VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY

UIT

TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

PROJECT REPORT

SUBJECT: COMPUTATIONAL THINKING

TOPIC

CLASSIFICATION HATE SPEECH ON SOCIAL NETWORK FACEBOOK

Instructor guide:	TS. Ngô Đức Thành
Students:	Đỗ Trọng Khánh - 19521676 Võ Phạm Duy Đức – 19521383 Trịnh Công Danh - 19521326
Class:	CS117.L22.KHCL

Ho Chi Minh City, day 21 month 7 year 2021

TABLE OF CONTENTS

I. Description of the problem

1. Introduction to the problem

- According to a newly published survey of Microsoft, Vietnam is in the top 5 countries with the lowest civility index in cyberspace (DCI).

- Although this survey result is detrimental to the image of the Vietnamese online community, it does not create a wave of protests from users. According to a survey on Zing.vn, 87% of readers agree with Microsoft's ranking of Vietnam in the top 5 countries that behave poorly on the Internet.

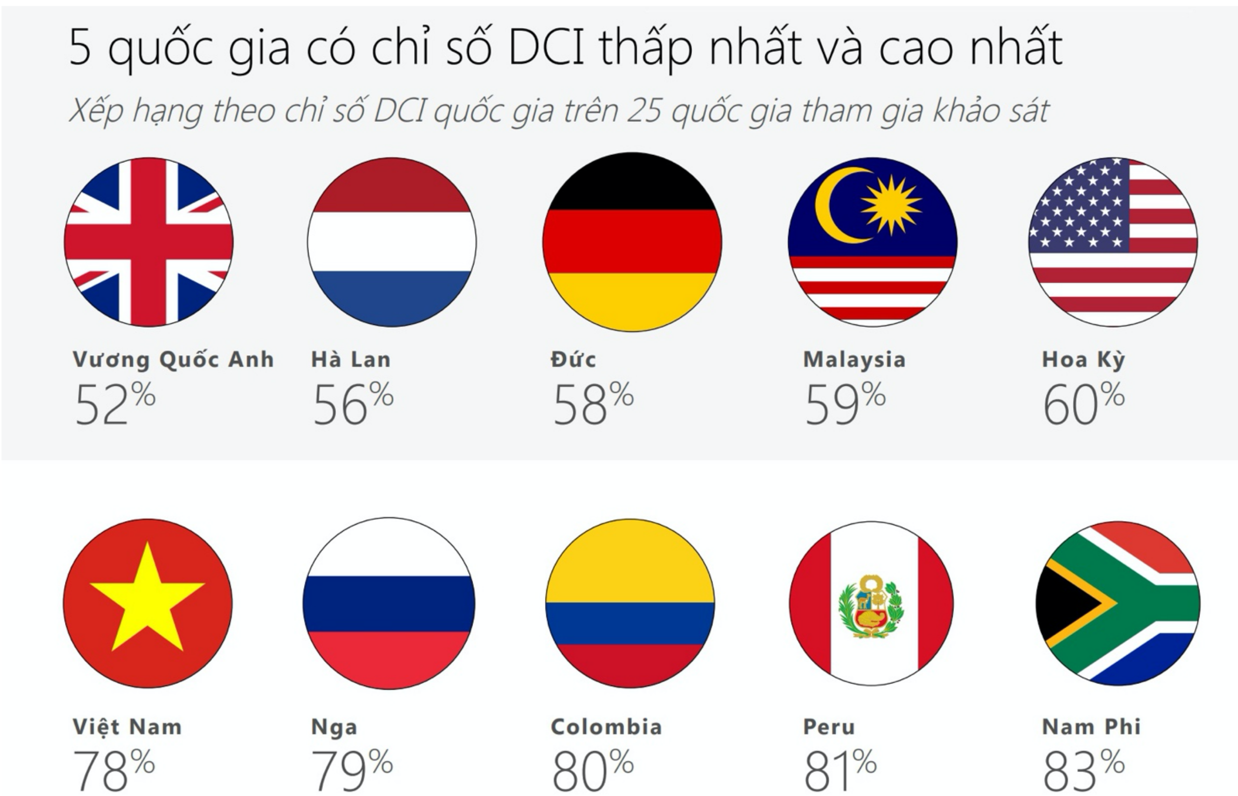


Image 1: Image from Zingnews.vn

- Since then, the group has selected articles that detect vulgar and offensive comments from the comments. The problem of detecting vulgar and offensive comments on the most popular social networking platform today is Facebook.

2. Description of input and output

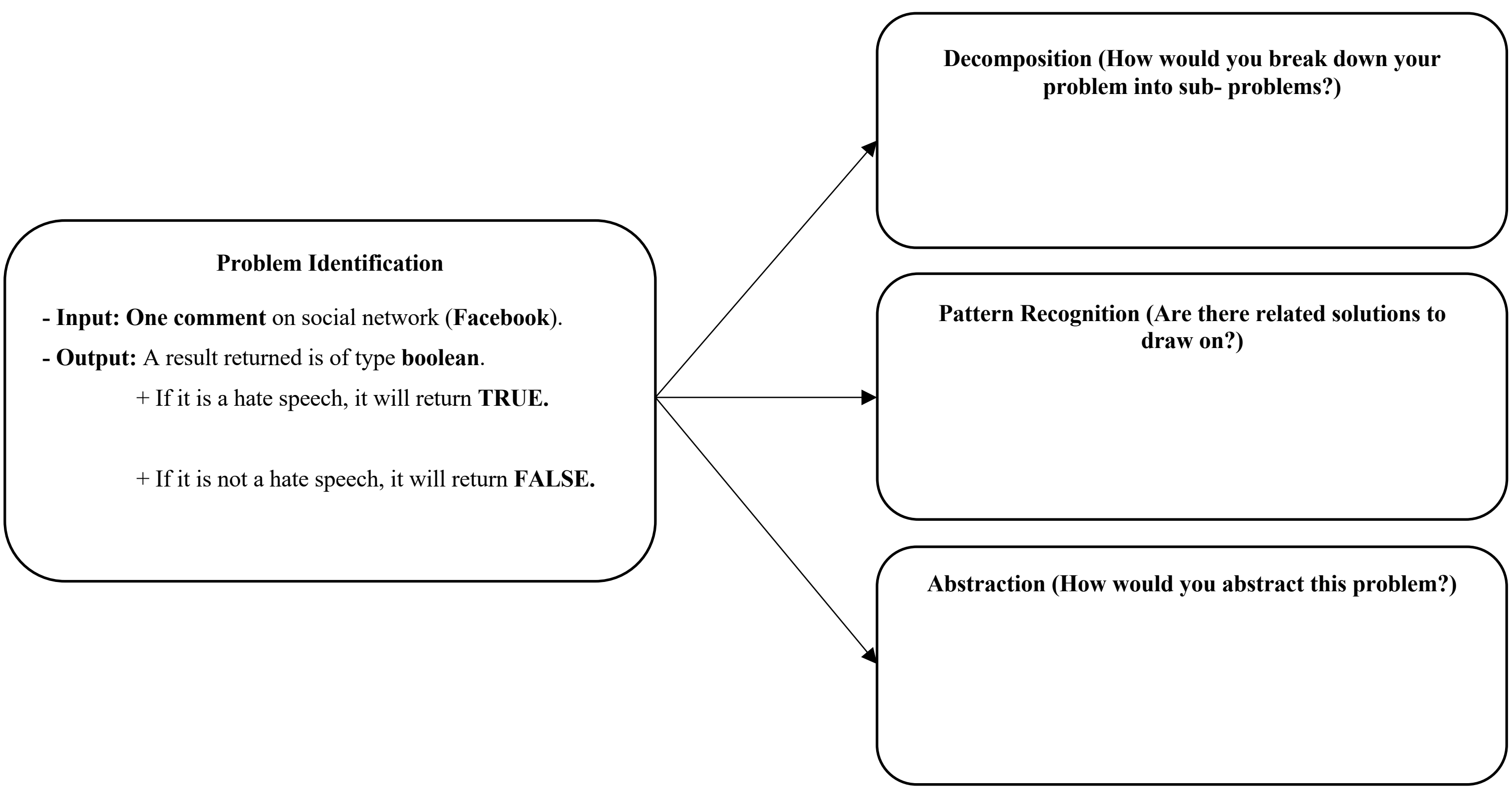
- **Input:** One hate speech **(comments)** on social network (**Facebook**).
- **Output:** A result returned is of type **boolean**.
    + If it is a hate speech, it will return **TRUE**.
    + If it is not hate speech, it will return **FALSE**.

3. Model evaluation

- **Data Availability**: The group will crawl comments on the social network **(facebook).** Label comment as **1** with **hate speech** and label comment as **0** with **not hate speech**.
- **Accuracy** is evaluated based on **the number of correctly predicted comments / predicted number of comments**.

II. Graphic Organizer

1. Interation 1



2. Interation 2

Problem Identification

- **Input: One comment** on social network (**Facebook**) in text format, **Vietnamese** language. The length of the comment sentence should not exceed **200 words**.
- **Output:** A result returned is of type **boolean**.
    + If it is a **hate speech** (Hate speech is public speech that expresses hate or encourages violence towards a person or group base on something such as race, religion, sex, or sexual orientation), it will return **TRUE.**

    + If it is **not** a **hate speech**, it will return **FALSE.**

Decomposition (How would you break down your problem into sub- problems?)

(1) Sub-Problem: Data collecttion
(2) Sub-Problem: Split train data and test data
(3) Sub-Problem: Preprocess data
(4) Sub-Problem: Process data
(5) Sub-Problem: Trainning model and Predict

Pattern Recognition (Are there related solutions to draw on?)

(1) Collect comments on Facebook by crawling data.

(2) Split data: 70% train data, 30% test data, and validation data is collected separately from the two data above.

Abstraction (How would you abstract this problem?)

3. Interation 3

Problem Identification

(3) Sub-Problem: Preprocess Data

Decomposition (How would you break down your problem into sub-problems?)

(1) Sub-Problem: Clean Data
(2) Sub-Problem: Words segmentation
(3) Sub-Problem: Convert comments to lowercase
(4) Sub-Problem: Remove icons and special characters

Pattern Recognition (Are there related solutions to draw on?)

(1): Remove Stop Words
(2): Dividing written text into meaningful units, such as words, sentences, or topics.

Abstraction (How would you abstract this problem?)

- Remove comments with only icon or special characters

4. Interation 4

Problem Identification

(4) Sub-Problem: Process Data

Decomposition (How would you break down your problem into sub-problems?)

(1) Sub-Problem: create vector for word

Pattern Recognition (Are there related solutions to draw on?)

(1): Using **TF-IDF** (Term Frequency - Inverse Document Frequency) technique

Abstraction (How would you abstract this problem?)

5. Interation 5

Problem Identification

(5) Sub-Problem: Training model and Predict

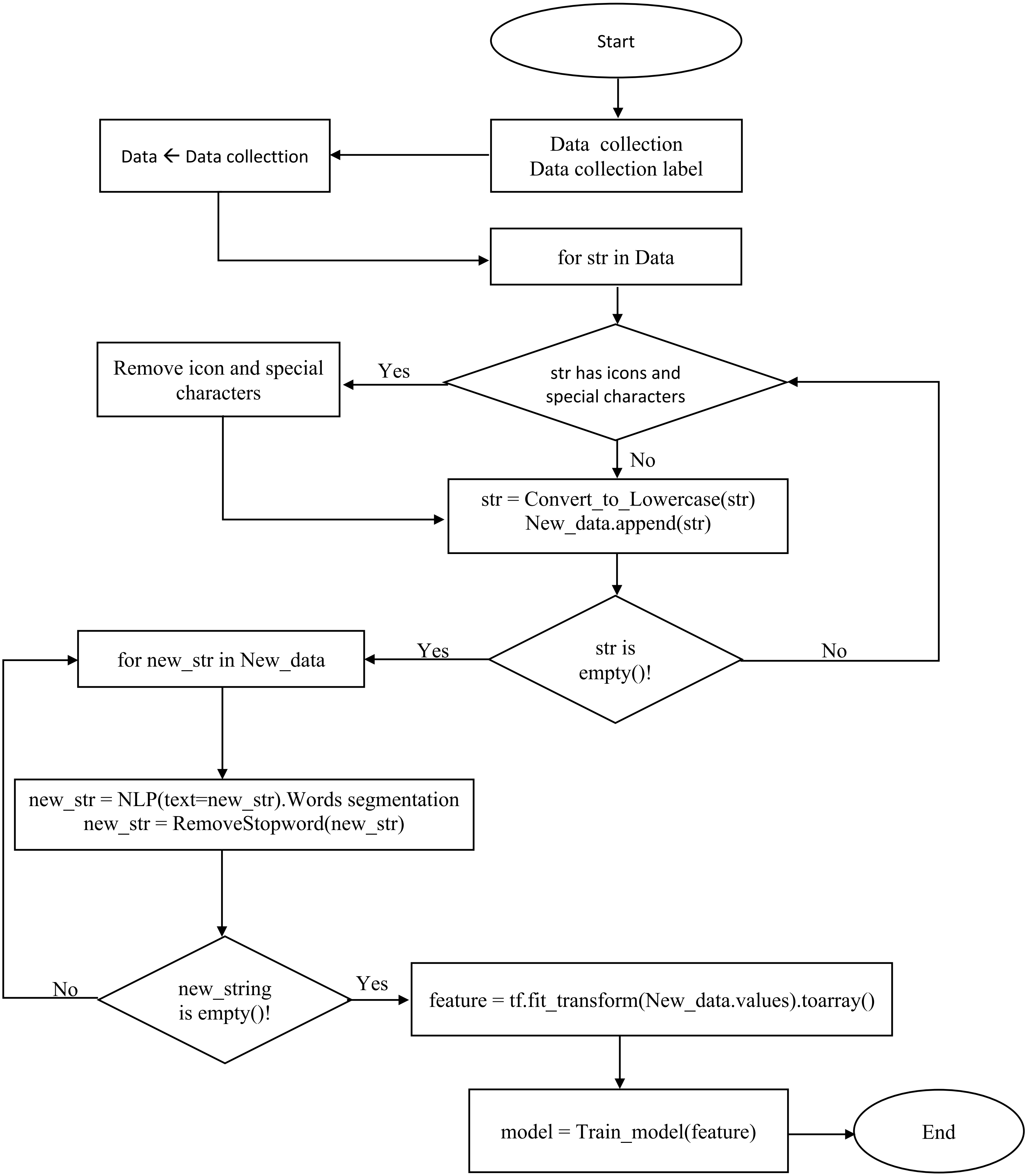
Decomposition (How would you break down your problem into sub-problems?)

(1) Support Vector Classification
(2) Model Naive Bayes
(3) Model Logistic Regression
(4) Model Decision Tree Classifier

Pattern Recognition (Are there related solutions to draw on?)

Abstraction (How would you abstract this problem?)

III. Algorithm Flowchart



IV. Techniques applied in the problem

- In this section, the group will **present** the above **Decomposition** techniques to solve this problem: **TF-IDF** (Term Frequency - Inverse Document Frequency), **Remove Stopword**, **Text segmentation**.

1. TF-IDF (Term Frequency – Inverse Document Frequency)

- **TF-IDF** (Term Frequency – Inverse Document Frequency) is a numerical statistic that is intended to reflect how important a word is to a **document** in a collection or **corpus**. It is often used as a **weighting factor** in searches of information retrieval, **text mining**, and **user modeling**.

- **TF**: Term Frequency (Frequency of a word) is the number of times the word appears in the text.

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

Image 2: Formula to calculate Term Frequency

In there:

- tf(t, d): the frequency of occurrence of the **word t** in the **text d**.
- f(t, d): Number of occurrences of the **word t** in the **text d.**
- max({f(w, d) : w ∈ d}): The number of occurrences of the word with **the most occurrences** in the **text d**.

- **IDF:** Inverse Document Frequency (Inverse of text frequency), helps to assess the importance of a word.

$$\mathrm{idf}(t,D) = \log\frac{|D|}{|\{d \in D : t \in d\}|}$$

Image 3: Formula to calculate Inverse Document Frequency

In there:

- idf(t, D): idf value of word t in corpus.
- |D|: Total number of texts in the **set D**.
- |{d ∈ D : t ∈ d}|: represents the number of documents in **set D** containing the **word t.**

2. Text segmentation

- **Text segmentation** is the process of dividing written text into meaningful units, such as **words**, **sentences**, or **topics**. The term applies both to **mental processes** used by humans when reading text, and to artificial processes implemented in computers, which are the subject of **natural language processing.**

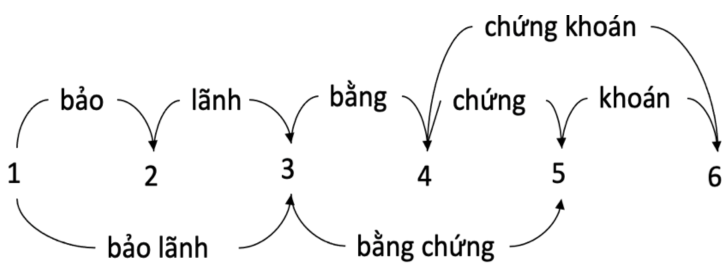


Image 4: Demo Text segmentation

- ex = " Nguyễn Quang Hải là cầu thủ bóng đá chuyên nghiệp của đội tuyển Việt Nam"

print NLP(text=ex).segmentation()

→ **"**Nguyễn_Quang_Hải là cầu_thủ bóng_đá chuyên_nghiệp của đội_tuyển Việt_Nam"

3. Remove Stopword

- **Remove Stopword** is to pick out the **important words** or **phrases** in a sentence and **remove unnecessary words** in the sentence. Those removed words are called **Remove StopWord** that have **no meaning** in the classification.

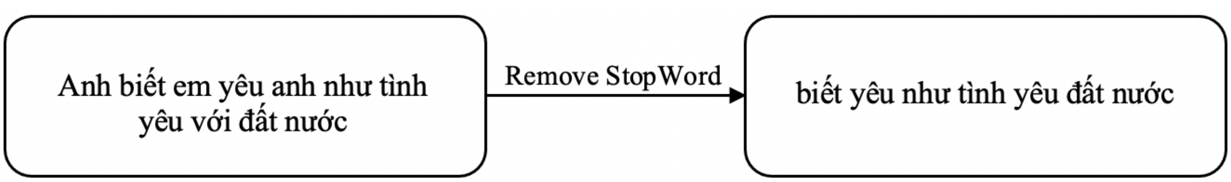


Image 5: Demo Remove StopWord

V. Description of the dataset.

- How to collect datasets:

  + Select a few articles on the social network facebook that caused a stir in the online community.

  + Get all comments in those articles by crawling (the way to crawl data and source code will be on the group's shared github).

- The group will take the data **available** on **github** to train the model. The group will then **crawl comments** on Facebook to do their own **Validation Data**.

- **4764 comments** – From the beginning, nearly **10000 comments** group has filtered out duplicate comments, comments only contain icons, comments have no meaning, …

  + 2121 comments are **hate speech**.

  + The remaining 2643 comments are **not hate speech**.

- Split data into: 70% Train data, 30% Test data from 4746 comments. In addition, the team collects more data to make validation data used to evaluate the model after training.

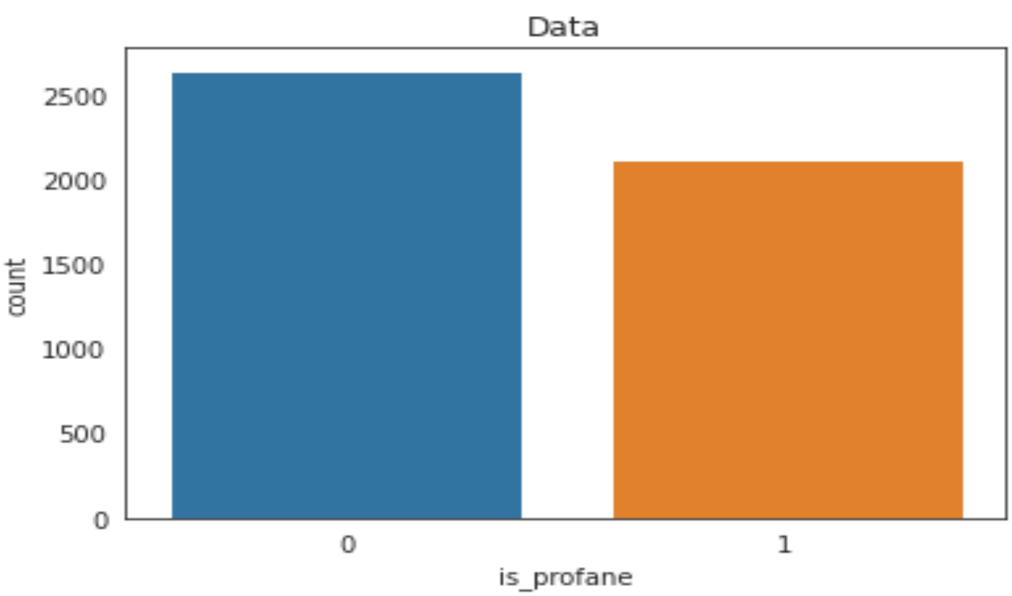


Image 6: The graph shows the amount of data

- Link repositories on the group's github:
https://github.com/trong-khanh-1109/CS117.L22.KHCL

VI. Result evaluation

- With the problem of classifying **hate speech** on the social network facebook, the group will use the following models to train:

- Support Vector Classification
- Model Naive Bayes
- Model Logistic Regression
- Model Decision Tree Classifier

1. Support Vector Classification

```
[27] print("Evaluating by model SVC...")
     model_SVC = LinearSVC()
     model_SVC.fit(X_train, Y_train)
     Y_pred_val = model_SVC.predict(X_val)
     Y_pred_test = model_SVC.predict(X_test)
```

- Results on validation set:

	precision	recall	f1-score	support
0	0.84	0.85	0.84	731
1	0.70	0.67	0.68	371
accuracy			0.79	1102
macro avg	0.77	0.76	0.76	1102
weighted avg	0.79	0.79	0.79	1102

→ **Accuracy: 79%**

- Results on test set:

	precision	recall	f1-score	support
0	0.83	0.91	0.86	771
1	0.88	0.78	0.82	659
accuracy			0.85	1430
macro avg	0.85	0.84	0.84	1430
weighted avg	0.85	0.85	0.85	1430

→ **Accuracy: 86%**

2. Model Naive Bayes

```
[32] print("Evaluating by model Naive Bayes...")
     model_NB = MultinomialNB()

     model_NB.fit(X_train, Y_train)
     Y_pred_val = model_NB.predict(X_val)
     Y_pred_test = model_NB.predict(X_test)
```

- Results on validation set:

	precision	recall	f1-score	support
0	0.79	0.91	0.85	731
1	0.75	0.51	0.61	371
accuracy			0.78	1102
macro avg	0.77	0.71	0.73	1102
weighted avg	0.77	0.78	0.77	1102

→ **Accuracy: 78%**

- Results on test set:

	precision	recall	f1-score	support
0	0.76	0.94	0.84	771
1	0.90	0.66	0.76	659
accuracy			0.81	1430
macro avg	0.83	0.80	0.80	1430
weighted avg	0.83	0.81	0.80	1430

→ **Accuracy: 81%**

3. Model Logistic Regression

```
[34] print("Evaluating by model Logistic Regression...")
     model_LG = LogisticRegression()

     model_LG.fit(X_train, Y_train)
     Y_pred_val = model_LG.predict(X_val)
     Y_pred_test = model_LG.predict(X_test)
```

- Results on validation set:

	precision	recall	f1-score	support
0	0.82	0.86	0.84	731
1	0.69	0.64	0.67	371
accuracy			0.78	1102
macro avg	0.76	0.75	0.75	1102
weighted avg	0.78	0.78	0.78	1102

→ **Accuracy: 78%**

- Results on test set:

	precision	recall	f1-score	support
0	0.78	0.94	0.85	771
1	0.91	0.69	0.78	659
accuracy			0.82	1430
macro avg	0.84	0.81	0.82	1430
weighted avg	0.84	0.82	0.82	1430

→ **Accuracy: 82%**

4. Model Decision Tree Classifier

```
[36] print("Evaluating by model Decision Tree Classifier...")
     model_DT = DecisionTreeClassifier()

     model_DT.fit(X_train, Y_train)
     Y_pred_val = model_DT.predict(X_val)
     Y_pred_test = model_DT.predict(X_test)
```

- Results on validation set:

	precision	recall	f1-score	support
0	0.82	0.86	0.84	731
1	0.69	0.64	0.67	371
accuracy			0.78	1102
macro avg	0.76	0.75	0.75	1102
weighted avg	0.78	0.78	0.78	1102

→ **Accuracy: 78%**

- Results on test set:

	precision	recall	f1-score	support
0	0.83	0.87	0.85	771
1	0.84	0.80	0.82	659
accuracy			0.84	1430
macro avg	0.84	0.83	0.84	1430
weighted avg	0.84	0.84	0.84	1430

→ **Accuracy: 84%**

VII. Reviews

- Model Decision Tree Classifier and Logistic Regression gives accuracy on the highest validation set (80%).

- Model SVC gives accuracy on the highest test set (85%).

VIII. References

[1]. https://vi.wikipedia.org/wiki/Tf%E2%80%93idf
[2]. https://en.wikipedia.org/wiki/Text_segmentation
[3]. https://github.com/ducvuuit/CS114.K21.KHTN
[4]. https://github.com/langmaninternet/VietnameseTextNormalizer?fbclid=IwAR3EN_3JNG16ZhBRYw2x4HHUqNTybyFBZ9xpkm4ABVCDUBzRj0elLm5Yyqo
[5]. https://github.com/stopwords/vietnamese-stopwords
[6]. https://kipalog.com/posts/Machine-Learning---NLP--Text-Classification-su-dung-scikit-learn----python