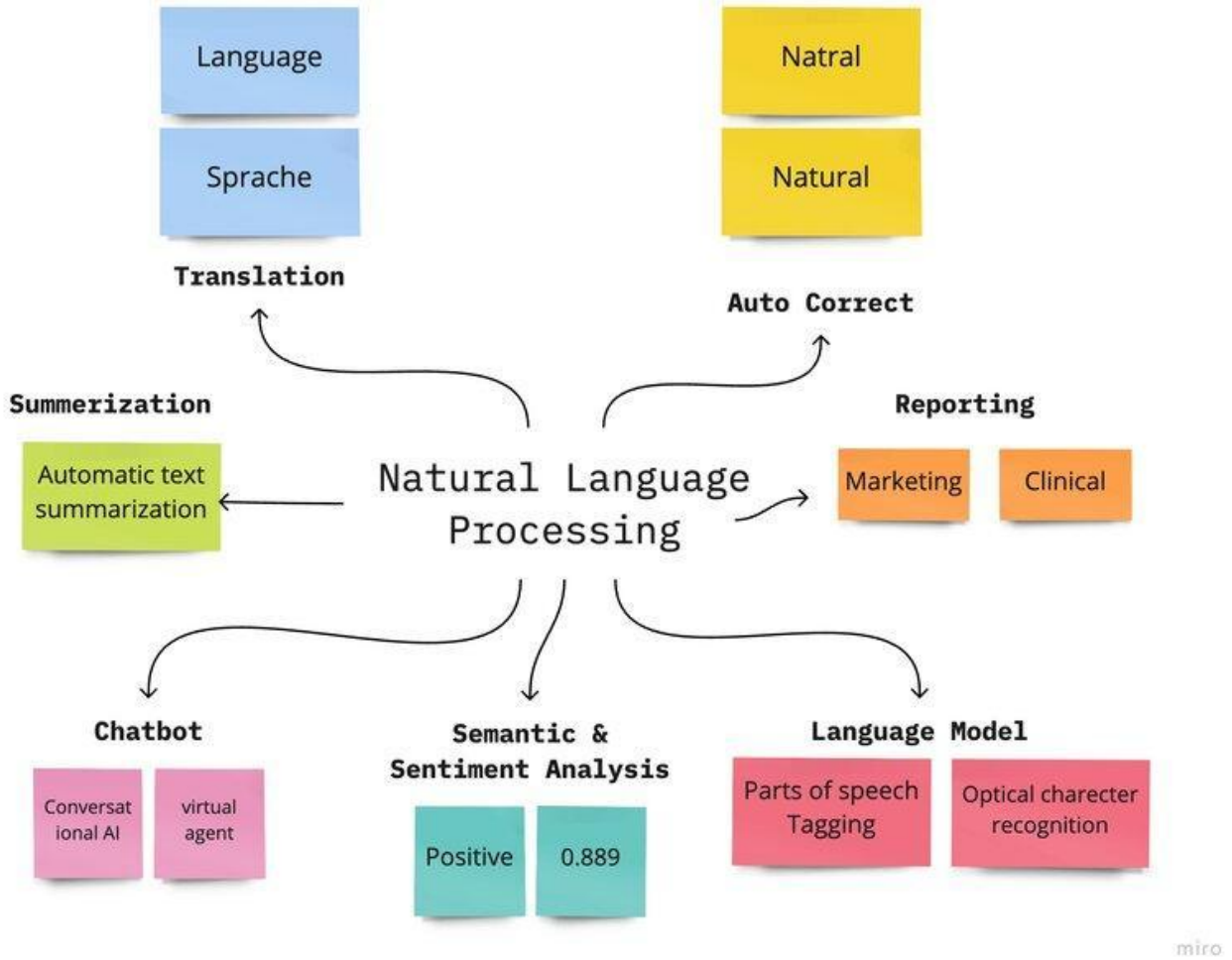
Outline

- Introduction
- Message Classification
- Code Implementation
- Improved Baseline
- Question

Introduction

Introduction

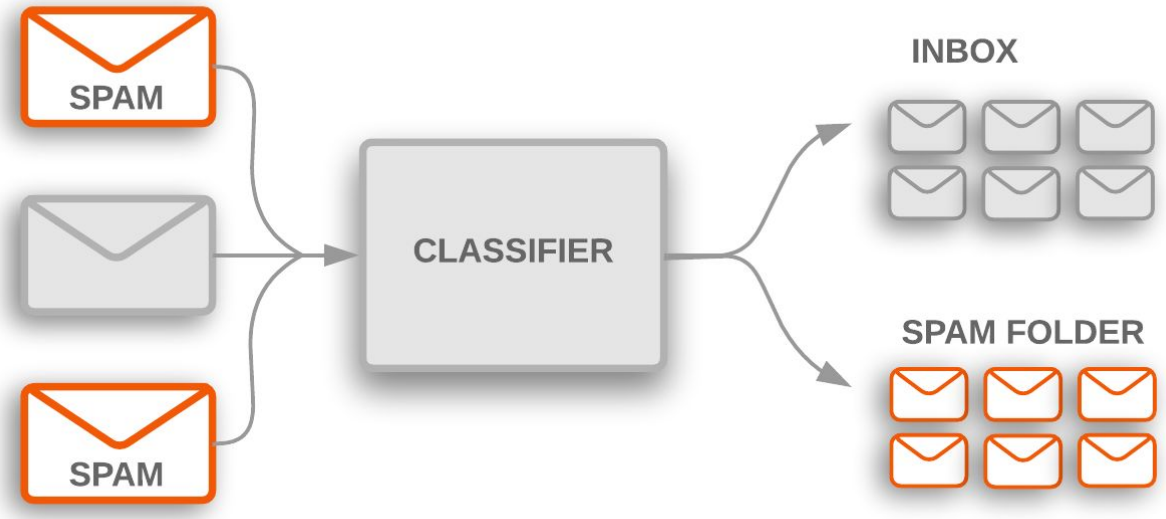
❖ Getting Started



https://www.researchgate.net/figure/Natural-Language-Processing_fig1_364842359

Introduction

❖ Text Classification

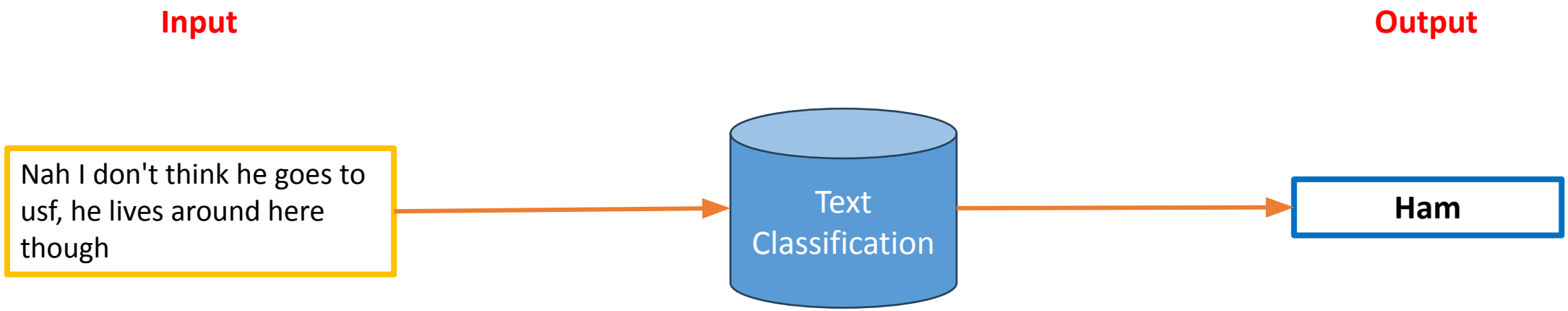


Text classification: A Natural Language Processing (NLP) task that involves categorizing text into predefined labels or classes. It is used to automatically assign a category to a text document, such as spam detection in emails, sentiment analysis of reviews, or topic classification of articles.

https://developers.google.com/machine-learning/guides/text-classification

Introduction

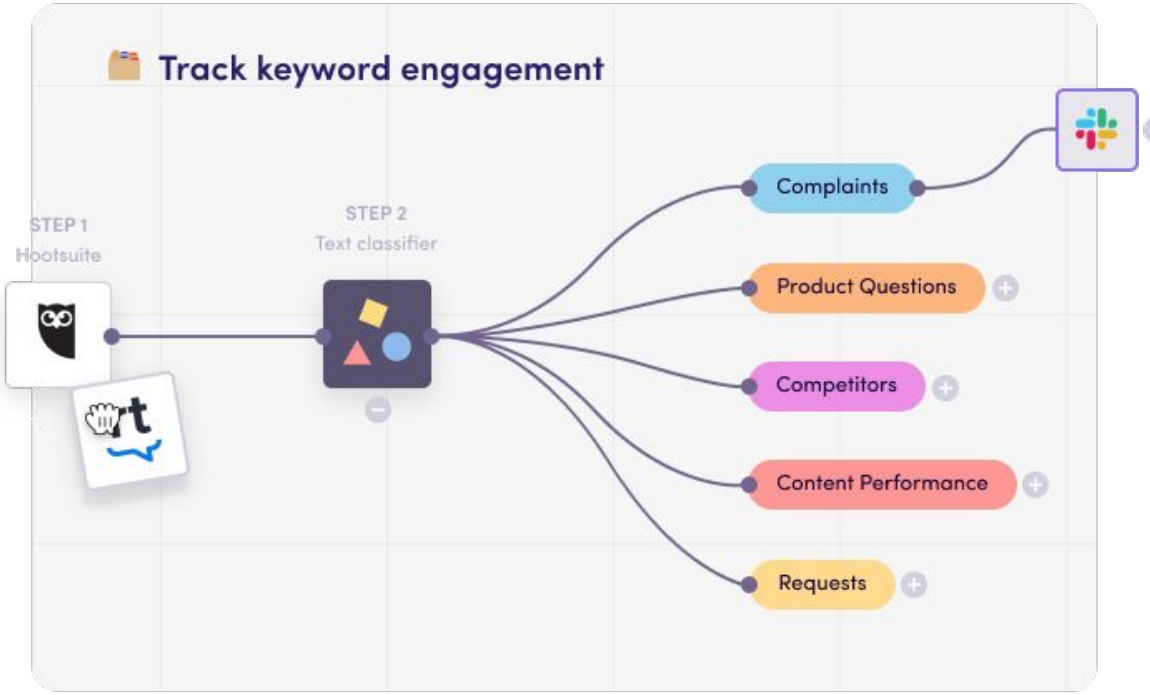
❖ Text Classification I/O



Text Classification: An NLP task that aims to classify a given text into pre-defined classes.

Introduction

❖ Applications

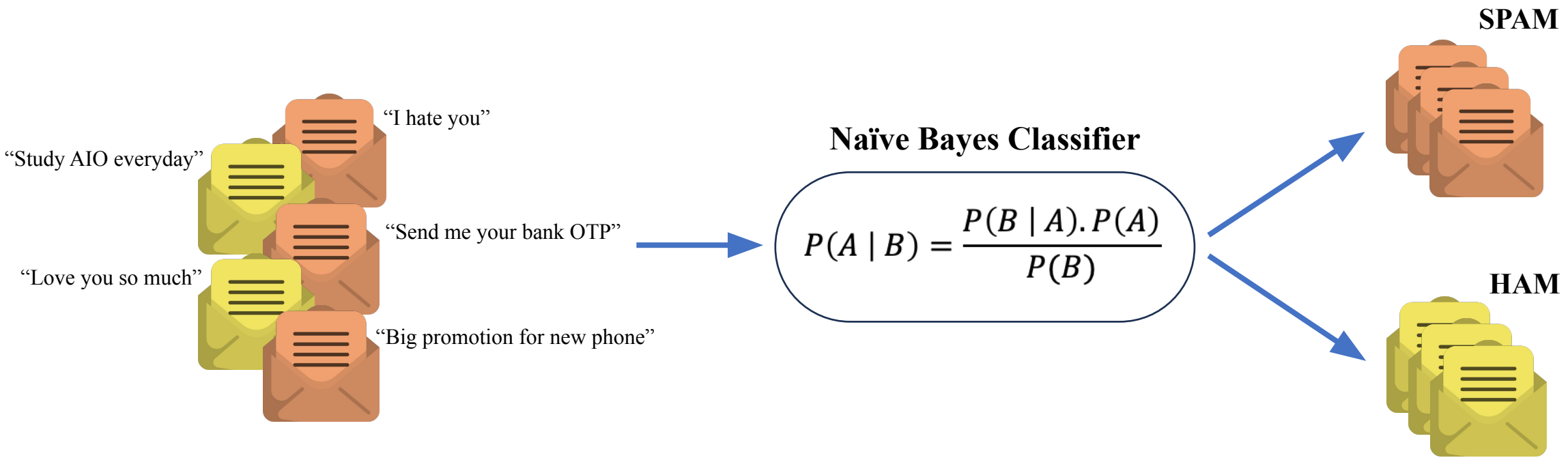


https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/
https://levity.ai/blog/9-text-classification-examples

Introduction

❖ Project Statement: Text Classification Naïve Bayes

Description: Given **Message Classification Dataset**, build a Naives Bayes model to determine whether a text message is spam message or not (ham).



https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

Message Classification

Message Classification

❖ Probability

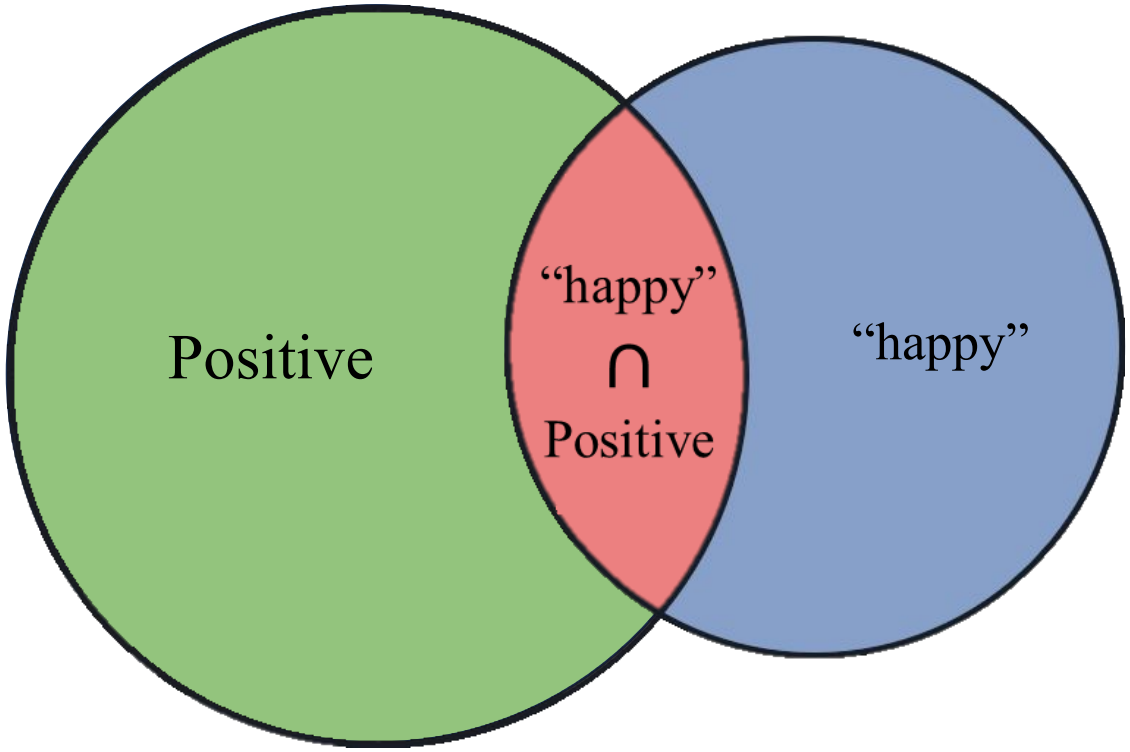
Positive	Positive	Positive	Positive	Positive
Positive	Positive	Positive	*Negative*	*Negative*
Negative	*Negative*	*Negative*	*Negative*	*Negative*
Negative	*Negative*	*Negative*	*Negative*	*Negative*

$$P(Pos) = \frac{N_{Pos}}{N} = \frac{8}{20} = 0.4$$

$$P(Neg) = 1 - P(Pos) = 1 - 0.4 = 0.6$$

Message Classification

❖ Conditional Probability



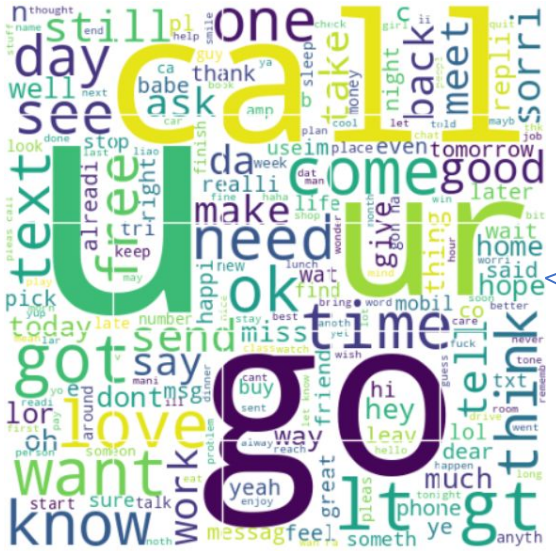
$$P("happy" \mid Pos) = \frac{P(Pos \cap "happy")}{P(Pos)}$$

Message Classification

❖ Bayes' Rule

The content of a letter consists of many **words** combined together

How do we know which letter is SPAM or HAM?



One of the ways is based on individual words

SPAM

"hate", "promotion", "sale", ...

? How do you know which group a word belongs to?

HAM

"happy", "love", "nice",...

Message Classification

❖ Bayes' Rule

How do you know which group a word belongs to?

Bayes' Rule: $$P(A \mid B) = \frac{P(B \mid A).P(A)}{P(B)}$$

$$P(Class \mid Words) = \frac{P(Words \mid Class)\ .P(Class)}{P(Words)}$$

$$P(Ham \mid w_1, \dots, w_n) = \frac{P(w_1, \dots, w_n \mid Ham).P(Ham)}{P(w_1, \dots, w_n)}$$

!!!

$$P(Ham \mid \text{"happy"}, \text{"hate"}, \text{"nice"}, \dots) = \frac{\mathbf{P(\text{"happy"}, \text{"hate"}, \text{"nice"}, \dots \mid Ham)}.P(Ham)}{P(\text{"happy"}, \text{"hate"}, \text{"nice"}, \dots)}$$

Message Classification

❖ Word Dependence

$$P("brown","sugar","pearl","milk","tea"|$$

Human:



“brown sugar pearl milk tea”

Naïve:





$$P("brown"|\ Ham).P("sugar"|\ Ham).P("pearl"|\ Ham).P("milk"|\ Ham).P("tea"|\ Ham)$$

Message Classification

❖ Naïve Bayes

Bayes' rule with the "Naïve" is the presence of one feature (word) does not affect the presence of another (word).

$$P("happy","hate","nice",\dots \mid Ham) = \prod_{i=1}^{n} P(word_i \mid y)$$

$$= P("happy" \mid Ham)\, P("hate" \mid Ham)\, P("nice" \mid Ham)\, P(" \quad " \mid Ham)$$

$$P(Ham \mid "happy","hate","nice",\dots) = \frac{\prod_{i=1}^{n} \boldsymbol{P(word_i \mid Ham)} \,.\, P(Ham)}{P("happy","hate","nice",\dots)}$$

Message Classification

❖ Remove constant

This value does not change when we consider different y classes.

$$P(Ham \mid "happy", "hate", "nice", \dots) = \frac{\prod_{i=1}^{n} P(word_i \mid Ham) \,.\, P(Ham)}{\boldsymbol{P("happy", "hate", "nice", \dots)}}$$

$$P(Spam \mid "happy", "hate", "nice", \dots) = \frac{\prod_{i=1}^{n} P(word_i \mid Spam) \,.\, P(Spam)}{\boldsymbol{P("happy", "hate", "nice", \dots)}}$$

Message Classification

❖ Remove constant

This value does not change when we consider different y classes.

$$P(Ham \mid "happy", "hate", "nice", \dots) = \frac{\prod_{i=1}^{n} P(word_i \mid Ham) \cdot P(Ham)}{P("happy", "hate", "nice", \dots)}$$

$$P(Spam \mid "happy", "hate", "nice", \dots) = \frac{\prod_{i=1}^{n} P(word_i \mid Spam) \cdot P(Spam)}{P("happy", "hate", "nice", \dots)}$$

Naïve Bayes Mail Classification :

Message Classification

❖ Project Statement

Description: Given **Message Classification Dataset**, build a Naives Bayes model to determine whether a text message is spam message or not (ham).

Spam	Not spam
	
"SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info"	"Nah I don't think he goes to usf, he lives around here though"

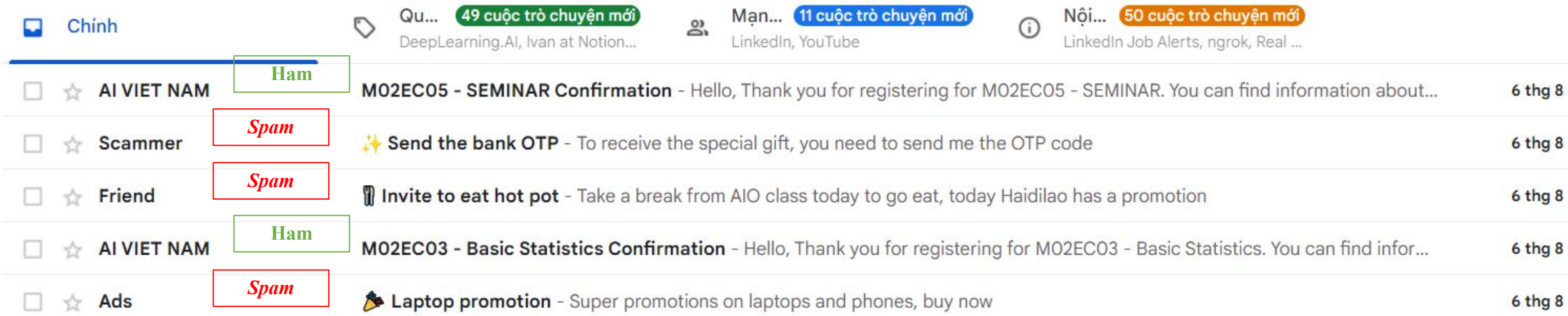
Message Classification

Introduction

Email classification is a task where we determine whether a given Mail is

- Spam (unsolicited or unwanted) *Spam*
- Ham (non-spam) Ham

Suppose you are "all in" AIO and you receive the following emails:



Message Classification

❖ Message Classification I/O

Spam	Not spam
	
"SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info"	"Nah I don't think he goes to usf, he lives around here though"

Problem Statement: Given a message, classify it into one of the two classes: Spam or Ham

Input: "Nah I don't think he goes to usf, he lives around here though"
Output: "Ham"

Message Classification Problem

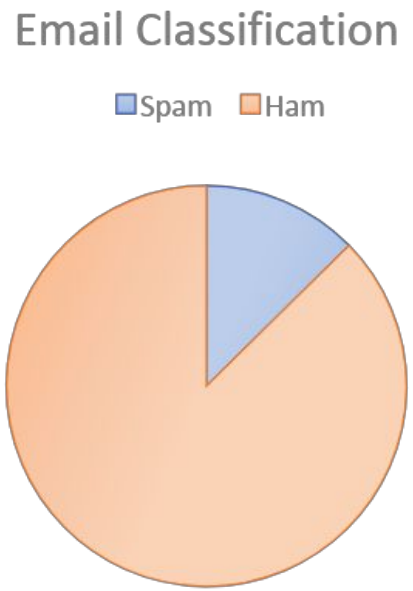
Message Classification

❖ Dataset

The data set includes 2 columns:

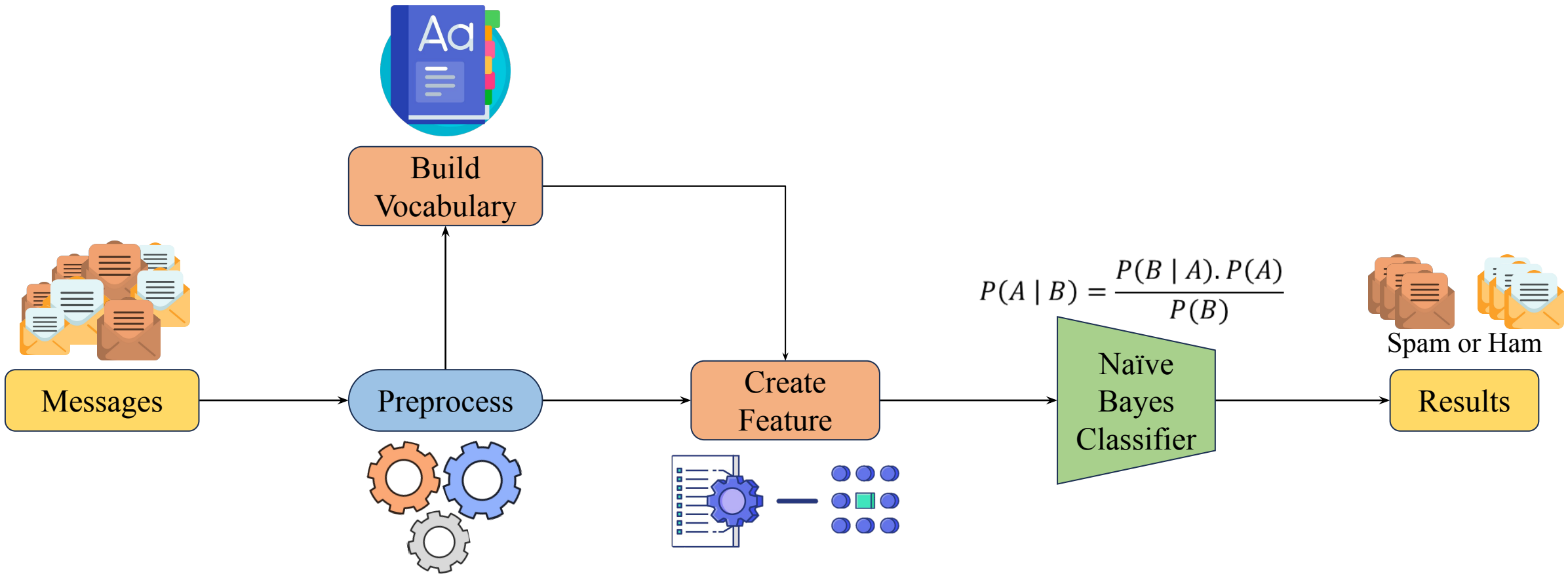
- Category (mail type)
- Message (mail content)

Category	Message
ham 87% spam 13%	5157 unique values
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got a...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entr...
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for



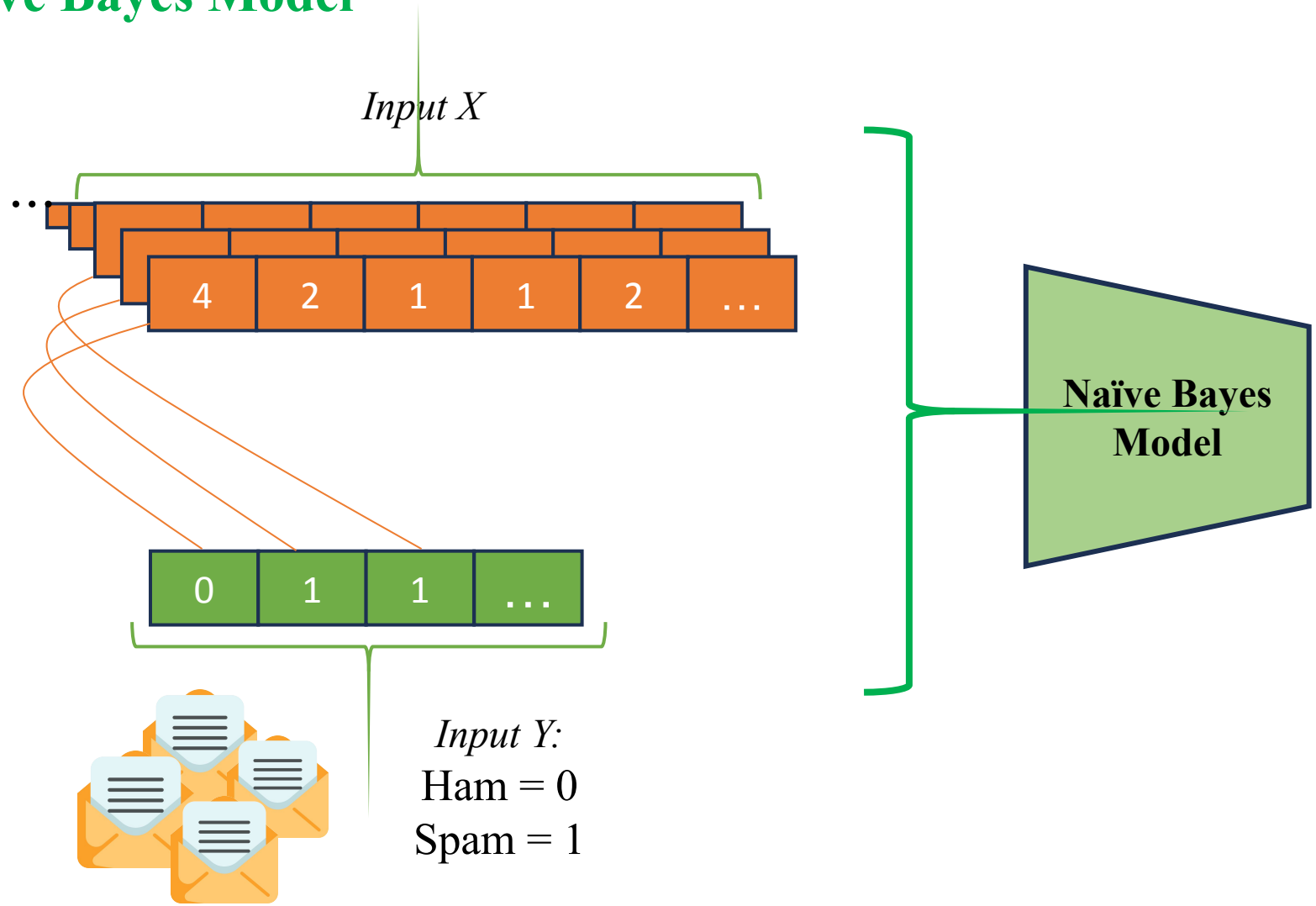
Message Classification

❖ Project Pipeline



Message Classification

❖ Input Naïve Bayes Model

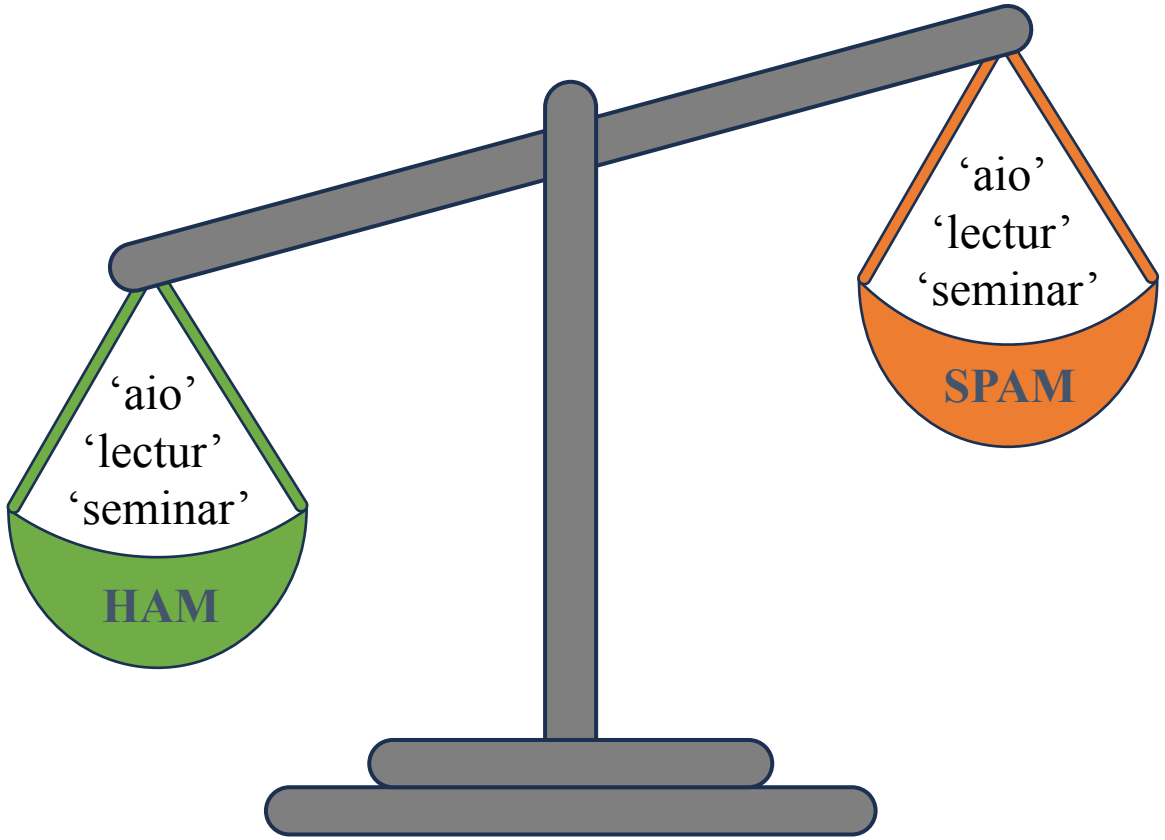


Message Classification

❖ Applying Naïve Bayes with Maximum A Posteriori Estimation

To classification, using MAP to estimation $P(y)$ and $P(word_i \mid y)$, where y is a label in training set.
Which y has the highest probability is the model result

$$\hat{y} = \boldsymbol{arg\,max}_{y} P(y) . \prod_{i=1}^{n} P(word_i \mid y)$$



Message Classification

❖ $P(x_i|y)$

Naïve Bayes Model

$$\hat{y} = arg\max_y P(y) . \prod_{i=1}^{n} P(x_i \mid y)$$

AIO	I	study	we	love
4	2	1	1	2
0	1	1	0	1
0	3	0	2	0

Suppose Input X:

0	0	1

Suppose Input Y:

	Count	
Word	Ham	Spam
aio	4 + 0 = 4	0
i	2 + 1 = 3	3
studi	1 + 1 = 2	0
we	1 + 0 = 1	2
love	2 + 1 = 3	0
V	**4+3+2+1+3=13**	**3+2=5**

Message Classification

❖ $P(x_i|y)$

Naïve Bayes Model

$$\hat{y} = arg\max_y P(y) . \prod_{i=1}^{n} P(x_i \mid y)$$

Word	Ham	Spam
aio	4 / 13	0 / 5
i	3 / 13	3 / 5
studi	2 / 13	0 / 5
we	1 / 13	2 / 5
love	3 / 13	0 / 5
V	**13**	**5**

→

Word	Ham	Spam
aio	0.308	0
i	0.231	0.6
studi	0.154	0
we	0.077	0.4
love	0.231	0
V	**13**	**5**

Message Classification

❖ $P(y)$

Naïve Bayes Model

$$\hat{y} = arg\max_{y} P(y) . \prod_{i=1}^{n} P(x_i \mid y)$$

Suppose Input Y:

0	0	1

$$P(Ham) = \frac{2}{N} = \frac{2}{3} = 0.667$$

$$P(Spam) = \frac{1}{N} = \frac{1}{3} = 0.334$$

Message Classification

❖ Predict

Naïve Bayes Model

Word	Ham	Spam
aio	0.308	0
i	0.231	0.6
studi	0.154	0
we	0.077	0.4
love	0.231	0
V	**13**	**5**

$P(Ham) = 0.667$

$P(Spam) = 0.334$

Predict

New mail: "**I study AIO, I love it**"

$$\hat{y} = arg\max_{y} P(y) . \prod_{i=1}^{n} P(word_i \mid y)$$

$$\hat{y}_{Ham} = (0.667).(0.231 * 0.154 * 0.308 * 0.231 * 0.231)$$
$$\approx 0.000389971807067304$$

$$\hat{y}_{Spam} = (0.334).(0.6 * 0.01 * 0.01 * 0.6 * 0.01)$$
$$\approx 0.00000012024$$

$$\hat{y}_{Ham} > \hat{y}_{Spam} \rightarrow Predict = Ham$$

Message Classification

❖ Problem 1 – Zero

Naïve Bayes Model

$$\hat{y} = arg\max_{y} P(y) \cdot \prod_{i=1}^{n} P(word_i \mid y)$$

Word	Ham	Spam
aio	4 / 13	0 / 5
i	3 / 13	3 / 5
studi	2 / 13	0 / 5
we	1 / 13	2 / 5
love	3 / 13	0 / 5
V	**13**	**5**

Word	Ham	Spam
aio	0.308	0 !
i	0.231	0.6
studi	0.154	0 !
we	0.077	0.4
love	0.231	0 !
V	**13**	**5**

Message Classification

❖ Problem 1 - Laplacian Smoothing

Word	Ham	Spam
aio	4	0
i	3	3
studi	2	0
we	1	2
love	3	0
V	**13**	**5**

$$P(x_i \mid y) = \frac{P(x_i) + 1}{V + N_{dictionary}}$$

Word	Ham	Spam
aio	(4+1)/(13+5)	(0+1)/(5+5)
i	(3+1)/(13+5)	(3+1)/(5+5)
studi	(2+1)/(13+5)	(0+1)/(5+5)
we	(1+1)/(13+5)	(2+1)/(5+5)
love	(3+1)/(13+5)	(0+1)/(5+5)
V	**13**	**5**

Message Classification

❖ Problem 1 - Laplacian Smoothing

$$P(x_i \mid y) = \frac{P(x_i) + 1}{V + N_{dictionary}}$$

Word	Ham	Spam
aio	(4+1)/(13+5)	(0+1)/(5+5)
i	(3+1)/(13+5)	(3+1)/(5+5)
studi	(2+1)/(13+5)	(0+1)/(5+5)
we	(1+1)/(13+5)	(2+1)/(5+5)
love	(3+1)/(13+5)	(0+1)/(5+5)

Word	Ham	Spam
aio	0.278	0.1
i	0.222	0.4
studi	0.167	0.1
we	0.111	0.3
love	0.222	0.1
Sum	**1**	**1**

Message Classification

❖ Problem 1 - Laplacian Smoothing

Word	Ham	Spam
aio	0.278	0.1
i	0.222	0.4
studi	0.167	0.1
we	0.111	0.3
love	0.222	0.1
Sum	**1**	**1**

Predict

New mail: "**I study AIO, I love it.**"

$$\hat{y}_{Ham} = (0.667).(0.222 * 0.167 * 0.278 * 0.222 * 0.222) \approx 0.00050794$$

$$\hat{y}_{Spam} = (0.334).(0.1 * 0.4 * 0.1 * 0.3 * 0.1) \approx 0.00012$$

$$\hat{y}_{Ham} > \hat{y}_{Spam} \rightarrow Predict = Ham$$

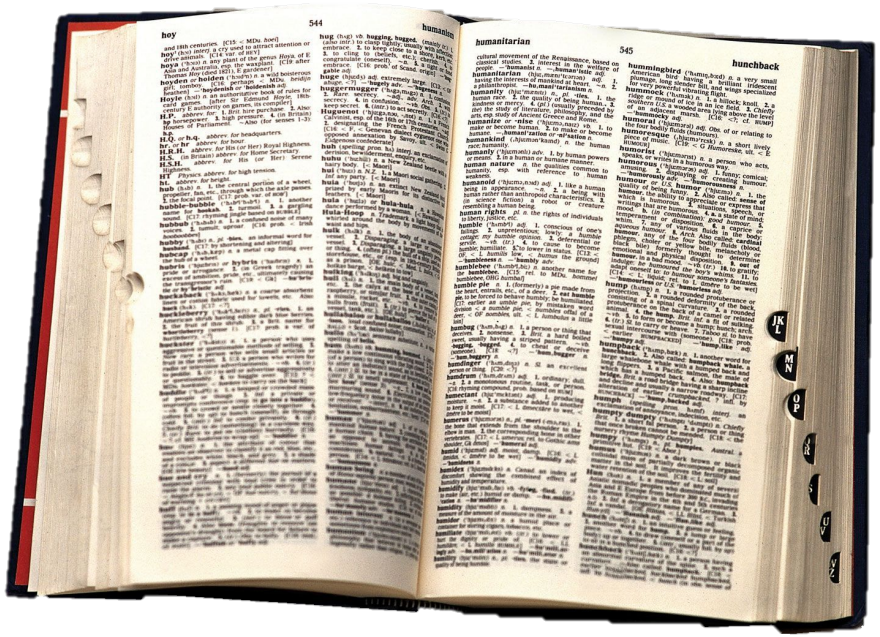
Message Classification

❖ Problem 2 – Log Likelihood

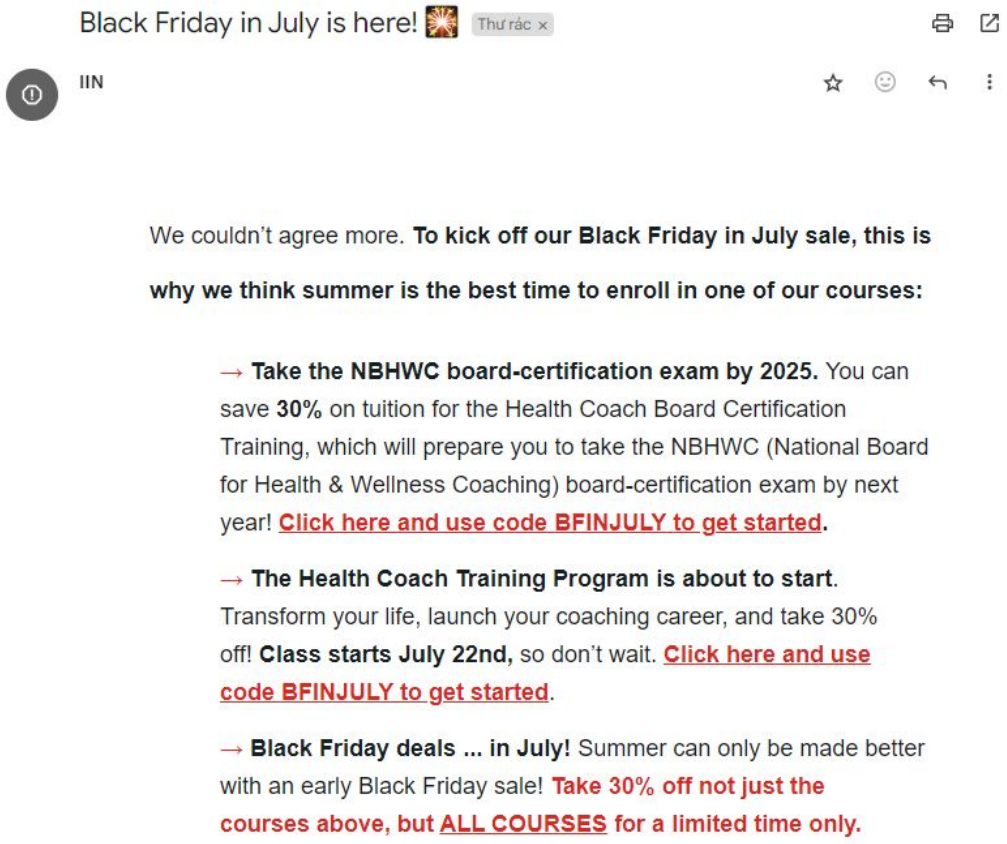
$\hat{y}_{Ham} \approx 0.00050794$!

$\hat{y}_{Spam} \approx 0.00012$!

In reality, the dictionary is huge



and the message content much longer



Black Friday in July is here! Thư rác ×

IIN

We couldn't agree more. **To kick off our Black Friday in July sale, this is why we think summer is the best time to enroll in one of our courses:**

→ **Take the NBHWC board-certification exam by 2025.** You can save **30%** on tuition for the Health Coach Board Certification Training, which will prepare you to take the NBHWC (National Board for Health & Wellness Coaching) board-certification exam by next year! **Click here and use code BFINJULY to get started.**

→ **The Health Coach Training Program is about to start.** Transform your life, launch your coaching career, and take 30% off! **Class starts July 22nd,** so don't wait. **Click here and use code BFINJULY to get started.**

→ **Black Friday deals ... in July!** Summer can only be made better with an early Black Friday sale! **Take 30% off not just the courses above, but ALL COURSES for a limited time only.**

The results will become very very small, gradually approaching 0

Message Classification

❖ Problem 2 – Log Likelihood

Take advantage of the **Scaling** property of *Logarithms*.

0.00001	-5
0.001	-3
0	Error
100	3
10000	5

$$\prod_{i=1}^{n} P(word_i \mid y) = \sum_{i=1}^{n} \log(P(word_i \mid y))$$

Message Classification

❖ Problem 2 – Log Likelihood

Word	Ham	Spam
aio	0.278	0.1
i	0.222	0.4
studi	0.167	0.1
we	0.111	0.3
love	0.222	0.1

Predict

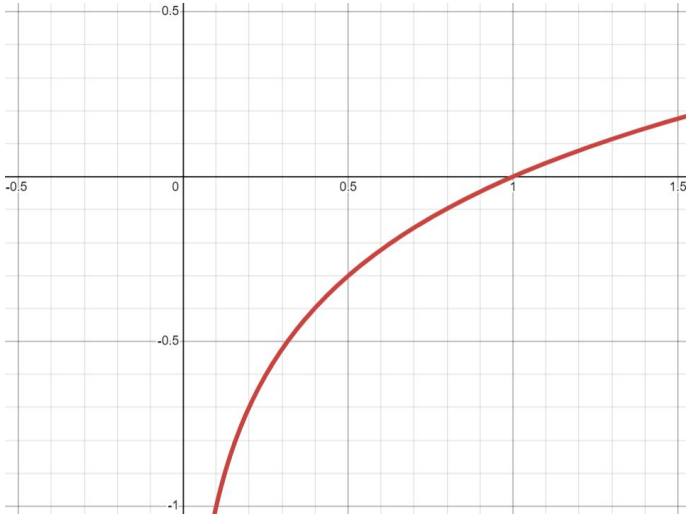
New mail: "**I study AIO, I love it.**"

$$\hat{y}_{Ham} = (0.667).(\log(0.222) + \log(0.167) + \log(0.278) + \log(0.222) + \log(0.222))$$
$$\approx (0.667).(-3.294) \approx -2.197098$$

$$\hat{y}_{Spam} = (0.334).(\log(0.1) + \log(0.4) + \log(0.1) + \log(0.3) + \log(0.1))$$
$$\approx (0.334).(-3.921) \approx -1.30955346$$

Because taking the Logarithms gives a value less than 1, the label

$$\hat{y}_{Ham} < \hat{y}_{Spam} \rightarrow Predict = Ham$$

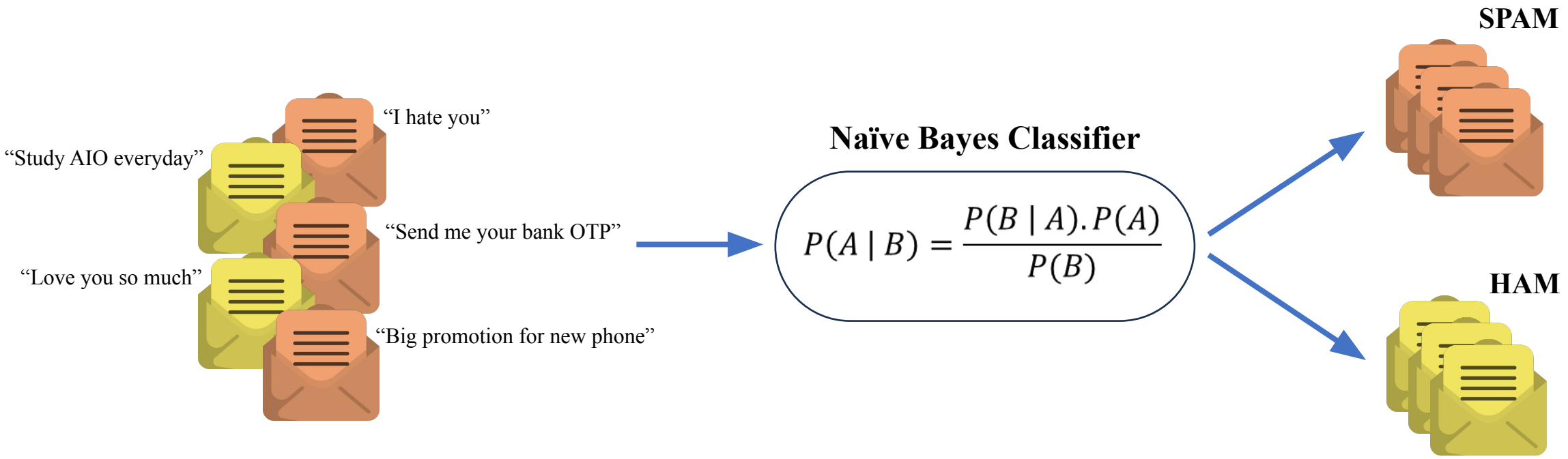


Code Implementation

Code Implementation

❖ Introduction

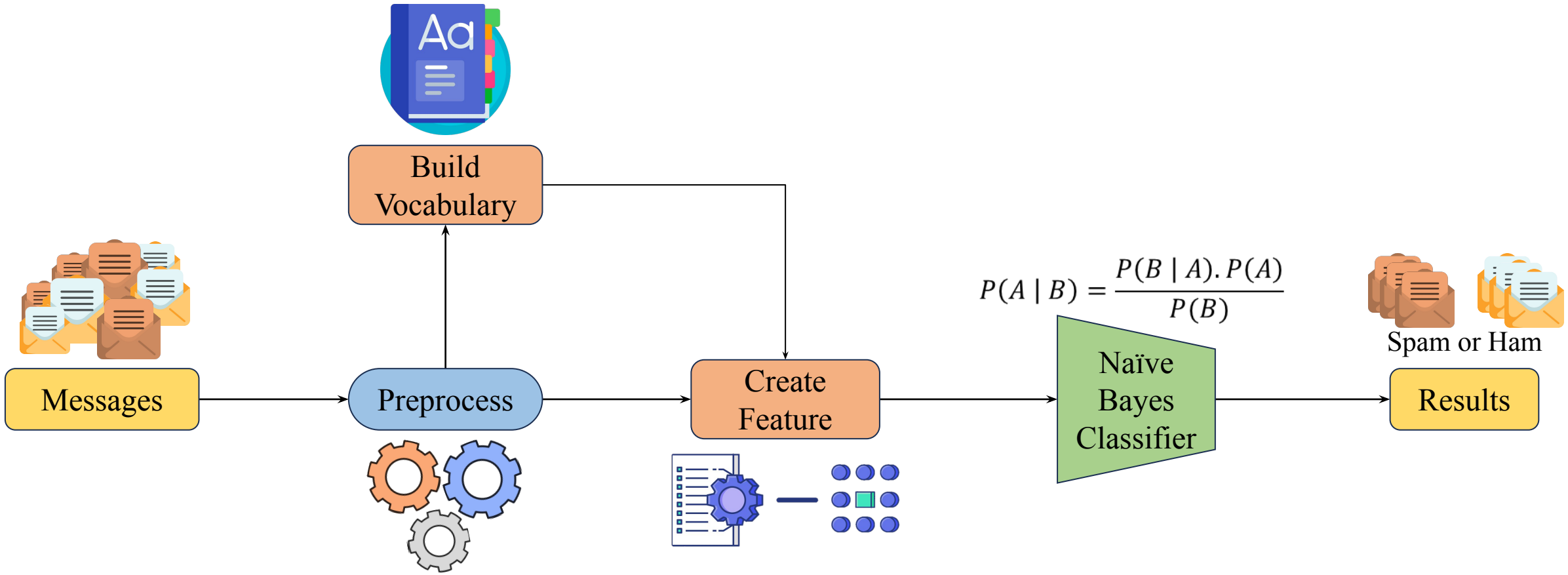
Description: Given **Message Classification Dataset**, build a Naives Bayes model to determine whether a text message is spam message or not (ham).



https://www.cs.unb.ca/~hzhang/publications/FLAIRS04ZhangH.pdf

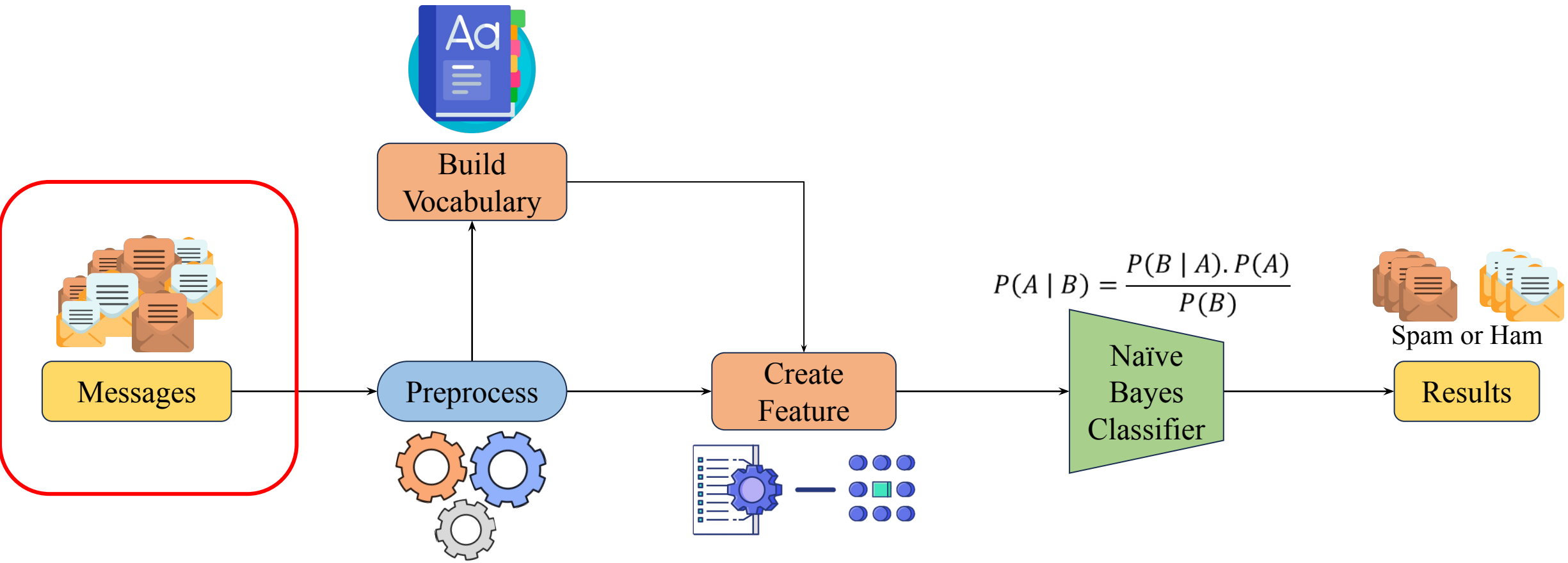
Code Implementation

❖ Project Pipeline



Code Implementation

❖ Project Pipeline



Code Implementation

❖ Coding step: Import libraries

```
1 import string
2 import nltk
3 nltk.download('stopwords')
4 nltk.download('punkt')
5 import pandas as pd
6 import numpy as np
7 import matplotlib.pyplot as plt
8
9 from sklearn.model_selection import train_test_split
10 from sklearn.naive_bayes import GaussianNB
11 from sklearn.metrics import accuracy_score
12 from sklearn.preprocessing import LabelEncoder
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

scikit-learn (sklearn): An open-source library for Python language that features various classification, regression and clustering algorithms.

Code Implementation

❖ Coding step: Download dataset

Description: Given **Message Classification Dataset**, build a Naives Bayes model to determine whether a text message is spam message or not (ham).

```
# https://drive.google.com/file/d/1N7rk-kfnDFIGMeX0ROVTjKh71gcgx-7R/view?usp=sharing
!gdown --id 1N7rk-kfnDFIGMeX0ROVTjKh71gcgx-7R
```

```
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132: FutureWarning: Option `--id`
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1N7rk-kfnDFIGMeX0ROVTjKh71gcgx-7R
To: /content/2cls_spam_text_cls.csv
100% 486k/486k [00:00<00:00, 6.59MB/s]
```

Code Implementation

❖ Coding step: Read dataset

To read .csv file, we use pandas.read_csv():

```
1 DATASET_PATH = '/content/2cls_spam_text_cls.csv'
2 df = pd.read_csv(DATASET_PATH)
3 df
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

Code Implementation

❖ Coding step: Read dataset

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

```
messages = df['Message'].values.tolist()
labels = df['Category'].values.tolist()
```

pandas.Series.to_list

`Series.to_list()` [source]

Return a list of the values.

These are each a scalar type, which is a Python scalar (for str, int, float) or a pandas scalar (for Timestamp/Timedelta/Interval/Period)

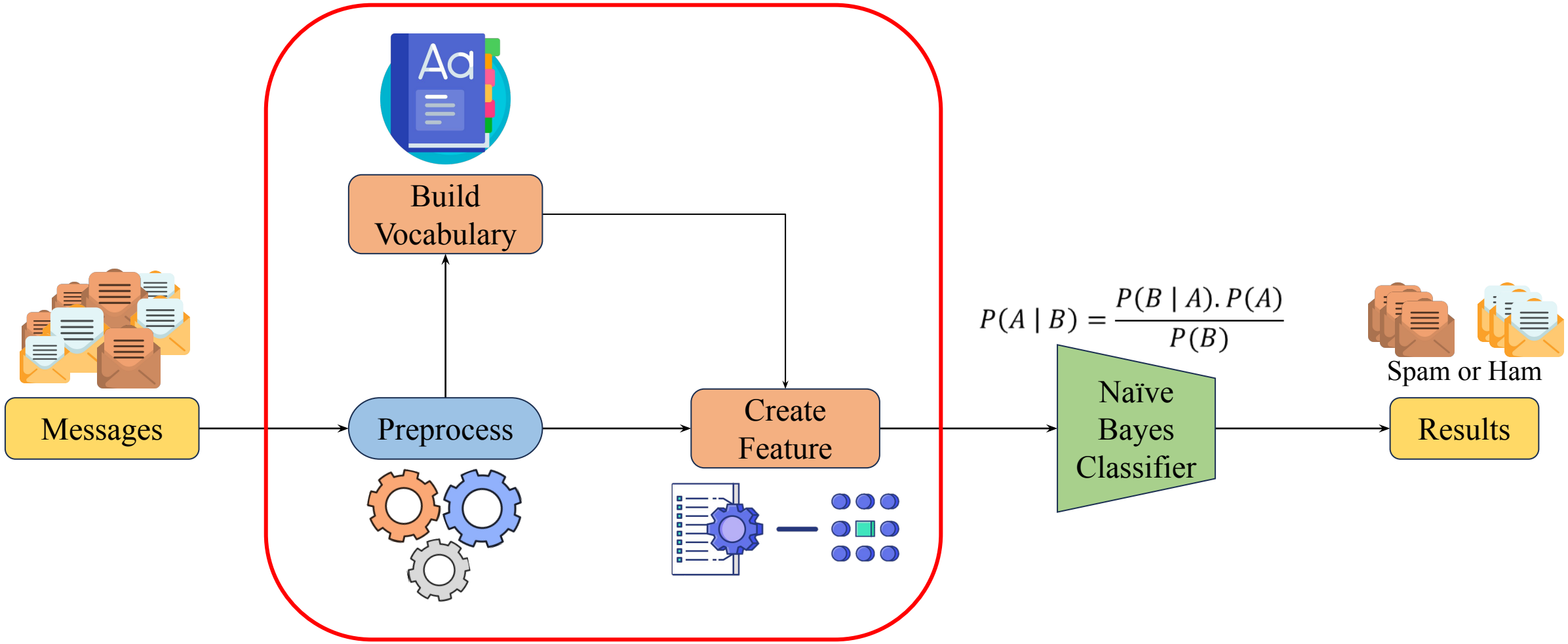
Returns:

list

https://pandas.pydata.org/docs/reference/api/pandas.Series.to_list.html

Code Implementation

❖ Project Pipeline



Code Implementation

❖ Text features

Corpus

doc1 = “deep learning book”

doc2 = “machine learning algorithm”

doc3 = “learning ai from scratch”

doc4 = “ai vietnam”

Tokenization →

[‘deep’, ‘learning’, ‘book’]

[‘machine’, ‘learning’, ‘algorithm’]

[‘learning’, ‘ai’, ‘from’, ‘scratch’]

[‘ai’, ‘vietnam’]

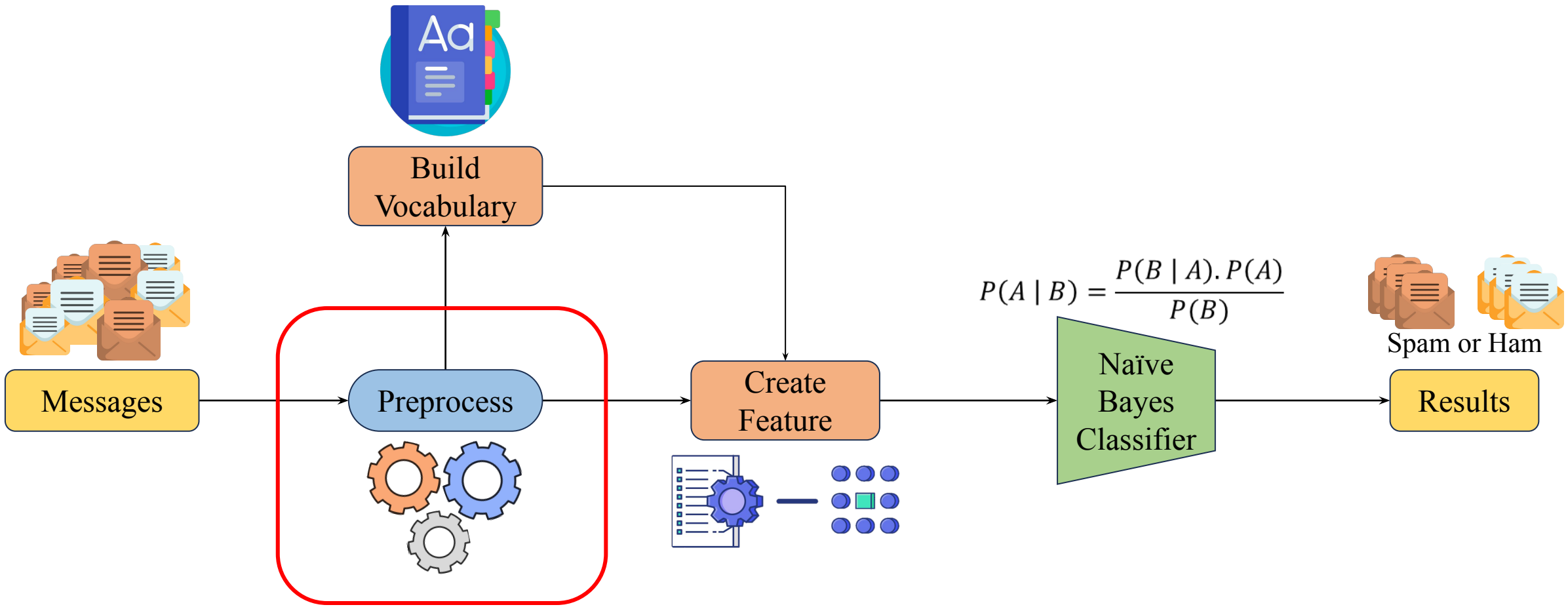
Vocabulary =	deep	learning	book	machine	algorithm	ai	from	scratch	vietnam

👉 **Given a string =** “vietnam machine learning deep learning book”

	deep	learning	book	machine	algorithm	ai	from	scratch	vietnam
BoW	1	2	1	1	0	0	0	0	1
Binary BoW	1	1	1	1	0	0	0	0	1

Code Implementation

❖ Project Pipeline



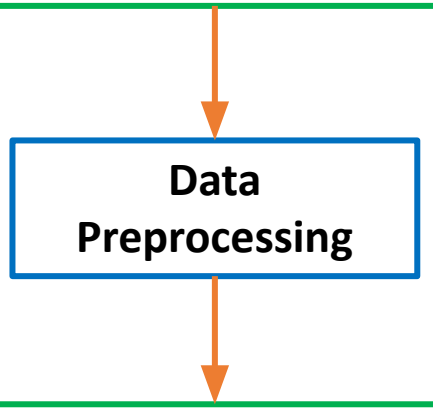
Code Implementation

❖ Coding step: Data preprocessing

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

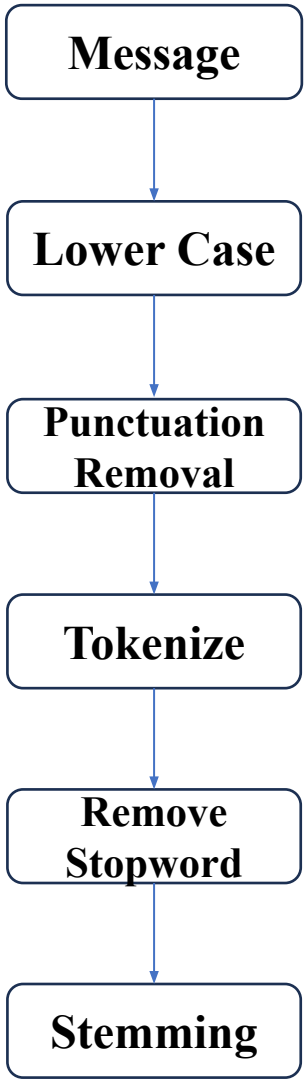
Raw Text: We love this! Would you go? #talk #makememories #unplug #relax #iphone #smartphone #wifi #connect... http://fb.me/6N3LsUpCu



Vector Representation: *array([4, 1, 5, 2, 0, 7, 9])*

Code Implementation

❖ Data Preprocessing Mail Content



"The cat's toys are scattered everywhere!"

Converts all text to lowercase

"the cat's toys are scattered everywhere!"

Eliminates all punctuation marks

"the cats toys are scattered everywhere"

Splits the text into individual words (tokens)

["the", "cats", "toys", "are", "scattered", "everywhere"]

Filters out common words that don't carry significant meaning

["cats", "toys", "scattered", "everywhere"]

Reduces words to their root form, grouping similar words together

["cat", "toy", "scatter", "everywher"]

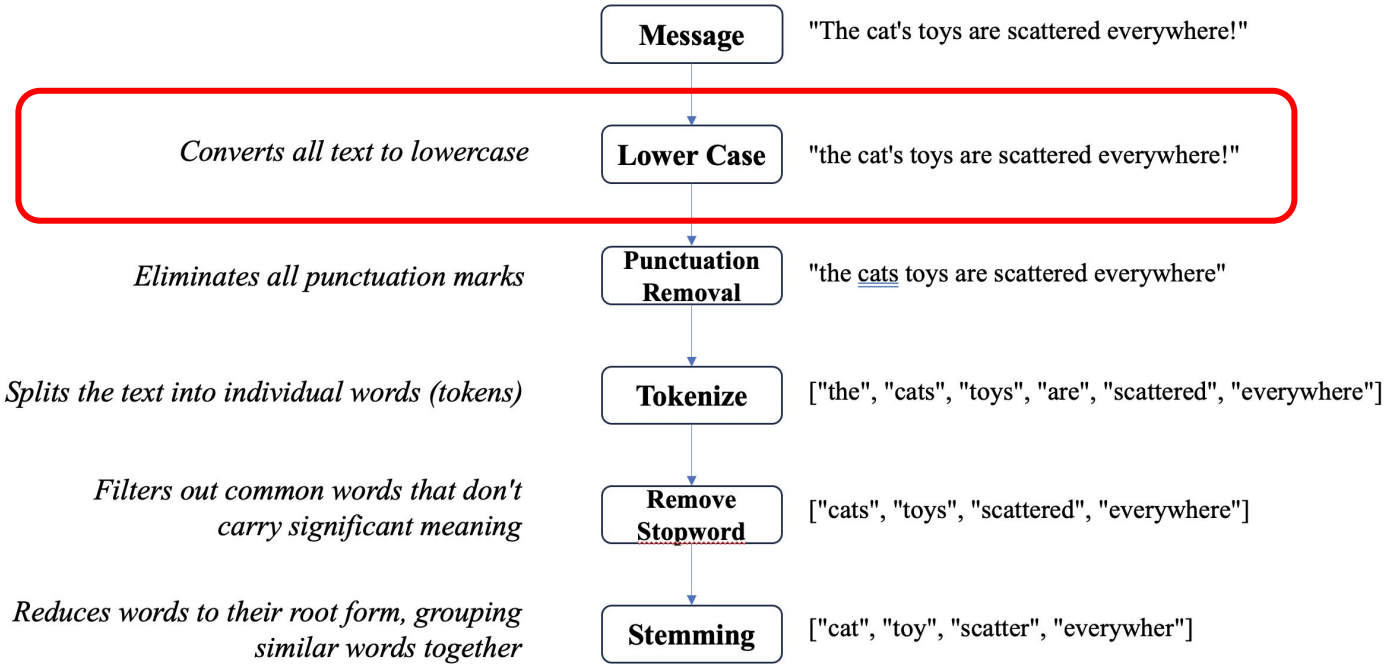
Code Implementation

❖ Data Preprocessing Mail Content

```python
def lowercase(text):
    return text.lower()

INPUT_TEXT = "The cat's toys are scattered everywhere!"
lowercase(INPUT_TEXT)
```

```
'the cat's toys are scattered everywhere!'
```



Code Implementation

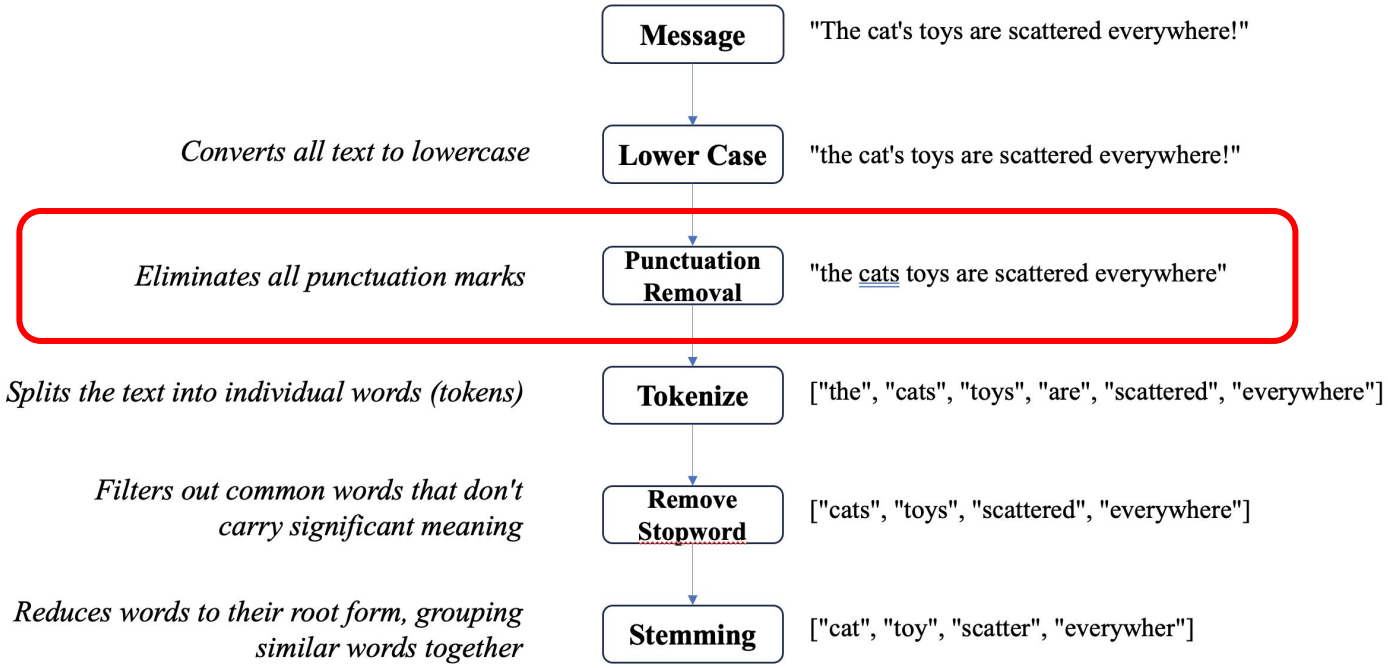
❖ Data Preprocessing Mail Content

```python
def punctuation_removal(text):
    translator = str.maketrans('', '', string.punctuation)

    return text.translate(translator)

INPUT_TEXT = "The cat's toys are scattered everywhere!"
INPUT_TEXT = lowercase(INPUT_TEXT)
punctuation_removal(INPUT_TEXT)
```

```
'the cats toys are scattered everywhere'
```



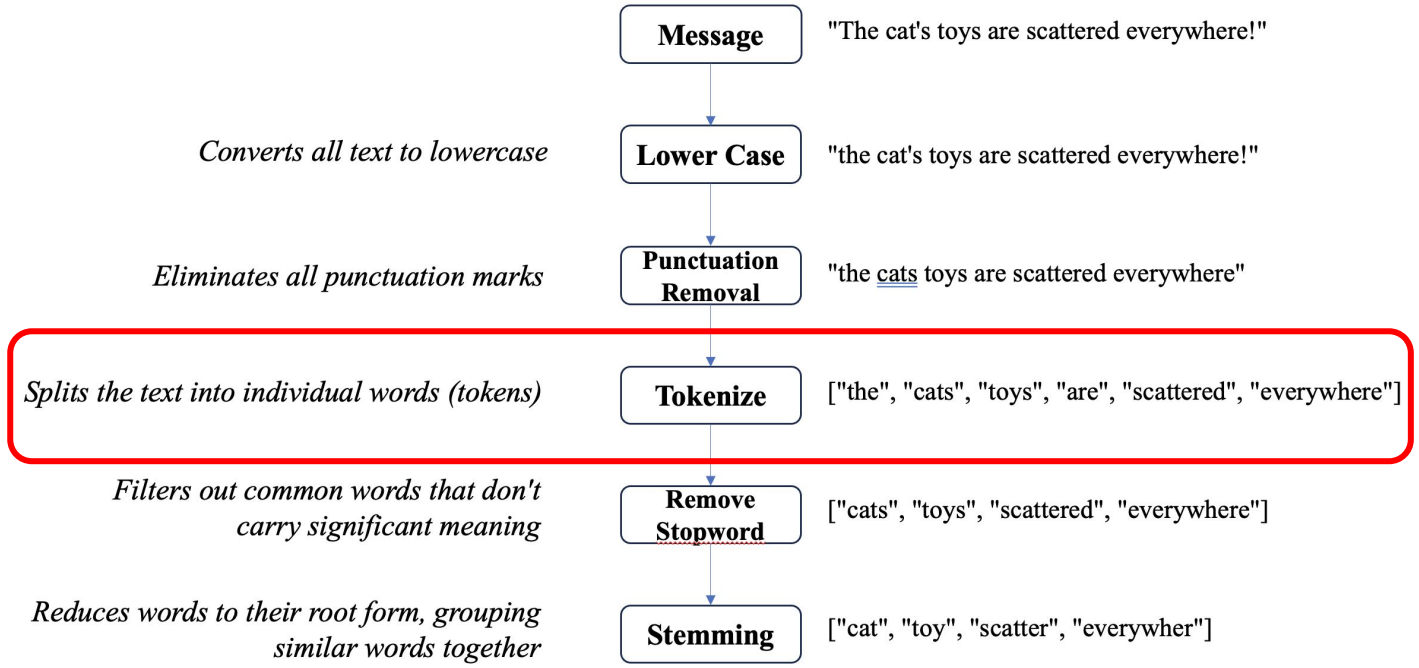
Code Implementation

❖ Data Preprocessing Mail Content

```python
def tokenize(text):
    return nltk.word_tokenize(text)

INPUT_TEXT = "The cat's toys are scattered everywhere!"
INPUT_TEXT = lowercase(INPUT_TEXT)
INPUT_TEXT = punctuation_removal(INPUT_TEXT)
tokenize(INPUT_TEXT)
```

```
['the', 'cats', 'toys', 'are', 'scattered', 'everywhere']
```



Code Implementation

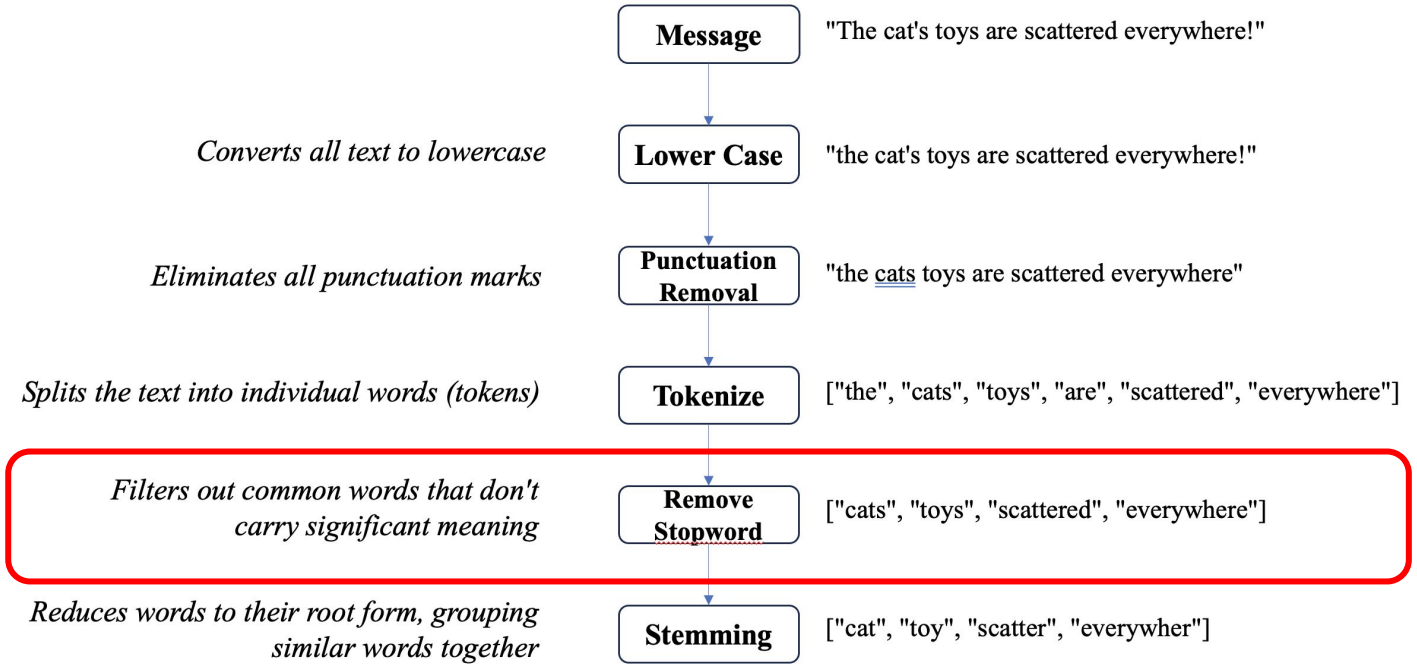
❖ Data Preprocessing Mail Content

```python
def remove_stopwords(tokens):
    stop_words = nltk.corpus.stopwords.words('english')

    return [token for token in tokens if token not in stop_words]

INPUT_TEXT = "The cat's toys are scattered everywhere!"
INPUT_TEXT = lowercase(INPUT_TEXT)
INPUT_TEXT = punctuation_removal(INPUT_TEXT)
INPUT_TEXT = tokenize(INPUT_TEXT)
remove_stopwords(INPUT_TEXT)
```

```
['cats', 'toys', 'scattered', 'everywhere']
```



Code Implementation

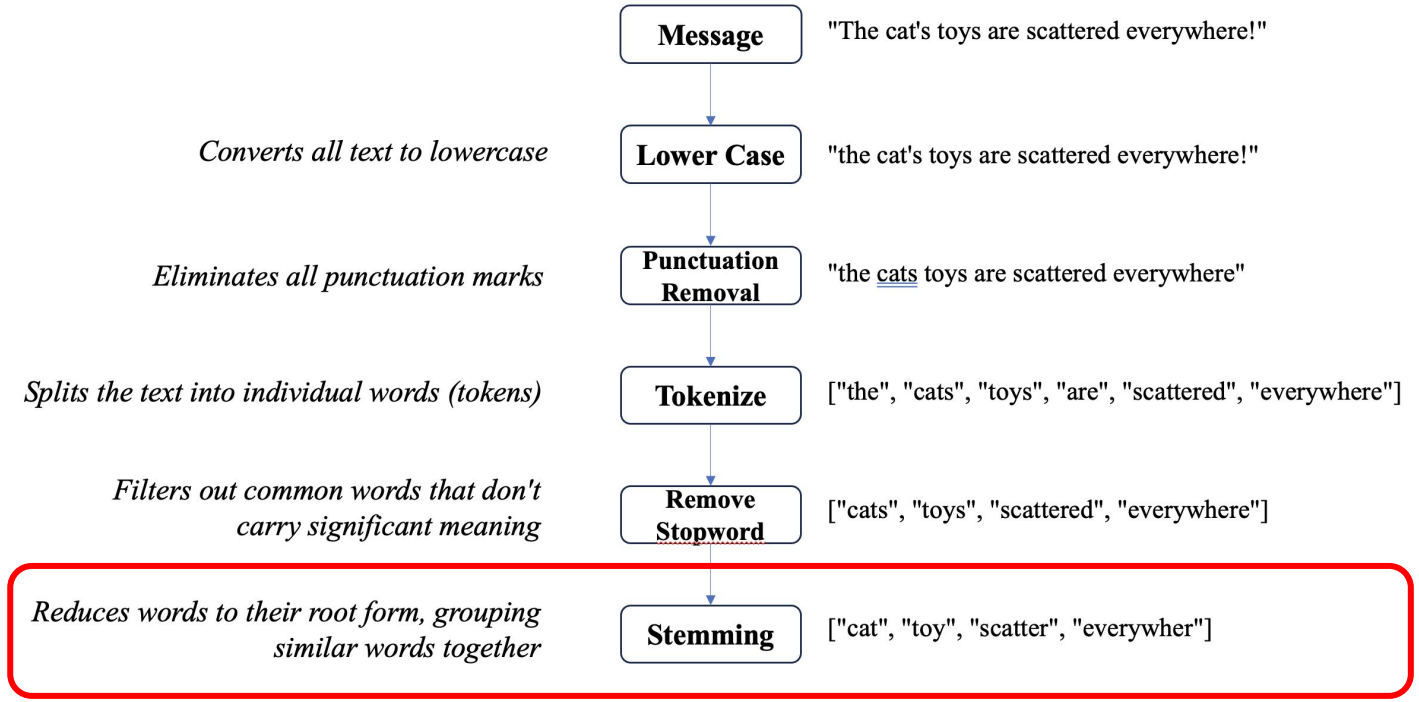
❖ Data Preprocessing Mail Content

```python
def stemming(tokens):
    stemmer = nltk.PorterStemmer()

    return [stemmer.stem(token) for token in tokens]

INPUT_TEXT = "The cat's toys are scattered everywhere!"
INPUT_TEXT = lowercase(INPUT_TEXT)
INPUT_TEXT = punctuation_removal(INPUT_TEXT)
INPUT_TEXT = tokenize(INPUT_TEXT)
INPUT_TEXT = remove_stopwords(INPUT_TEXT)
stemming(INPUT_TEXT)
```

```
['cat', 'toy', 'scatter', 'everywher']
```



Code Implementation

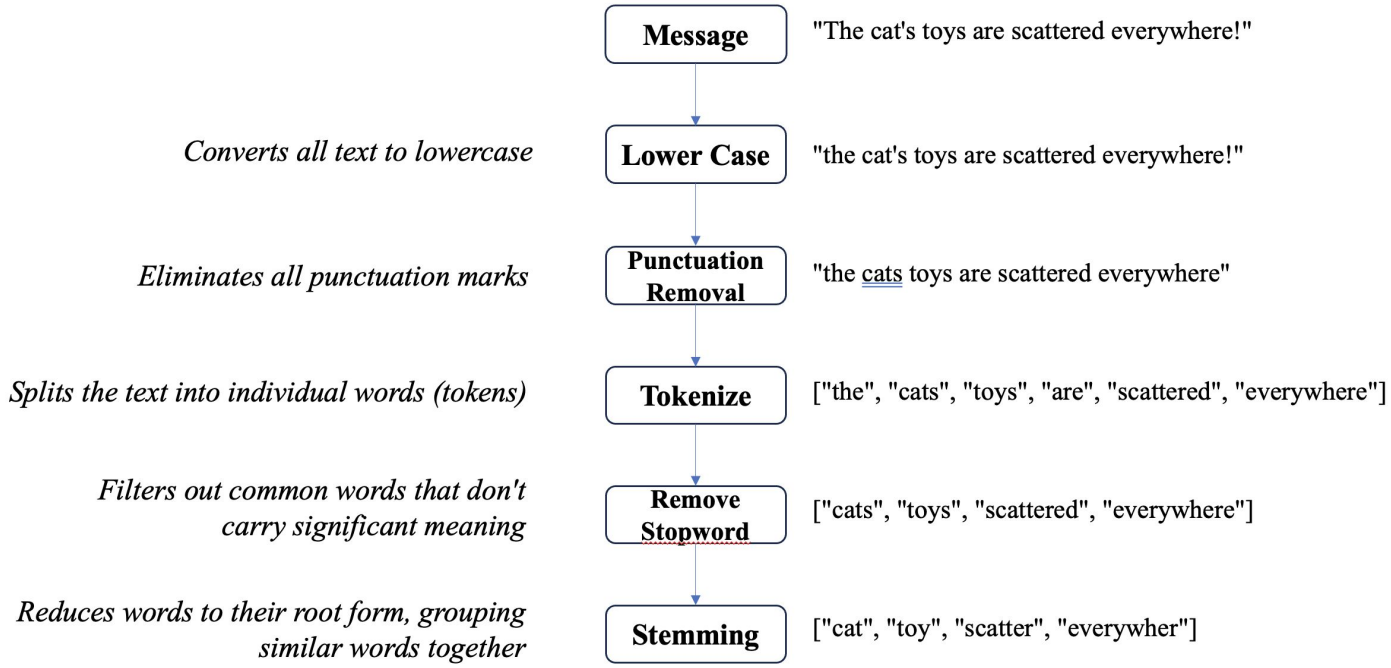
❖ Data Preprocessing Mail Content

```python
def preprocess_text(text):
    text = lowercase(text)
    text = punctuation_removal(text)
    tokens = tokenize(text)
    tokens = remove_stopwords(tokens)
    tokens = stemming(tokens)

    return tokens

INPUT_TEXT = "The cat's toys are scattered everywhere!"
preprocess_text(INPUT_TEXT)
```

```
['cat', 'toy', 'scatter', 'everywher']
```



Code Implementation

❖ Data Preprocessing Mail Content

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

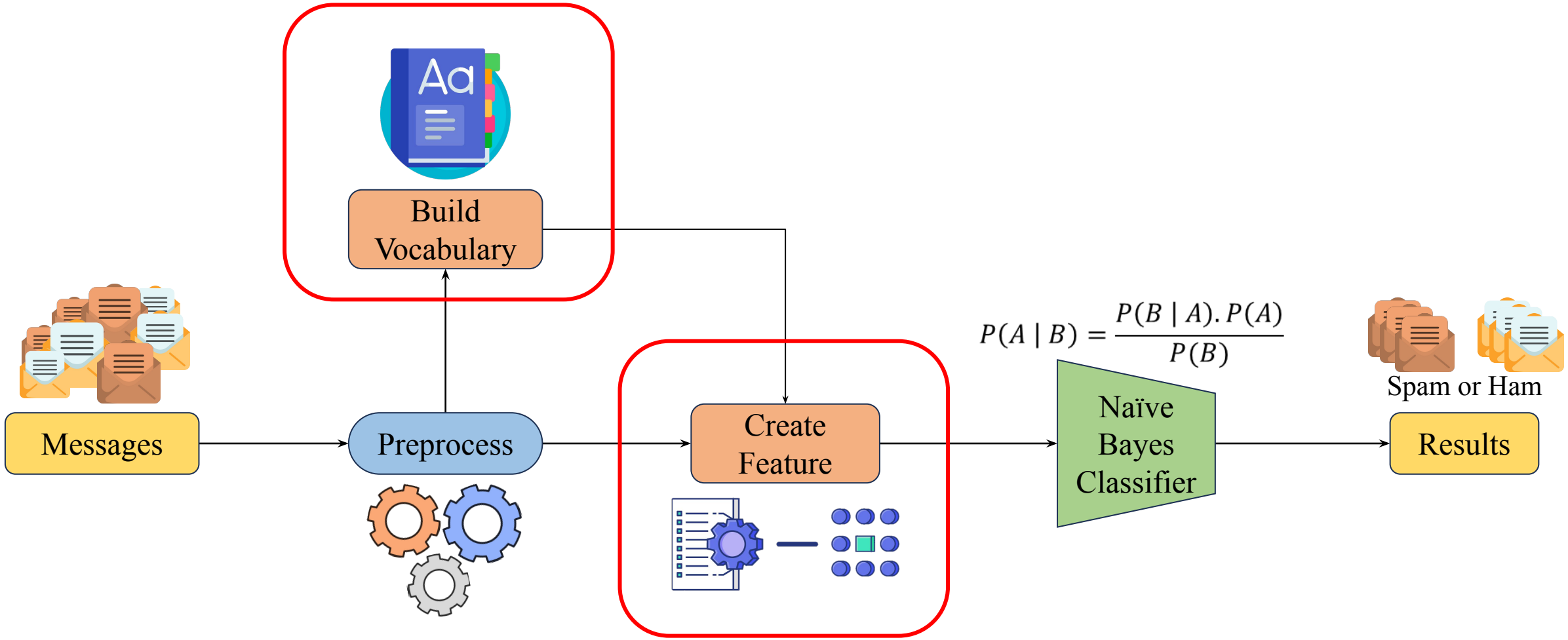
```
1 messages = [preprocess_text(message) for message in messages]
```

```
1 messages
```

```
[['go',
  'jurong',
  'point',
  'crazi',
  'avail',
  'bugi',
  'n',
  'great',
  'world',
  'la',
  'e',
  'buffet',
  'cine',
  'got',
  'amor',
  'wat'],
 ['ok', 'lar', 'joke', 'wif', 'u', 'oni'],
```

Code Implementation

❖ Project Pipeline



Code Implementation

❖ Build Dictionary

Preprocess

Build Vocabulary

*builds a vocabulary (dictionary) from all **unique** tokens across all mails*

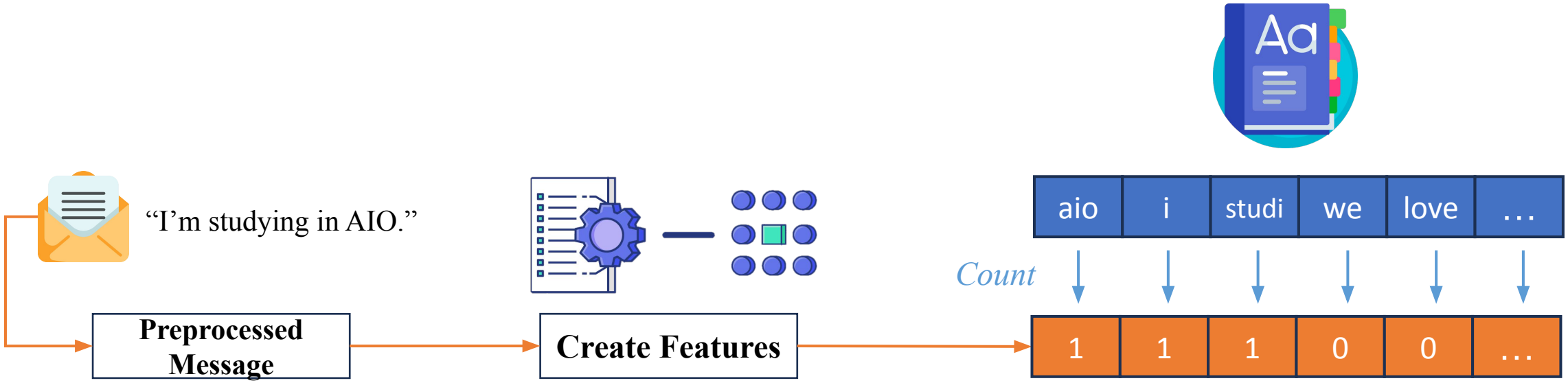
Preprocess

unique

aio	i	studi	we	love	…

Code Implementation

❖ Create Features



counts how many times each word from the Vocabulary appears in a message

Code Implementation

❖ Coding step: Build vocab

```python
def create_dictionary(messages):
    dictionary = []

    for tokens in messages:
        for token in tokens:
            if token not in dictionary:
                dictionary.append(token)

    return dictionary

dictionary = create_dictionary(messages)
dictionary
```

```
['go',
 'jurong',
 'point',
 'crazi',
 'avail',
 'bugi',
 'n',
 'great',
 'world',
```

Build Vocabulary

*builds a vocabulary (dictionary) from all **unique** tokens across all mails*

Code Implementation

❖ Coding step: Create features

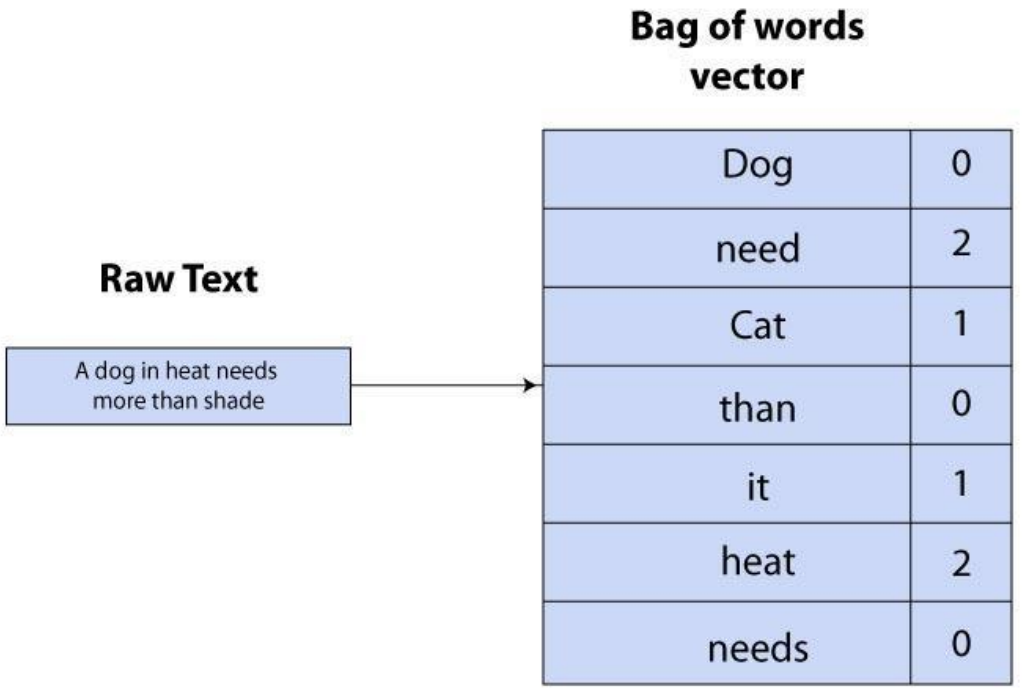
```python
def create_features(tokens, dictionary):
    features = np.zeros(len(dictionary))

    for token in tokens:
        if token in dictionary:
            features[dictionary.index(token)] += 1

    return features

dictionary = create_dictionary(messages)
X = np.array([create_features(tokens, dictionary) for tokens in messages])
print(X.shape)
print(X)
```

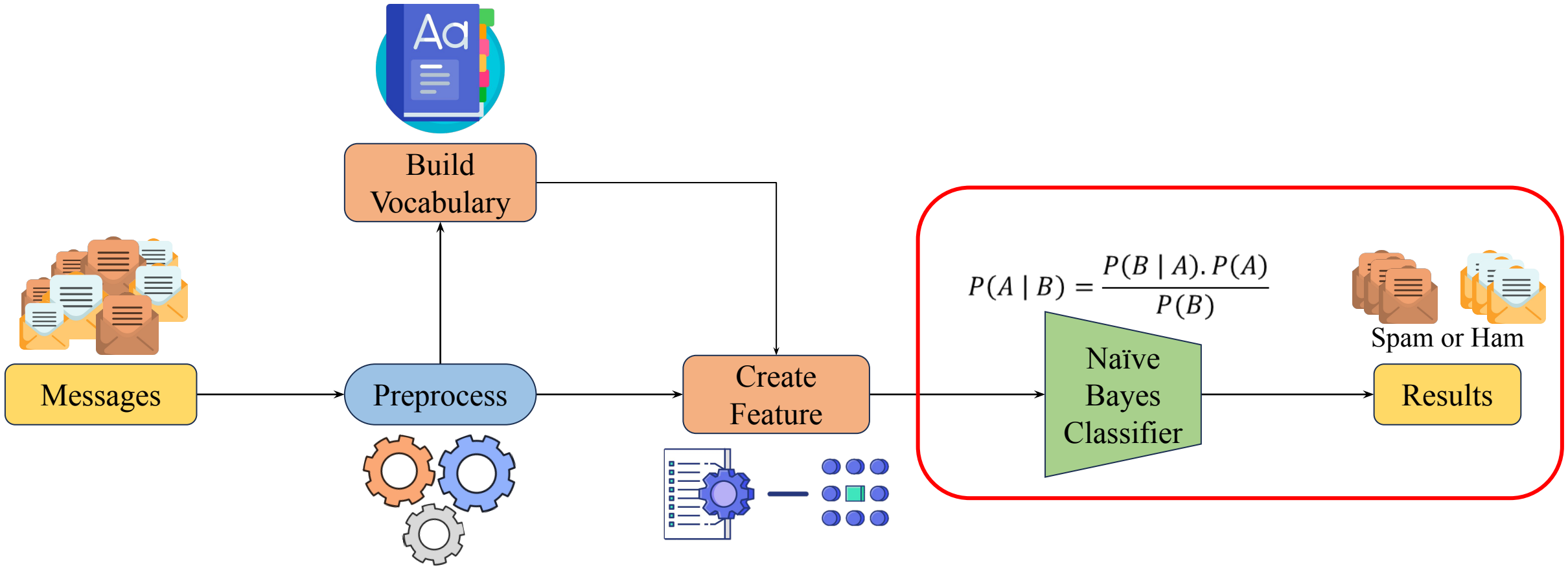
```
(5572, 8166)
[[1. 1. 1. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 1. 1.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```



https://www.analyticssteps.com/blogs/an-optimum-approach-towards-the-bag-of-words-with-code-illustration-in-python

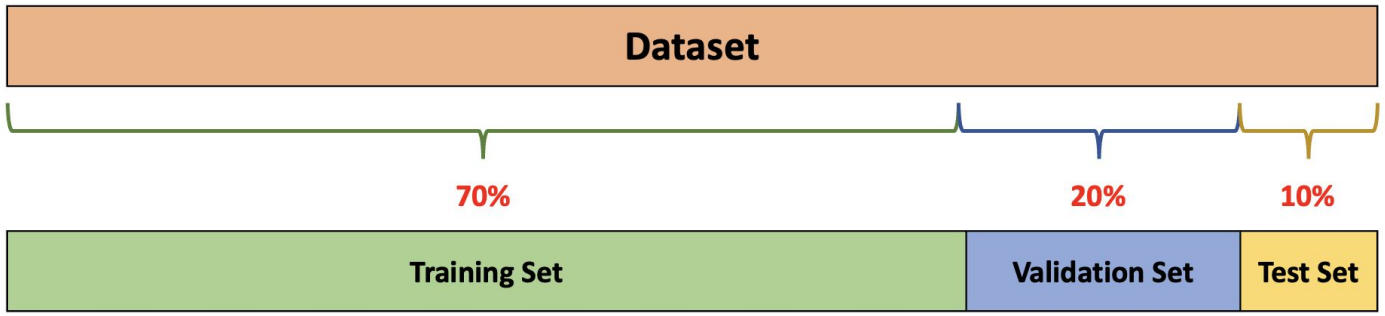
Code Implementation

❖ Project Pipeline



Code Implementation

❖ Coding step: Split train, val, test set




```python
VAL_SIZE = 0.2
TEST_SIZE = 0.125
SEED = 0
IS_SHUFFLE = True

X_train, X_val, y_train, y_val = train_test_split(
    X, y,
    test_size=VAL_SIZE,
    shuffle=IS_SHUFFLE,
    random_state=SEED
)
X_train, X_test, y_train, y_test = train_test_split(
    X_train, y_train,
    test_size=TEST_SIZE,
    shuffle=IS_SHUFFLE,
    random_state=SEED
)
```

```python
print(f'Number of training samples: {X_train.shape[0]}')
print(f'Number of val samples: {X_val.shape[0]}')
print(f'Number of test samples: {X_test.shape[0]}')
```

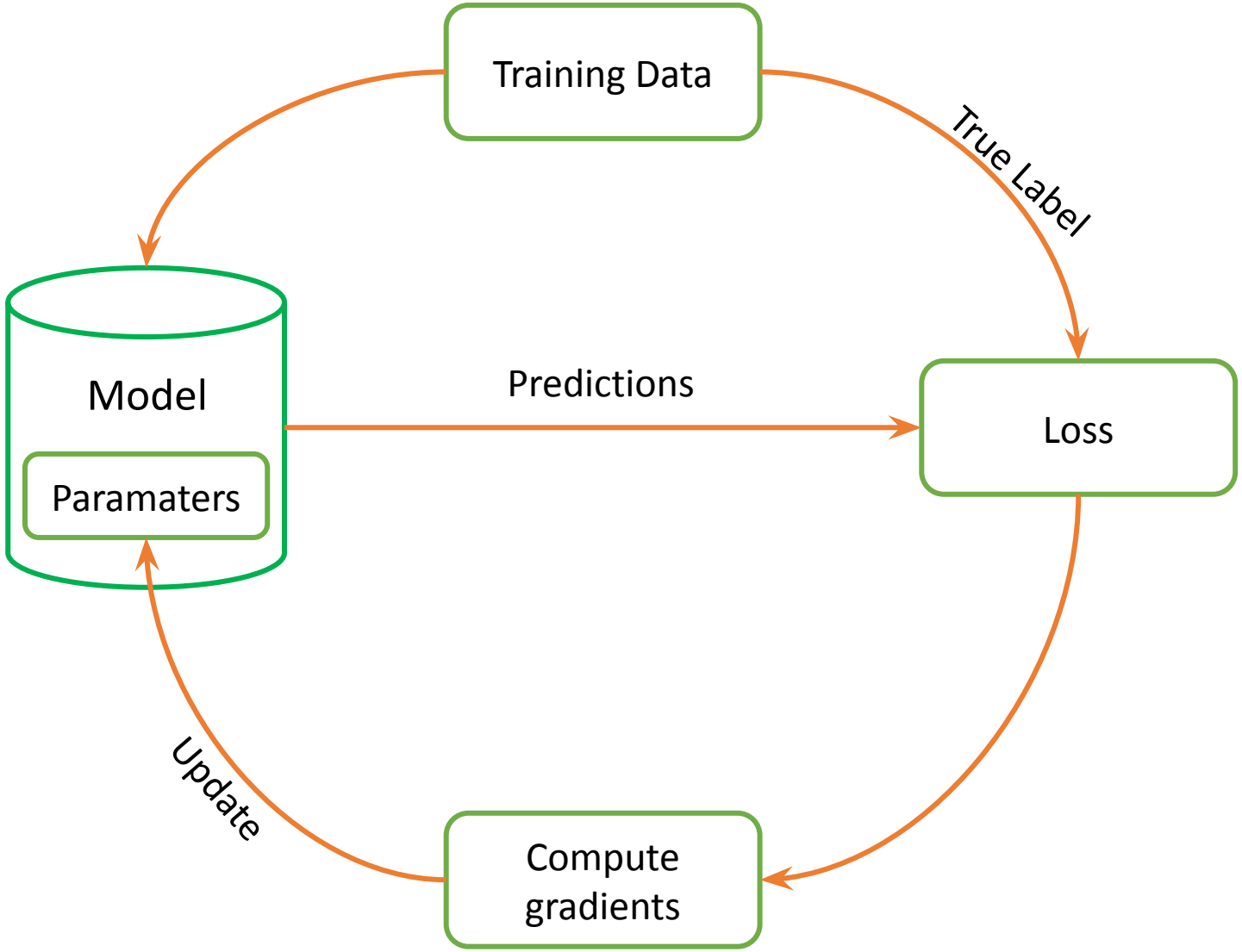
```
Number of training samples: 3899
Number of val samples: 1115
Number of test samples: 558
```

Code Implementation

❖ Coding step: Training model

```
1 %%time
2 model = GaussianNB()
3 print('Start training...')
4 model = model.fit(X_train, y_train)
5 print('Training completed!')
```

```
Start training...
Training completed!
CPU times: user 397 ms, sys: 162 ms, total: 559 ms
Wall time: 633 ms
```



Code Implementation

❖ Coding step: Evaluation

$$accuracy = \frac{true\_predictions}{n\_samples}$$

```
1 y_val_pred = model.predict(X_val)
2 y_test_pred = model.predict(X_test)
3 val_accuracy = accuracy_score(y_val, y_val_pred)
4 test_accuracy = accuracy_score(y_test, y_test_pred)
5 print(f'Val accuracy: {val_accuracy}')
6 print(f'Test accuracy: {test_accuracy}')
```

```
Val accuracy: 0.8816143497757848
Test accuracy: 0.8602150537634409
```

Code Implementation

❖ Coding step: Prediction

```python
def predict(text, model, dictionary):
    processed_text = preprocess_text(text)
    features = create_features(text, dictionary)
    features = np.array(features).reshape(1, -1)
    prediction = model.predict(features)
    prediction_cls = le.inverse_transform(prediction)[0]

    return prediction_cls
```

```python
test_input = 'I am actually thinking a way of doing something useful'
prediction_cls = predict(test_input, model, dictionary)
print(f'Prediction: {prediction_cls}')
```

```
Prediction: ham
```

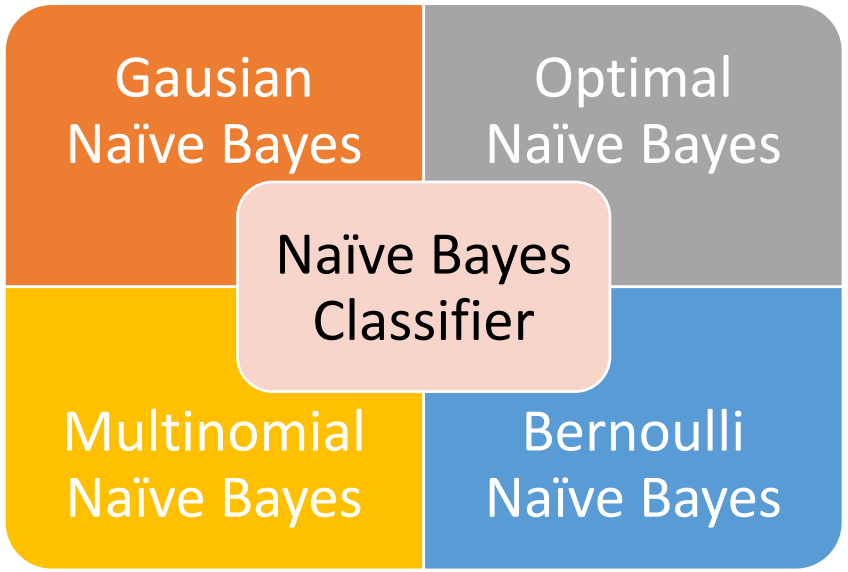
QUIZ

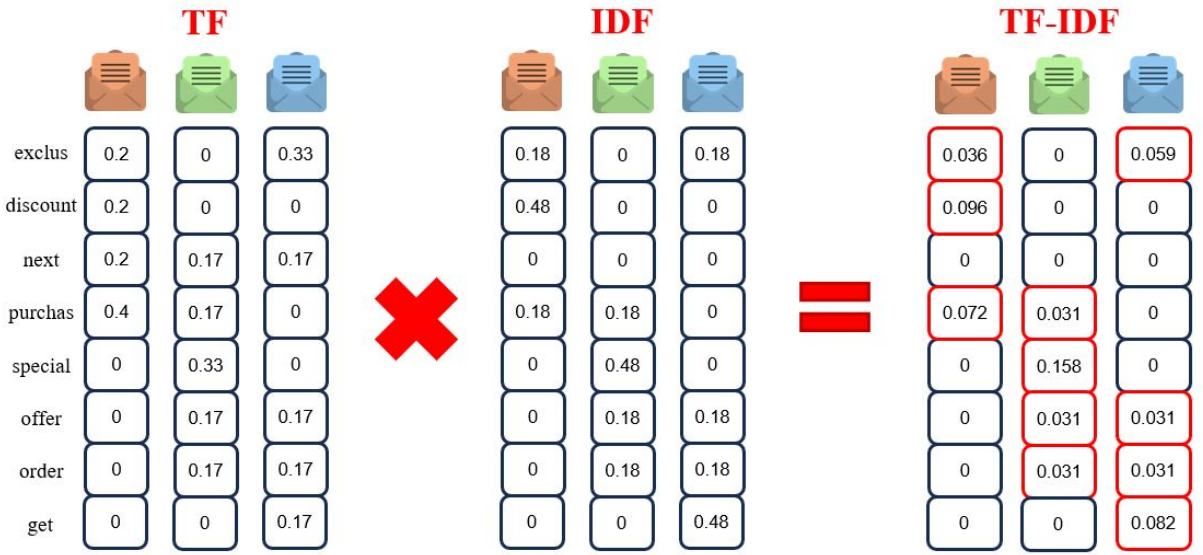
Improved Baseline

Improved Baseline

Introduction

1. Use other Naive Bayes variables.
2. Use advanced feature extraction techniques such as TF-IDF, BoW, N-grams, …





Improved Baseline

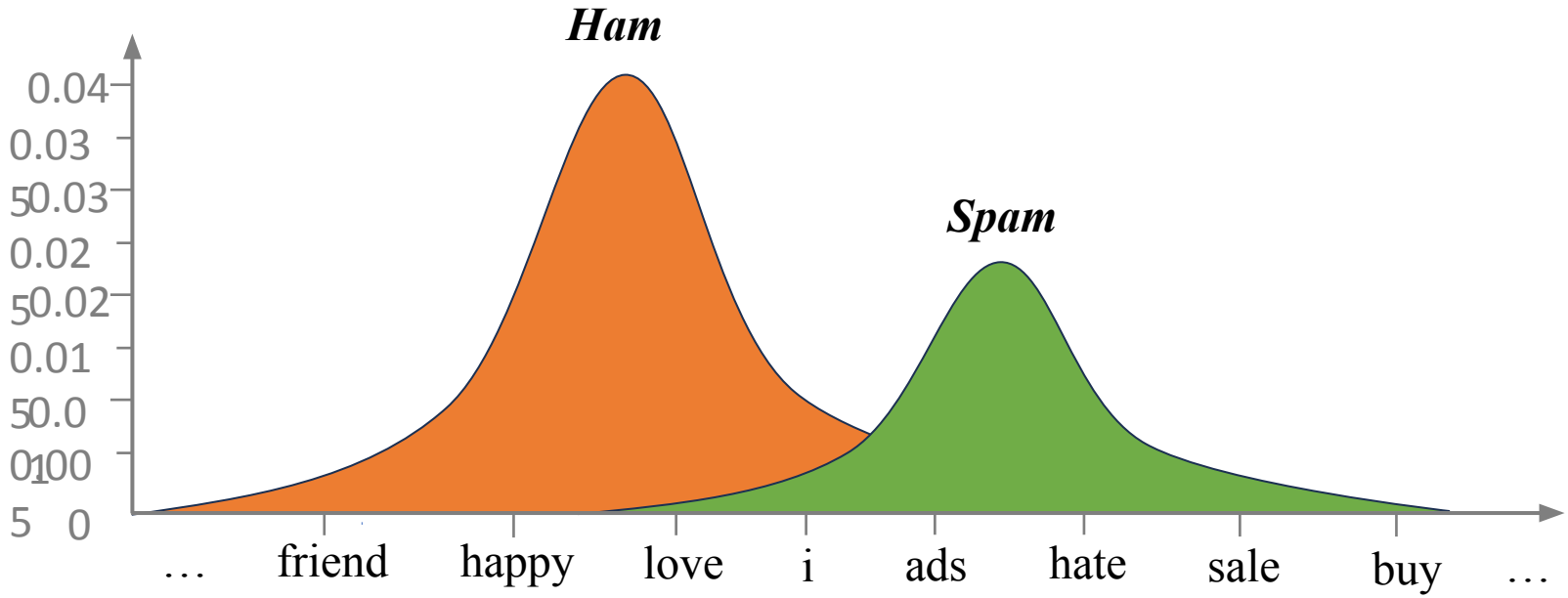
1. Different Naïve Bayes Classifier

Gaussian Naïve Bayes Model

The different naive Bayes classifiers differ mainly by the assumptions $P(x_i \mid y)$.

Gaussian Naive Bayes assumes that:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right), \text{ where mean } \mu \text{ and var } \sigma \text{ are estimated.}$$

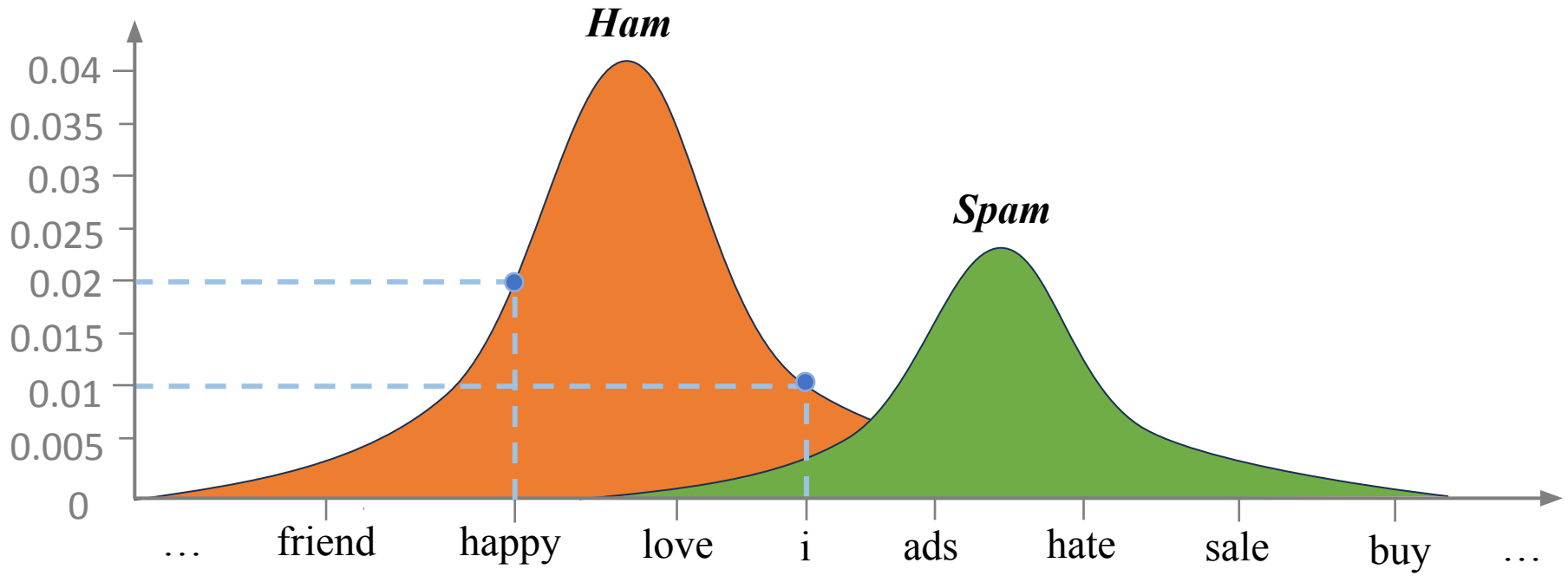


Improved Baseline

1. Different Naïve Bayes Classifier

New Message **A**: *"I am happy"*

$$P(\boldsymbol{A} \mid Ham) = P("i"|Ham).\ P("happy"|Ham) = 0.01 * 0.02 = 0.002$$

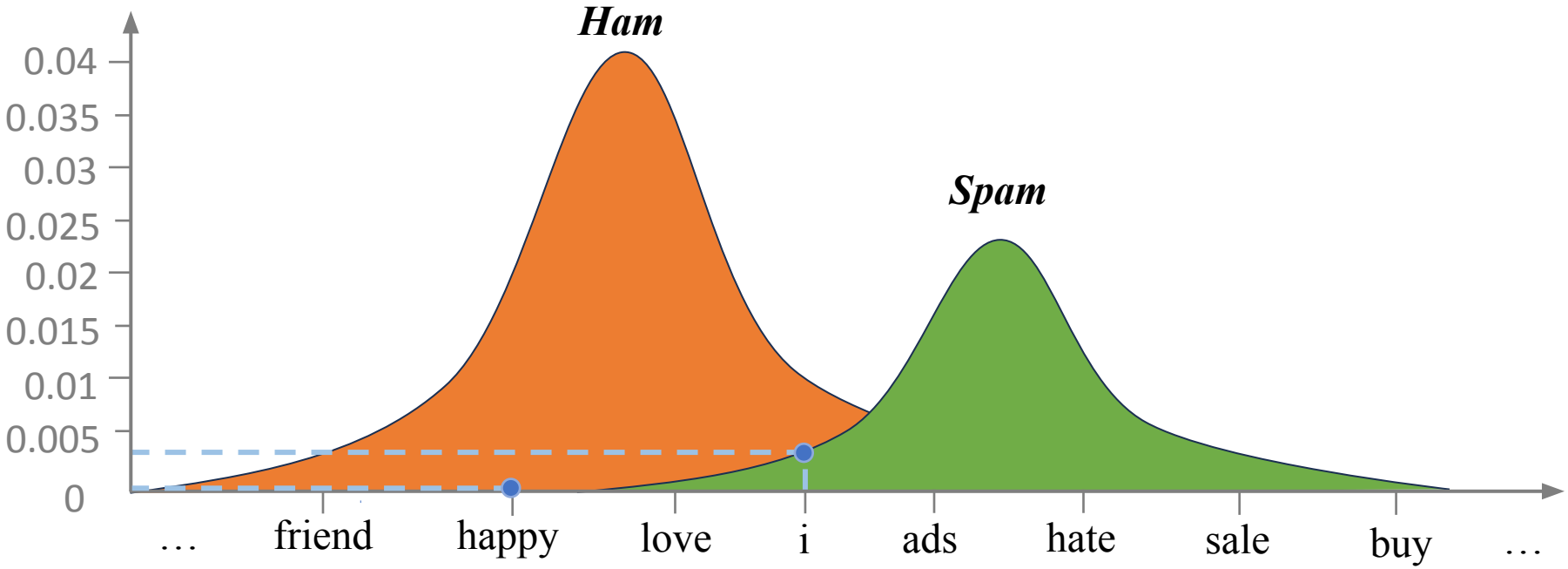


Improved Baseline

1. Different Naïve Bayes Classifier

New Message **A**: *"I am happy"*

$$P(\boldsymbol{A} \mid Spam) = P("i"|Spam).\ P("happy"|Spam) = 0.003 * 0.00001 = 0.0000003$$

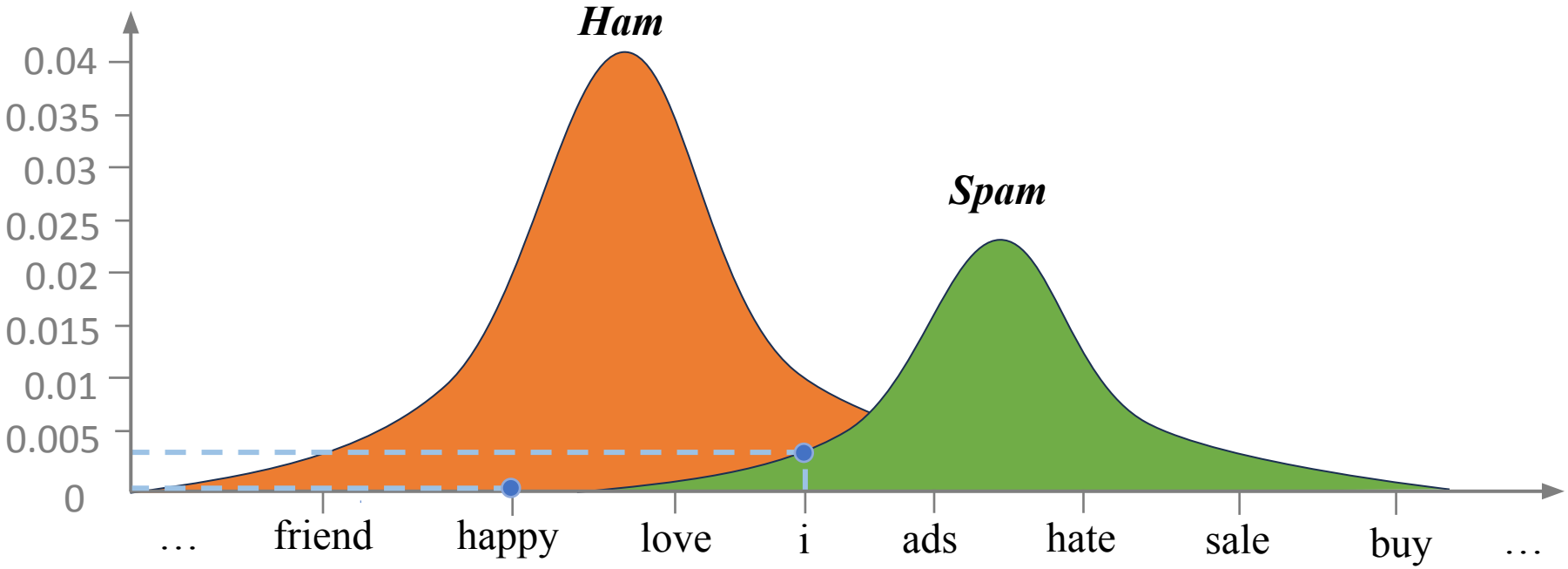


Improved Baseline

1. Different Naïve Bayes Classifier

New Message **A**: *"I am happy"*

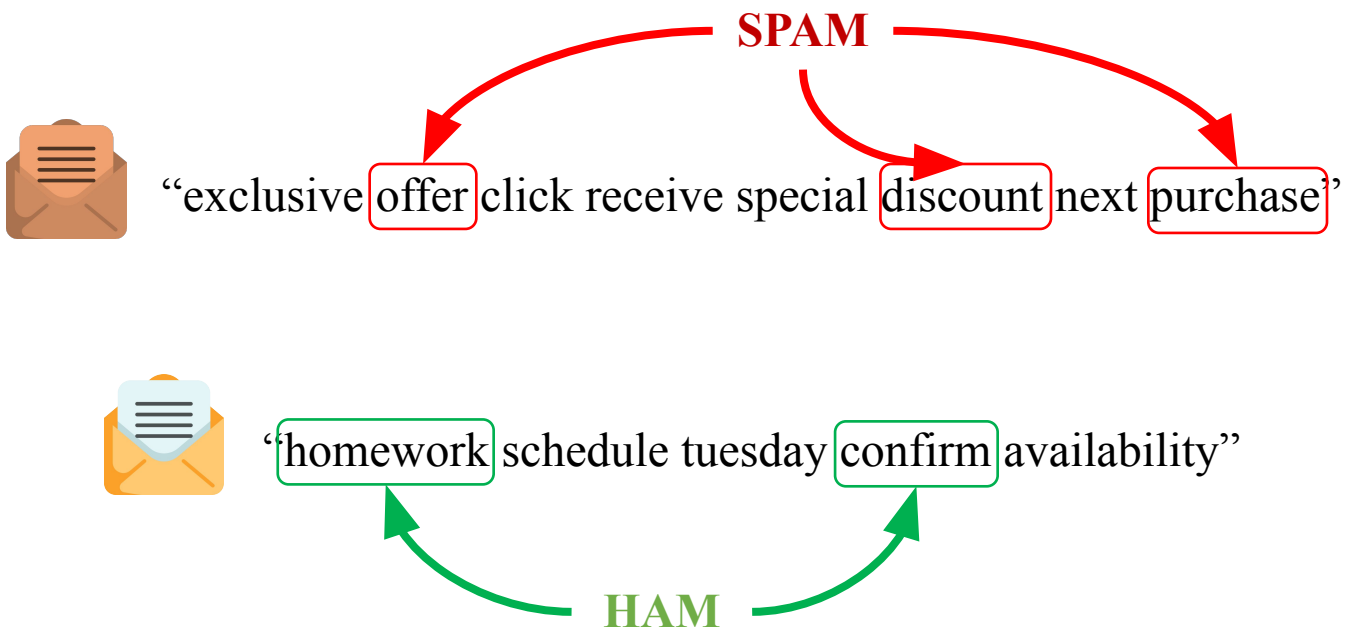
$$P(\boldsymbol{A} \mid Spam) = P("i"|Spam).\ P("happy"|Spam) = 0.003 * 0.00001 = 0.0000003$$



Improved Baseline

2. Different Feature Extraction Techniques

TF-IDF (Term Frequency-Inverse Document Frequency), measure used to evaluate the importance of a word in a document relative to a collection of documents (corpus), to identify words that are significant in a specific document but not too common across all documents.



Improved Baseline

❖ 2. Different Feature Extraction Techniques

Term Frequency measures how frequently a word appears in a document.

$$TF = \frac{Number\ of\ times\ \boldsymbol{word_i}\ appears\ in\ \boldsymbol{mail}}{Total\ number\ of\ \boldsymbol{words}\ in\ \boldsymbol{mail}}$$


"exclus discount next purchas purchas"


"special offer next purchas special order"


"get exclus offer next exclus order"



exclus	0.2 = 1/5	0	0.33
discount	0.2	0	0
next	0.2	0.17	0.17
purchas	0.4 = 2/5	0.17	0
special	0	0.33	0
offer	0	0.17	0.17
order	0	0.17	0.17
get	0	0	0.17

Improved Baseline

❖ 2. Different Feature Extraction Techniques

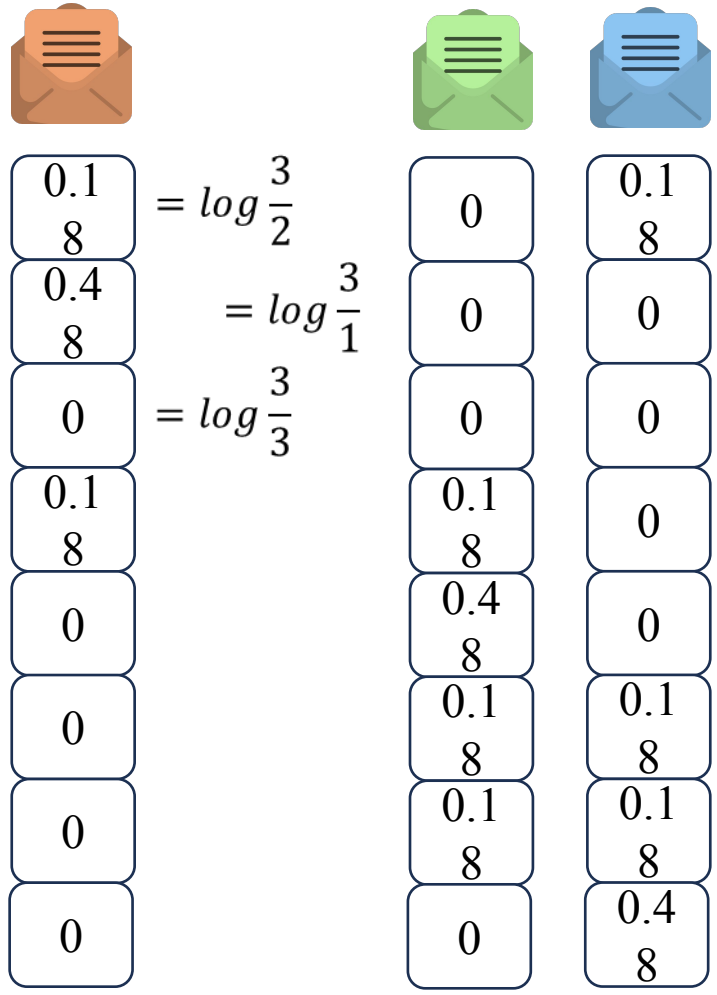
Inverse Document Frequency measures how important a word is across all mail.

$$IDF = log \frac{Number\ of\ \textbf{mails}}{Number\ of\ \textbf{mails containing word}_i}$$


"exclus discount next purchas purchas"


"special offer next purchas special order"

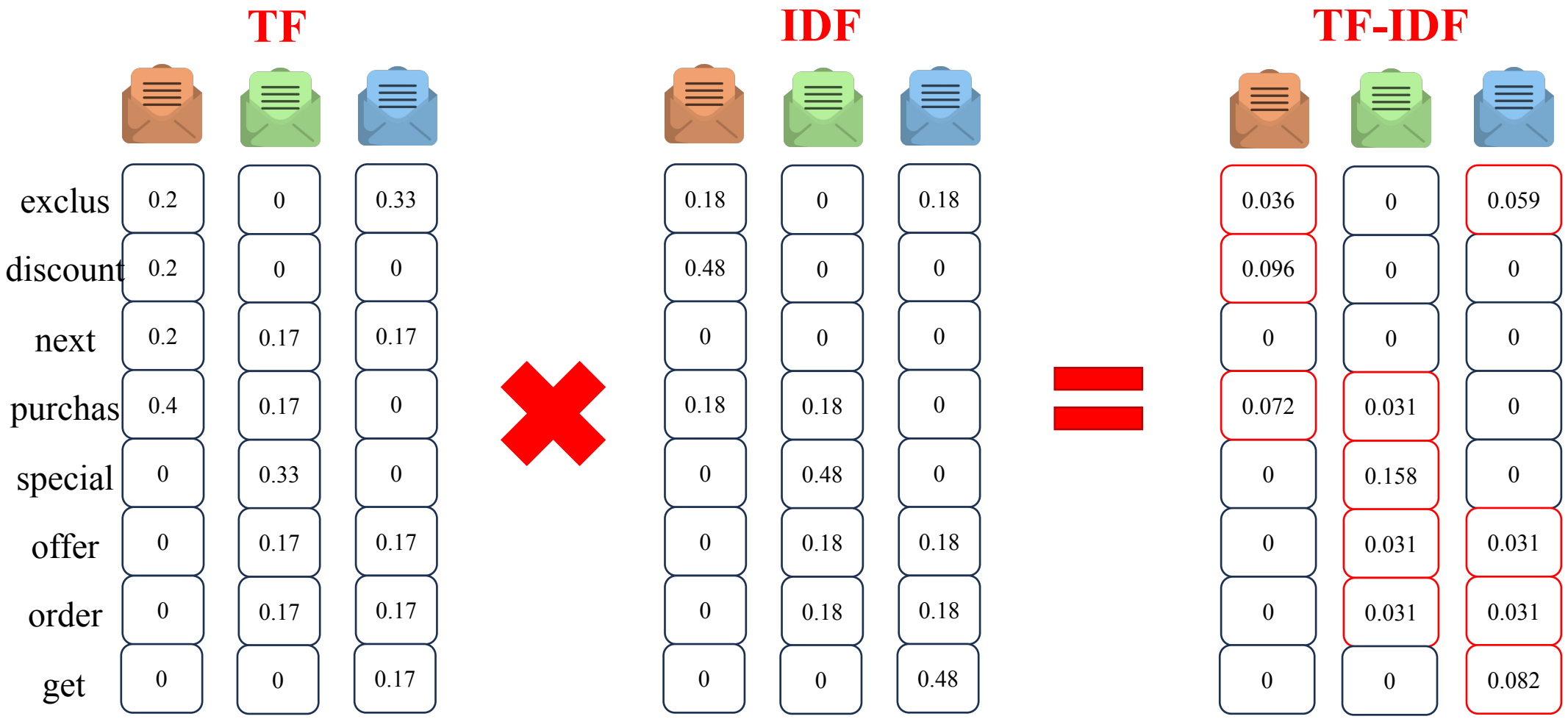

"get exclus offer next exclus order"



Improved Baseline

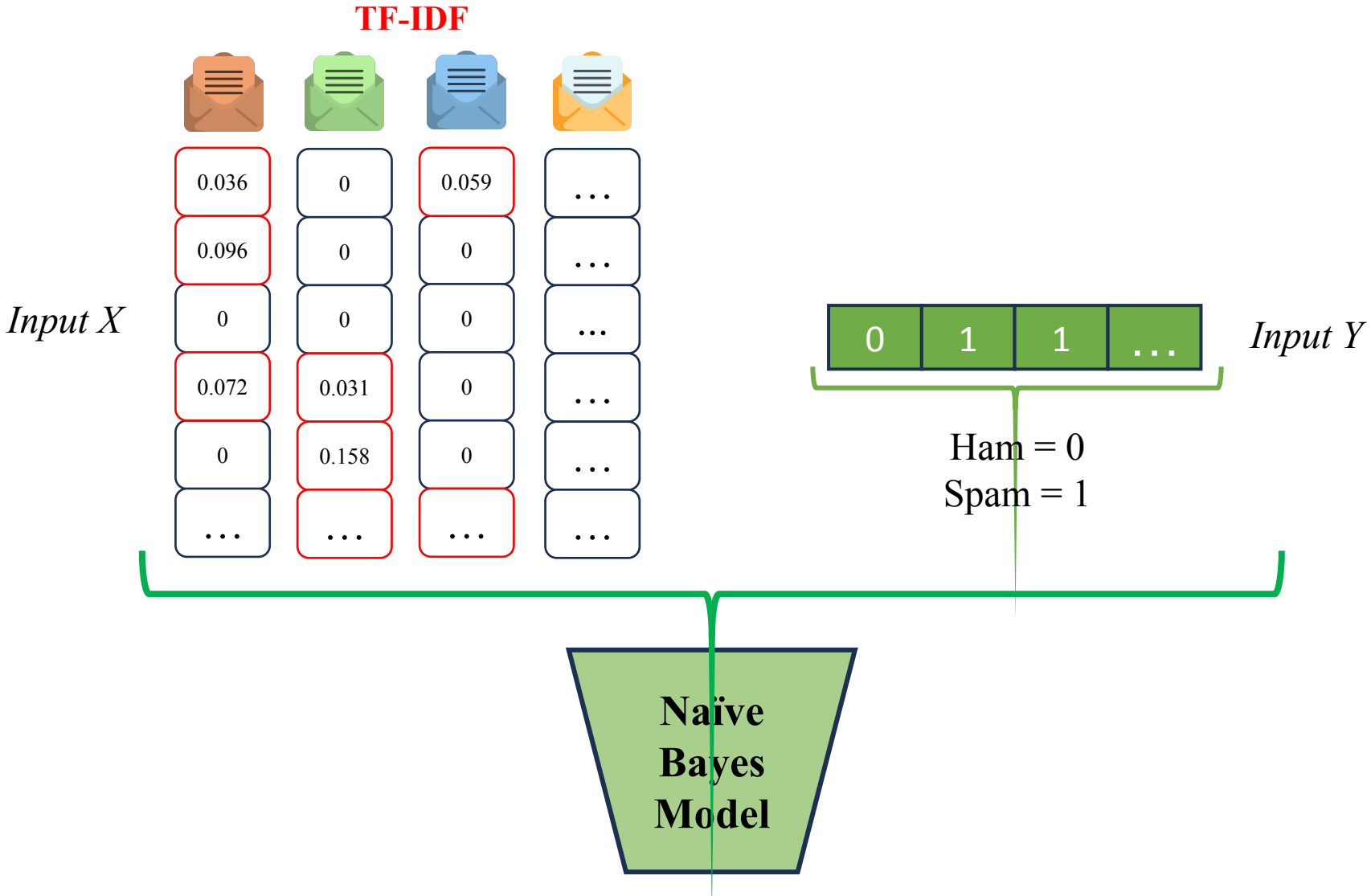
2. Different Feature Extraction Techniques

Combines TF and IDF to determine the importance of a word in a mail relative to the entire mails.



Improved Baseline

❖ Input Naïve Bayes Model



Summarization

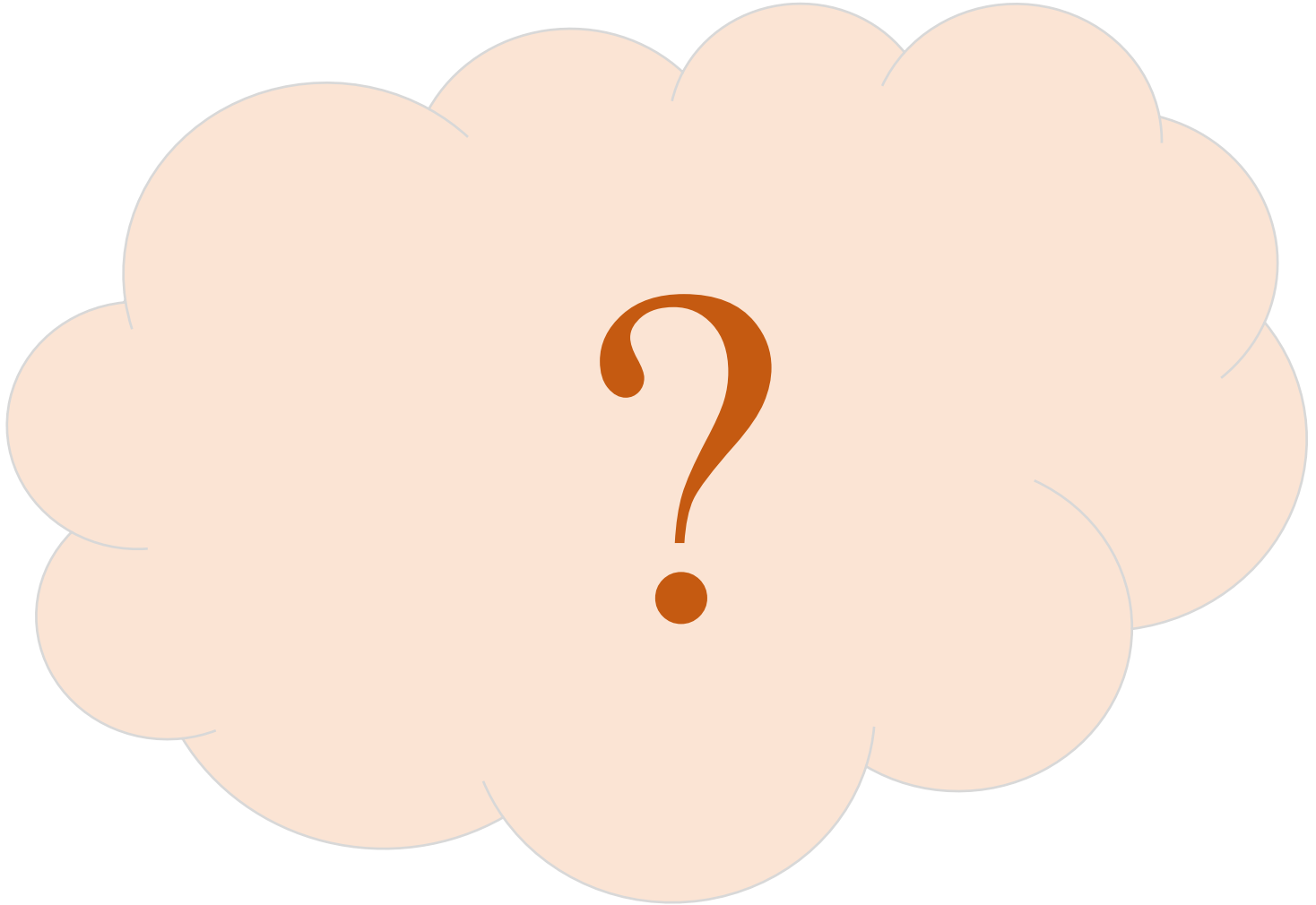
❖ What we have learned so far

Spam	Not spam
	
"SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info"	"Nah I don't think he goes to usf, he lives around here though"

Summarization:

- Discussed about Naïve Bayes algorithm.
- Discussed how to apply Naïve Bayes algorithm to solve the task of Sentiment Analysis in NLP.
- Discussed on an improved baseline to outperform Naïve Bayes.

Question




Thank you!