

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



**BÁO CÁO ĐỒ ÁN**  
**MÔN HỌC: LẬP TRÌNH PYTHON**  
**CHO MÁY HỌC**

**Đề tài:**

**Cây phân loại quyết định**  
**(Decision Tree Classifier)**

**Lớp: CS116.M12.KHCL**

**Giảng viên hướng dẫn: TS. Nguyễn Vinh Tiệp**

**Sinh viên thực hiện:**

Phan Nguyễn Thành Nhân	19521943
Phạm Minh Long	19521797
Tạ Huỳnh Đức Huy	19521634

Thành phố Hồ Chí Minh, ngày 07 tháng 12 năm 2021

# MỤC LỤC

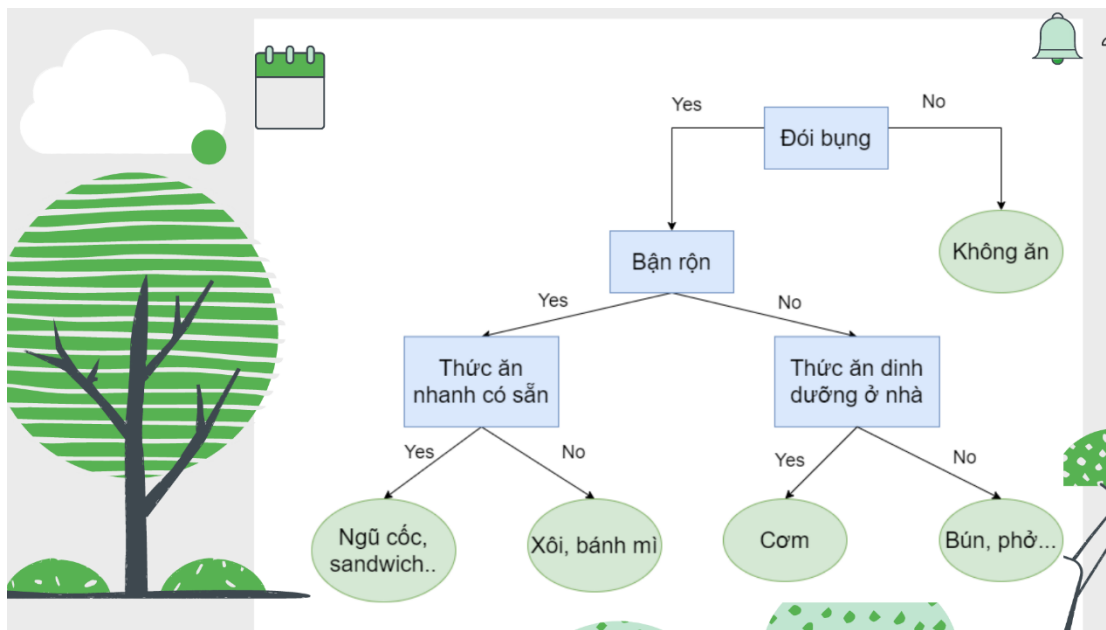
<b>I. Giới thiệu .....</b>	<b>1</b>
<b>1. Giới thiệu tổng quan.....</b>	<b>1</b>
<b>2. Cây quyết định.....</b>	<b>2</b>
<b>II. Cây phân loại quyết định .....</b>	<b>3</b>
<b>1. Thuật toán.....</b>	<b>3</b>
<b>1.1. Mô hình thuật toán .....</b>	<b>3</b>
<b>1.2. Các siêu tham số .....</b>	<b>4</b>
<b>2. Áp dụng vào bài toán thực nghiệm .....</b>	<b>6</b>
<b>2.1. Mô tả bài toán .....</b>	<b>6</b>
<b>2.2. Mô tả bộ dữ liệu .....</b>	<b>7</b>
<b>2.3. Thực nghiệm áp dụng và đánh giá kết quả .....</b>	<b>9</b>
<b>2.4 Thử nghiệm với các kỹ thuật khác .....</b>	<b>14</b>
<b>III. Kết luận .....</b>	<b>16</b>
<b>1. Nhận xét chung .....</b>	<b>16</b>
<b>2. Ưu nhược điểm của thuật toán.....</b>	<b>16</b>
<b>3. Kết luận .....</b>	<b>17</b>
<b>IV. Tài liệu tham khảo.....</b>	<b>18</b>

# I. Giới thiệu

## 1. Giới thiệu tổng quan

Ngày nay, có rất nhiều các phương pháp phân tích cấu trúc logic của bộ dữ liệu số thông qua sự thay đổi các trạng thái khác nhau của một số thuộc tính. Mục tiêu chính của việc phân tích là đưa ra các quy tắc toán học xác định nhằm phân loại các dữ liệu khác nhau và giúp đưa ra quyết định. Gần gũi với mỗi người nhất là *Cây quyết định* (Decision Tree), với việc được áp dụng hằng ngày trong cuộc sống của chúng ta.

Một ví dụ thực tế, *Cây quyết định* có thể xây dựng dựa trên việc mỗi khi bạn cảm thấy đói bụng với các thuộc tính cơ bản như khoảng thời gian bạn có hay loại thức ăn mà bạn muốn giúp đưa ra một phương án tối ưu để đưa ra quyết định.



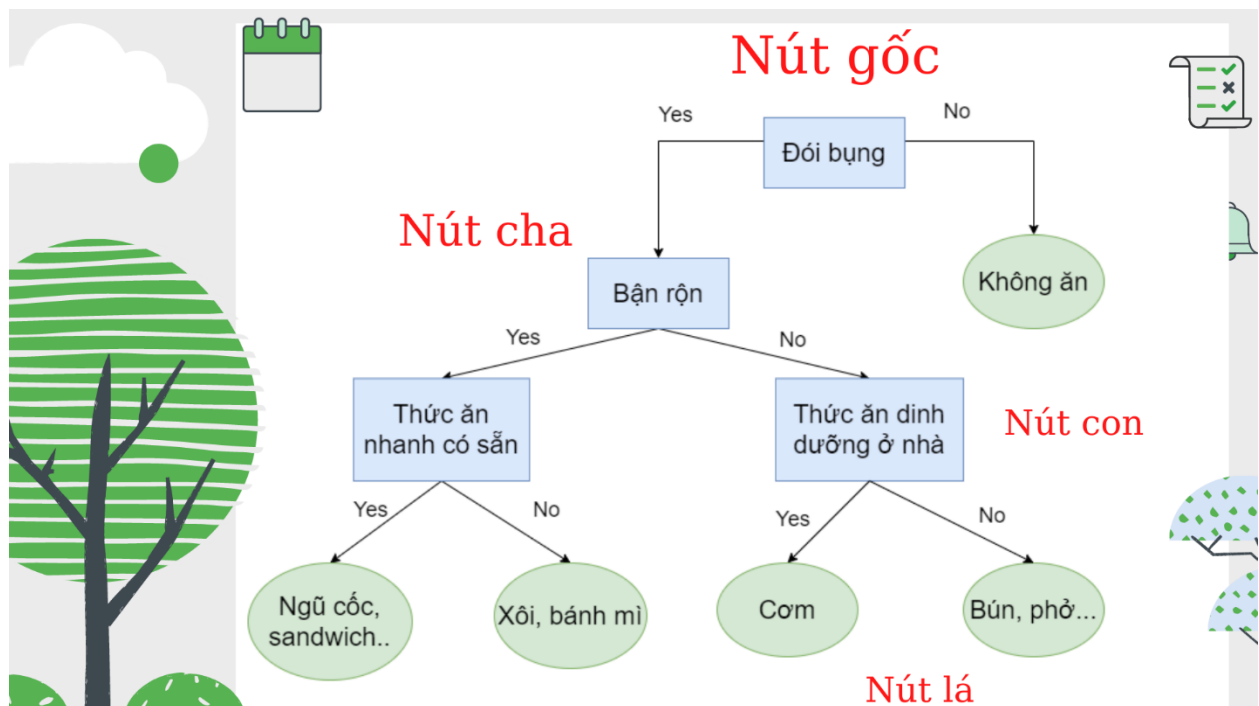
Hình 1: Ảnh minh họa phân tích các quyết định khi đói bụng

Đây là một dạng của *Cây quyết định*. Mô hình *Cây quyết định* có thể xây dựng dựa trên tập hợp các quyết định theo thứ bậc một cách trình tự dùng để phân lớp giá trị các đối tượng. Trong cuộc sống như ví dụ trên, *Cây quyết định* đại diện các quy tắc giúp chúng ta đưa ra quyết định chính

xác. *Cây quyết định* cũng có thể được mở rộng thêm về chiều sâu nếu có thêm lựa chọn giữa quán ăn cũ hay quán mới, ...

## 2. Cây quyết định

Về ý tưởng, *Cây quyết định* được xây dựng đơn giản giống với cách mà con người sử dụng. Khi chia dữ liệu thành các phần khác nhau, đầu tiên phải dựa vào các câu hỏi. Từng bước, dữ liệu được phân tách thành các phần và cuối cùng còn lại những mẫu cần thiết. Đây là bản chất của toàn bộ thuật toán.



Hình 2: Kiến trúc của một Cây quyết định khi áp dụng vào bài toán “Đói bụng”

Để giải thích rõ hơn về khái niệm *Cây quyết định*, chúng tôi sẽ dùng một số thuật ngữ phổ biến:

- *Nút* (Node): là thành phần chính chứa các câu hỏi phân tách và kết quả của *Nút cha*.
- *Nút gốc* (Root node): là nút bắt đầu mỗi cây và không chứa dữ liệu của *Nút cha* (là duy nhất).

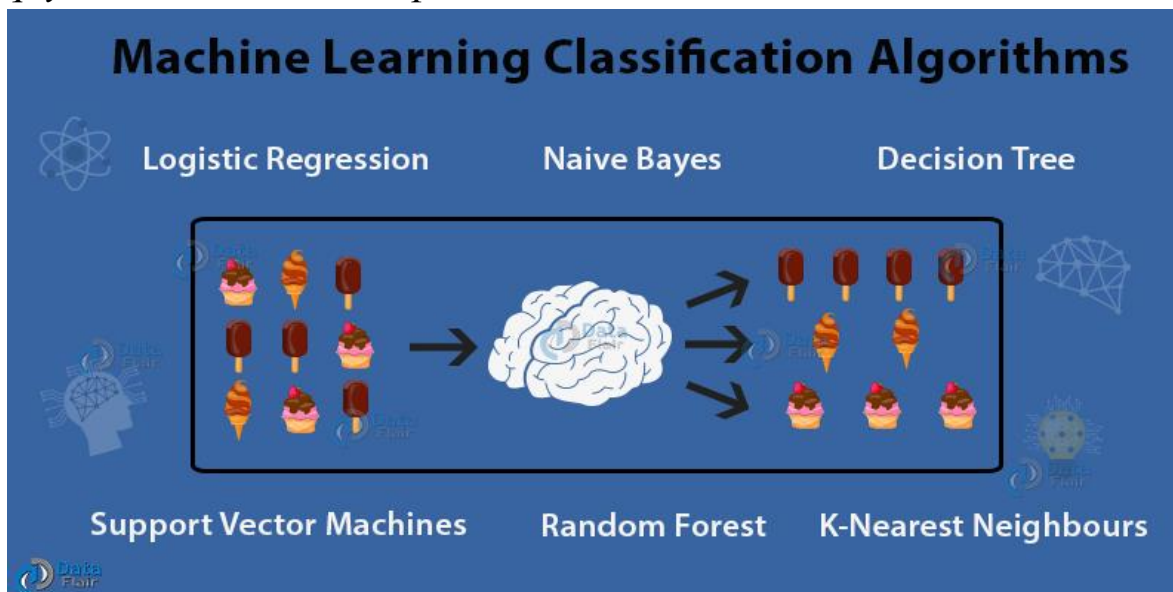
- *Nút cha* (Parent node): là nút chứa câu hỏi để phân chia dữ liệu nút hiện tại.
- *Nút con* (Child node): là nút chứa dữ liệu sau khi được lọc từ *Nút cha*.
- *Nút lá* (Leaf node): là những nút cuối của mỗi cây, không có bất kỳ câu hỏi nào và chứa tập hợp con của bộ dữ liệu đại diện cho câu trả lời của các câu hỏi trước.

## II. Cây phân loại quyết định

### 1. Thuật toán

#### 1.1. Mô hình thuật toán

Trong *Máy học* (Machine Learning), bên cạnh một vài thuật toán phổ biến như k-Nearest Neighbors, Naïve Bayes, Support Vector Machines, ..., *Cây quyết định* vẫn mang lại giá trị của riêng nó. *Cây quyết định* là một mô hình *Tự học có giám sát* (Supervised learning) dùng trong các bài toán *phân loại* (Classification) hoặc *hồi quy* (Regression). Trong phân mục này, chúng tôi sẽ đi qua khái niệm chính của mô hình *Cây quyết định* cho bài toán *phân loại*.



Hình 3: Các thuật toán phân loại phổ biến trong Máy học

Thông thường, để phân loại, mô hình sẽ được huấn luyện dựa trên bộ dữ liệu đi kèm với bộ *nhãn* (labels) tương ứng cho trước. Sau đó, mô hình dự đoán một *nhãn* trong số các *nhãn* đã huấn luyện cho một *mẫu* (sample) mới. Các *mẫu* mới bao gồm các *thuộc tính/đặc trưng* (attributes/features) giống như bộ dữ liệu huấn luyện. Mỗi *mẫu* là đều có sự kết hợp riêng về giá trị của các *thuộc tính*.

Về cơ bản, thuật toán được xây dựng sau khi duyệt qua tất cả các *đặc trưng*, các *đặc trưng* có tính chọn lọc thấp (chứa nhiều *tạp chất* - impurity) thường sẽ được chọn làm *nút cha* hơn làm *nút con*. Điều này có nghĩa các *đặc trưng* có nhiều *tạp chất* thường sẽ ở gần *nút gốc*. Ngược lại, các *đặc trưng* có ít *tạp chất* sẽ gần *nút lá* và thường mang ý nghĩa quyết định loại *nhãn* của *mẫu*.

Cuối cùng, một *Cây quyết định* được xây dựng chứa số lượng lớn câu hỏi và các câu trả lời tương ứng. Dữ liệu đầu vào là một vector các đặc trưng, đầu ra là *nhãn* dự đoán. Để dự đoán *nhãn*, các *đặc trưng* của *mẫu* sẽ lần lượt được chọn lọc thông qua các câu hỏi theo thứ tự từ *nút gốc* qua các *nút* tới *nút lá*. Mỗi *nút lá* chứa xác suất cho mỗi *nhãn* và mô hình sẽ chọn *nhãn* có xác suất cao nhất để trả về kết quả cuối cùng.

## 1.2. Các siêu tham số

Trong phần này, chúng tôi sẽ tìm hiểu về các siêu tham số của thuật toán *Cây quyết định* đã được xây dựng bởi thư viện Scikit-learn. Thư viện Scikit-learn đã cung cấp cho chúng ta rất nhiều siêu tham số của thuật toán nhưng chúng tôi đã lựa chọn ra tám siêu tham số có ảnh hưởng nhất để tìm hiểu và tinh chỉnh trong quá trình thực nghiệm các bài toán thực tế.



Hình 4: Các siêu tham số mà nhóm đã lựa chọn

### 1.2.1. Tham số *criterion*

Trong *Cây quyết định*, tham số *criterion* đóng vai trò là chiến lược lựa chọn các *đặc trưng* dùng trong cho các câu hỏi trong các *nút*. Tham số *criterion* có hai giá trị là ‘gini’ và ‘entropy’, mặc định trong thư viện *scikit-learn* là “gini”. Giữa hai chiến lược “gini” và “entropy” không có sự khác biệt lớn về độ chính xác, tuy nhiên chiến lược “gini” có thời gian tính toán nhanh hơn vì tốn ít chi phí hơn so với chiến lược “entropy”.

### 1.2.2. Tham số *max\_deep*

Trong hầu hết các thuật toán về cây thì chiều cao hay độ sâu là tham số không thể thiếu. Đối với *Cây quyết định*, tham số *max\_deep* được thư viện *scikit-learn* định nghĩa là số nguyên đại diện độ sâu tối đa của cây mà tại đó sẽ là *nút lá*. Trong *scikit-learn* mặc định tham số này sẽ là **None**. Trong quá trình áp dụng bài toán, tùy vào mục đích sử dụng và tùy vào bộ dữ liệu của bài toán thì chúng ta sẽ tinh chỉnh tham số này khác nhau, nếu chọn giá trị thấp sẽ giúp mô hình dự đoán nhanh hơn nhưng ít chính xác, còn nếu chọn giá trị cao sẽ dễ dẫn đến *sự quá khớp* (overfitting) và bị chậm.

### 1.2.3. Tham số *min\_samples\_split*

Trong mô hình *Cây quyết định*, *min\_samples\_split* là số lượng *mẫu* (nút con) tối thiểu cần thiết để một *nút cha* tách ra, mang giá trị số nguyên hoặc số thực. Nếu là số thực sẽ trả số nguyên nhỏ nhất lớn hơn hoặc bằng tích giữa *min\_samples\_split* và số lượng *mẫu* của bộ dữ liệu. Mặc định, *min\_samples\_split* mang giá trị bằng 2, tùy vào từng bài toán chúng ta có thể điều chỉnh tham số để tránh *sự quá khớp* (overfitting).

### 1.2.4. Tham số *min\_samples\_leaf*

Tương tự như *min\_samples\_split*, tham số *min\_samples\_leaf* là số *nút lá* tối thiểu cần thiết để một *nút cha* tách ra khi cây đã đạt tới độ sâu cuối cùng. Mặc định tham số này mang giá trị bằng 1.

### 1.2.5. Tham số *max\_features*

Là số lượng *đặc trưng* yêu cầu dùng để phân chia tại các *nút*, mang giá trị là số nguyên hoặc số thực, ngoài ra miền giá trị của tham số có thể là *auto*, *sqrt*, *log2*. Mặc định là None. Giá trị *max\_feature* càng cao giúp tăng độ chính xác nhưng bù lại tốn nhiều thời gian hơn.

#### **1.2.6. Tham số *max\_leaf\_nodes***

Là số lượng *nút lá* tối đa, mang giá trị số nguyên. Mặc định là None. Tham số *max\_leaf\_nodes* dùng để giới hạn số lượng *nút lá*, chỉ bao gồm các *nút lá* tốt được chọn.

#### **1.2.7. Tham số *min\_impurity\_decrease***

Cây sẽ không sinh ra *nút con* nếu *tập chất* nhiều hơn ngưỡng. Với ngưỡng tạo *nút* là số thực. Tham số *min\_impurity\_decrease* được dùng để cân bằng giữa *sự quá khớp* và độ chính xác. Tham số có giá trị là số thực, mặc định là 0.0. Giá trị thấp mang lại độ chính xác cao, rủi ro *sự quá khớp* cũng tăng lên và ngược lại.

#### **1.2.8. Tham số *ccp\_alpha***

*Cây con* (subtree) có độ phức tạp thấp hơn sẽ bị lược bỏ. Với hệ số *cắt tỉa* (pruning) là số thực dương. Được dùng để cân bằng giữa *sự quá khớp* và độ chính xác. Giá trị thấp mang lại độ chính xác cao, rủi ro *sự quá khớp* cũng tăng lên và ngược lại.

## **2. Áp dụng vào bài toán thực nghiệm**

### **2.1. Mô tả bài toán**

Trong thời buổi công nghệ ngày càng phát triển như hiện nay, việc tiếp cận thông tin trở nên cực kỳ nhanh chóng nhờ sự phát triển của Internet và cách thiết bị thông minh. Thế nhưng, đi kèm với những thông tin hữu ích là những bài viết, chủ đề sai lệch sự thật, châm biếm nhằm các mục đích khác nhau. Với mục tiêu giúp mọi người tránh tiếp cận những thông tin không bổ ích, chúng tôi xây dựng mô hình Decision Tree Classifier trên [bộ dữ liệu phân loại tin châm biếm](#) từ Kaggle.





Hình 5: Một bài báo châm biếm trên trang theonion.com

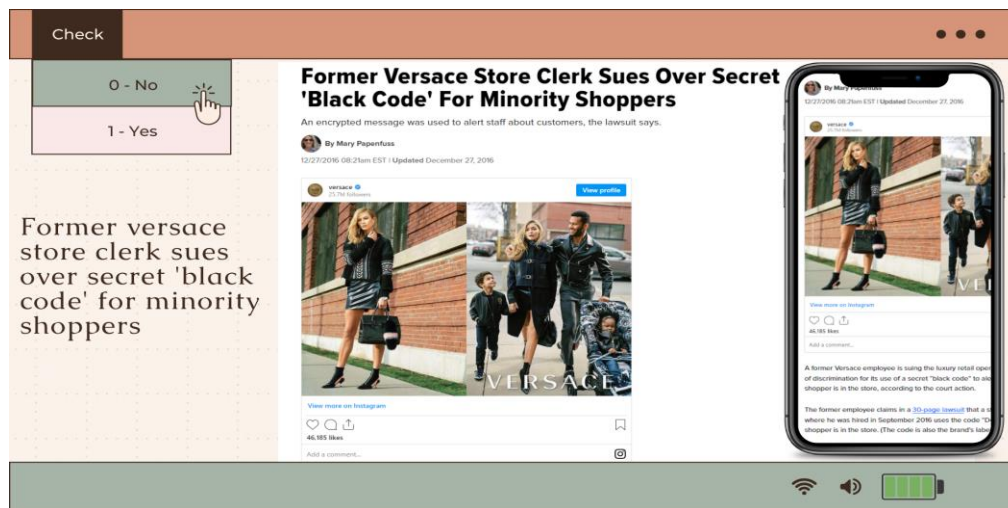
## 2.2. Mô tả bộ dữ liệu

Đây là bộ dữ liệu các tiêu đề bài báo điện tử bằng tiếng Anh được thu thập từ 2 nguồn chính. Cái tiêu đề thu thập từ [TheOnion](http://theonion.com) mang tính chất châm biếm. Đối với các tin tức đúng mang lại giá trị cao được lấy từ [HuffPost](http://huffpost.com).

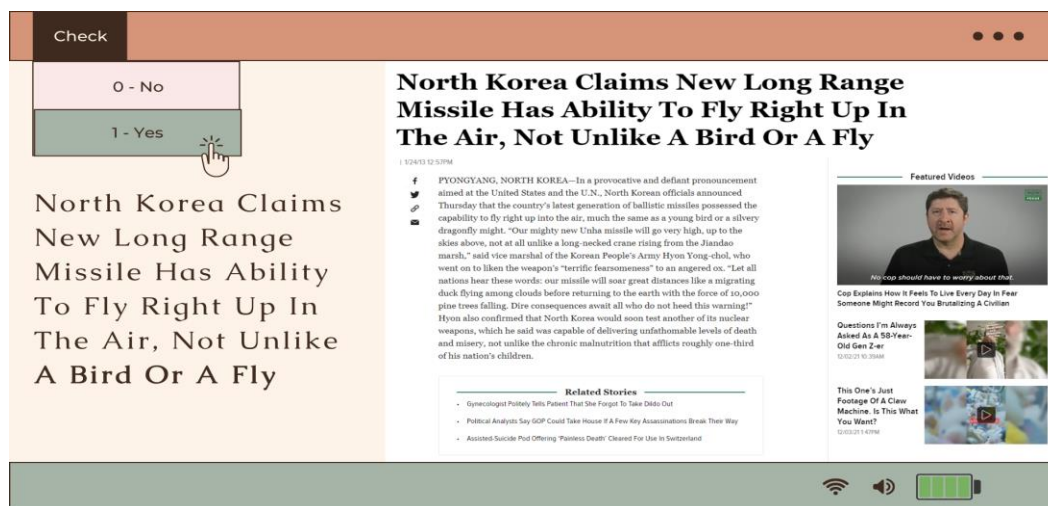
Bộ dữ liệu gồm hai loại tập tin, với mỗi tập tin gồm ba thành phần chính sau:

- + *is\_sarcastic* (có trào phúng): 1 tương ứng với tiêu đề là trào phúng, 0 đại diện cho tiêu đề không trào phúng
- + *headline*: tiêu đề của bài báo
- + *article\_link*: đường dẫn đến bài báo tương ứng

## Ví dụ minh họa về bộ dữ liệu:



Hình 6: Bài viết được đánh giá là không “châm biếm”



Hình 7: Bài viết được đánh giá là “châm biếm”

Đánh giá tổng quan về bộ dữ liệu trên kaggle, chúng tôi rút ra những ưu điểm như sau. *Thứ nhất*, các tiêu đề được viết bởi các chuyên gia nên hình thức và cách viết luôn ở dạng chuẩn, không có lỗi chính tả giúp cho việc phân loại dễ dàng hơn. *Thứ hai*, vì mục đích của trang TheOnion là tạo ra các tin tức châm biếm nên dữ liệu ít nhiều hơn. *Cuối cùng*, chủ đề các bài báo khép kín trong vài phạm vi nên nâng cao được hiệu quả dự đoán cho mô hình.

Bên cạnh đó, ngoài các ưu điểm trên, các bài báo thu thập vẫn còn mang một vài nhược điểm như sau. Về tính đa dạng, các bài báo chỉ bao gồm một vài chủ đề tin tức cố định, cho nên độ đa dạng của các bài báo là rất thấp. Về ngôn ngữ, các bài báo sử dụng ngôn ngữ và văn phong theo tiếng Anh-Mỹ nên bị hạn chế nếu dùng cho các chuẩn tiếng Anh khác.

### 2.3. Thực nghiệm áp dụng và đánh giá kết quả

Để xây dựng mô hình *Cây phân loại quyết định*, chúng ta cần đọc hai tập tin dữ liệu json bằng thư viện Pandas. Sau khi có được dữ liệu, chúng ta loại bỏ cột thuộc tính *article\_link* không cần thiết và thực hiện nối hai bộ dữ liệu thành một bằng hàm *concat()* của Pandas.

```
[ ] data1 = pd.read_json('/content/Sarcasm_Headlines_Dataset.json', lines=True)
data1.head(3)
```

	article_link	headline	is_sarcastic
0	https://www.huffingtonpost.com/entry/versace-b...	former versace store clerk sues over secret 'b...	0
1	https://www.huffingtonpost.com/entry/roseanne-...	the 'roseanne' revival catches up to our thorn...	0
2	https://local.theonion.com/mom-starting-to-fea...	mom starting to fear son's web series closest ...	1

```
[ ] data2 = pd.read_json('/content/Sarcasm_Headlines_Dataset_v2.json', lines=True)
data2.head(3)
```

	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-sci...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...

```
[ ] df = pd.concat([data1[['headline', 'is_sarcastic']], data2[['headline', 'is_sarcastic']]], ignore_index=True)
df.head(3)
```

	headline	is_sarcastic
0	former versace store clerk sues over secret 'b...	0
1	the 'roseanne' revival catches up to our thorn...	0
2	mom starting to fear son's web series closest ...	1

Hình 8: Đọc và tiền xử lý dữ liệu

Tiếp đến là kiểm tra phân phối của biến phụ thuộc, tức phân phối các giá trị cột *is\_sarcastic* nhằm xác định tính cân đối của dữ liệu.



Hình 9: Vẽ biểu đồ kiểm tra độ phân phối của dữ liệu

Do mức độ chênh lệch không quá nhiều giữa hai giá trị, chúng ta chuyển sang chuẩn hóa thuộc tính của dữ liệu (tức cột *headline*). Vì dữ liệu đầu vào của mô hình yêu cầu một vector số nên cần biến đổi các tiêu đề về đúng định dạng. Chúng tôi sử dụng hàm *TfidfVectorizer()* của thư viện *scikit-learn* chuyển đổi dữ liệu dạng chữ sang vector ma trận đặc trưng *TF-IDF*. Mục tiêu của hàm là đếm số lần xuất hiện của các từ.

*TF-IDF* (Term Frequency - Inverse Document Frequency) là một kỹ thuật dùng trong khai thác dữ liệu văn bản. Trọng số này được dùng để đánh giá tầm quan trọng của một từ trong văn bản. Giá trị cao thể hiện độ quan trọng cao và nó phụ thuộc vào số lần từ xuất hiện nhưng bù lại bởi tần suất của từ đó trong tập dữ liệu. Giá trị trọng số được tính bằng tích

giữa TF (Term Frequency - Tần suất xuất hiện của từ) và IDF (Inverse Document Frequency - Nghịch đảo tần suất của văn bản) theo công thức sau:

$$\begin{aligned} \text{tf}(t, d) &= \frac{f(t, d)}{\max\{f(w, d) : w \in d\}} \\ \text{idf}(t, D) &= \log \frac{|D|}{|\{d \in D : t \in d\}|} \end{aligned}$$

Hình 10: Công thức TF-IDF để đánh giá tầm quan trọng của một từ trong văn bản

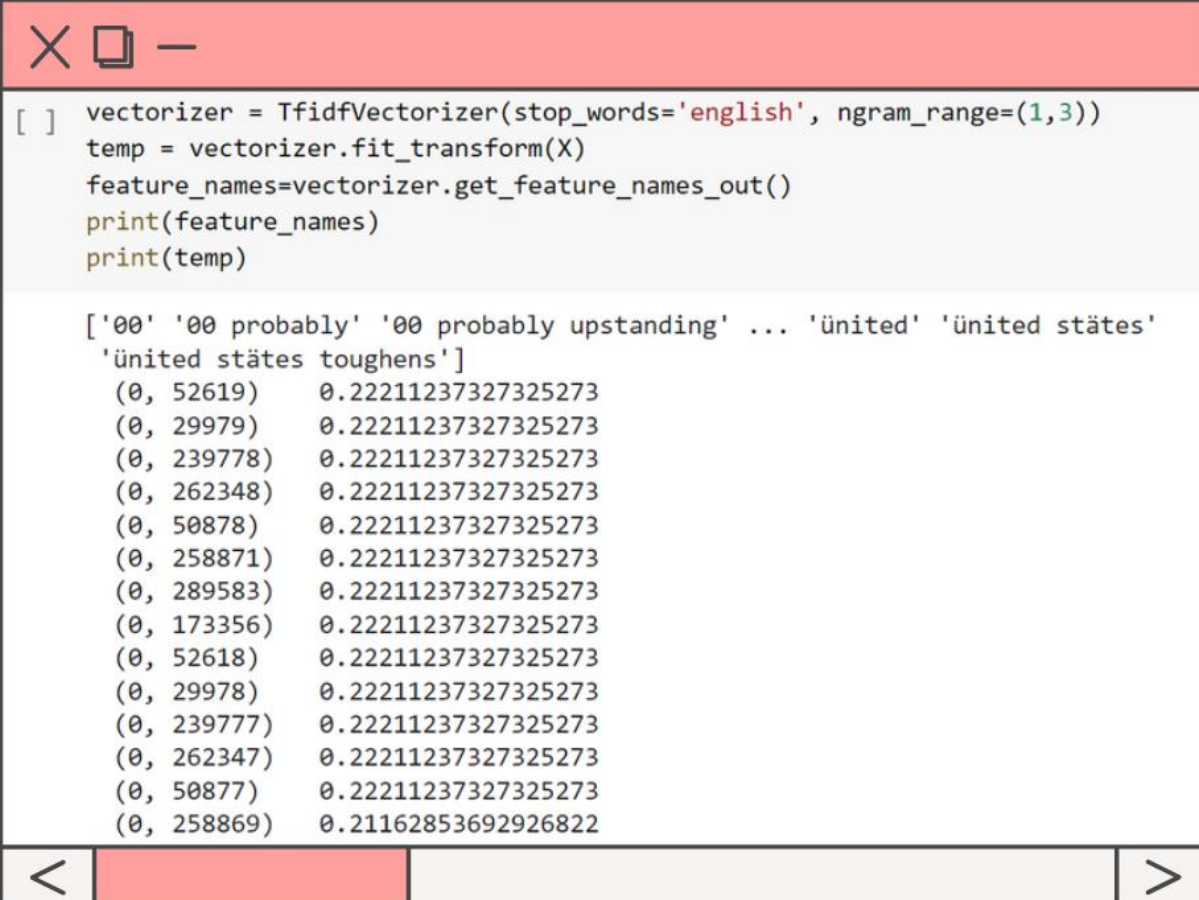
Trong đó:

- $\text{tf}(t, d)$ : Tần suất xuất hiện của từ  $t$  trong văn bản  $d$
- $f(t, d)$ : Số lần xuất hiện của từ  $t$  trong văn bản  $d$
- $\max\{f(w, d) : w \in d\}$ : Số lần xuất hiện của từ có số lần xuất hiện nhiều nhất trong văn bản  $d$
- $\text{idf}(t, D)$ : giá trị  $\text{idf}$  của từ  $t$  trong tập dữ liệu  $D$
- $|D|$ : tổng số văn bản trong tập dữ liệu  $D$
- $|\{d \in D : t \in d\}|$ : số văn bản trong tập  $D$  có chứa từ  $t$

Ngoài ra nhằm tăng tính chính xác và lọc bớt nhiễu, chúng tôi đã loại bỏ các *từ dừng* (stop-words) nhưng không mang lại ý nghĩa cho câu. Trong tiếng Việt, các từ này bao gồm như “bị”, “bởi”, ... Còn đối với tiếng Anh có các từ như “so”, “that”, ... Vì ngôn ngữ sử dụng trong tiêu đề của bộ dữ liệu là tiếng Anh, nên ở đây chúng tôi bỏ *từ dừng* tiếng Anh. Hàm *TfidfVectorizer()* sẽ tự chuyển đổi các ký tự in hoa thành in thường, đồng thời cũng loại bỏ các ký tự đặc biệt. *ngram\_range=(1,3)* được dùng



để đếm mối liên hệ của ba từ liên tiếp nhau. Giá trị trả về là một vector giá trị *TF-IDF* của các từ và cụm từ.



```
[ ] vectorizer = TfidfVectorizer(stop_words='english', ngram_range=(1,3))
    temp = vectorizer.fit_transform(X)
    feature_names=vectorizer.get_feature_names_out()
    print(feature_names)
    print(temp)
```

```
['00' '00 probably' '00 probably upstanding' ... 'üited' 'üited stätes'
 'üited stätes toughens']
(0, 52619)    0.22211237327325273
(0, 29979)    0.22211237327325273
(0, 239778)   0.22211237327325273
(0, 262348)   0.22211237327325273
(0, 50878)    0.22211237327325273
(0, 258871)   0.22211237327325273
(0, 289583)   0.22211237327325273
(0, 173356)   0.22211237327325273
(0, 52618)    0.22211237327325273
(0, 29978)    0.22211237327325273
(0, 239777)   0.22211237327325273
(0, 262347)   0.22211237327325273
(0, 50877)    0.22211237327325273
(0, 258869)   0.21162853692926822
```

Hình 11: Tiêu đề của bài báo sau khi loại bỏ stop word và chuyển sang ma trận TD-IDF

Cuối cùng, chúng ta thực hiện khởi tạo và huấn luyện mô hình bằng hàm dựng sẵn của thư viện *scikit-learn*. Trong quá trình huấn luyện, chúng tôi đã chia dữ liệu ra một cách ngẫu nhiên, điều đó dẫn tới việc mỗi lần huấn luyện khác nhau mô hình lại cho một độ chính xác khác nhau. Do đó để tăng tính khách quan, chúng tôi đã sử dụng K-Fold với  $n\_splits$  bằng năm để đánh giá độ chính xác, đồng thời chúng tôi cũng đo lường thời gian thực thi của mô hình để so sánh hiệu suất của mô hình *Cây phân loại* với các mô hình khác trong thư viện *scikit-learn*.



```
[ ] # Create Model
DTsC = DecisionTreeClassifier()
vectorizer = TfidfVectorizer(stop_words='english', ngram_range=(1,3))

# Make pipeline
pipeline = make_pipeline(vectorizer, DTsC, verbose=True)

# K-Fold Cross Valuation with K = 5
t0=time.time()
scores = cross_val_score(pipeline, X, Y, cv= KFold(n_splits=5, shuffle=True, random_state=28))
t1=time.time()

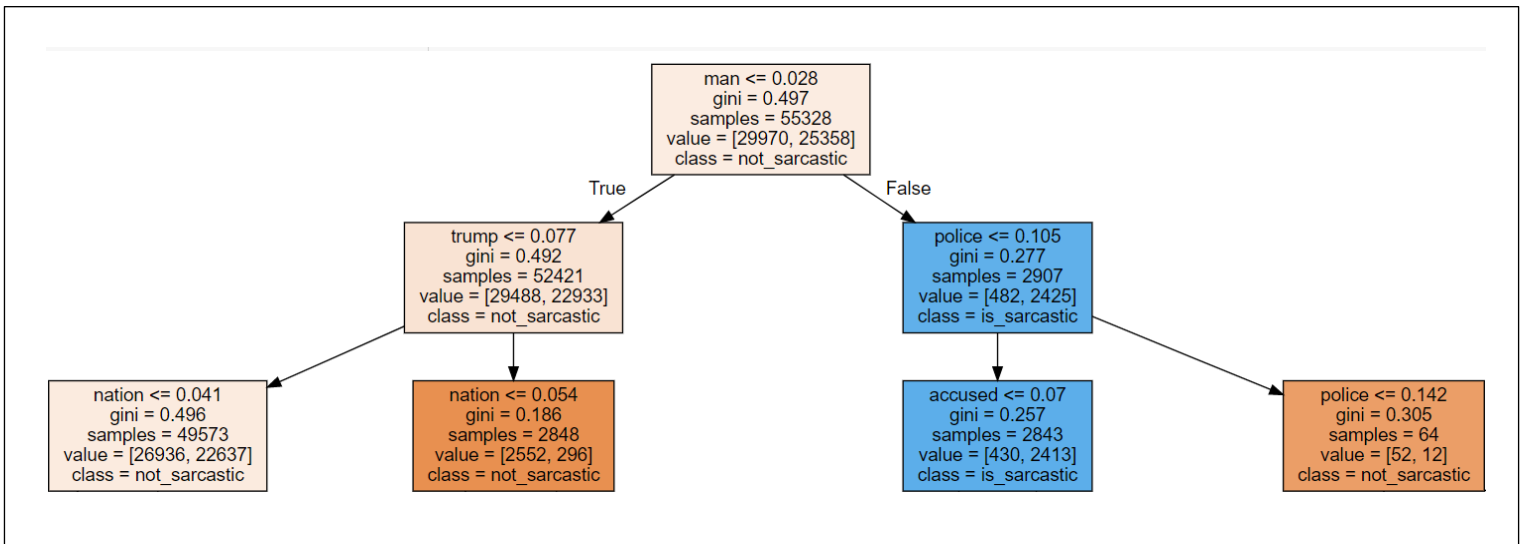
[Pipeline] ... (step 1 of 2) Processing tfidfvectorizer, total= 2.5s
[Pipeline] (step 2 of 2) Processing decisiontreeclassifier, total= 1.7min
[Pipeline] ... (step 1 of 2) Processing tfidfvectorizer, total= 2.4s
[Pipeline] (step 2 of 2) Processing decisiontreeclassifier, total= 1.6min
[Pipeline] ... (step 1 of 2) Processing tfidfvectorizer, total= 2.3s
[Pipeline] (step 2 of 2) Processing decisiontreeclassifier, total= 1.7min
[Pipeline] ... (step 1 of 2) Processing tfidfvectorizer, total= 2.3s
[Pipeline] (step 2 of 2) Processing decisiontreeclassifier, total= 1.6min
[Pipeline] ... (step 1 of 2) Processing tfidfvectorizer, total= 2.4s
[Pipeline] (step 2 of 2) Processing decisiontreeclassifier, total= 1.7min

[ ] print("Decision Tree Classifier K-fold accuracy:", scores.mean())
print("Time executed:", t1-t0)

Decision Tree Classifier K-fold accuracy: 0.9289329593649539
Time executed: 507.64650893211365
```

Hình 12: Kết quả đánh giá độ chính xác của mô hình sau khi huấn luyện

Sau khi huấn luyện xong mô hình, chúng ta có thể vẽ mô phỏng một *Cây quyết định* (như Hình 13), ứng với mỗi nút trong cây thể hiện sự ảnh hưởng của một từ trong tiêu đề bài báo đối với kết quả phân loại của mô hình.



Hình 13: Ảnh mô phỏng Cây quyết định sau khi huấn luyện mô hình

## 2.4 Thử nghiệm với các kỹ thuật khác

### 2.4.1. Thử nghiệm bằng phương pháp Grid Search

Trong quá trình huấn luyện mô hình, Nhóm chọn phương pháp Grid Search để thử nghiệm nhiều miền giá trị khác nhau của bộ tham số nhưng không có sự cải thiện rõ ràng hơn về độ chính xác cùng với thời gian chạy thuật toán. Đối với Grid Search, nhóm duyệt qua các tổ hợp giá trị trong miền cho trước.

Kết quả trả về cho thấy đối với bài toán đã nêu, bộ tham số mặc định mô hình *Cây phân loại* của thư viện *Scikit-learn* cho ra kết quả dự đoán với độ chính xác cao với tốc độ ổn định nhất. Điều này đạt được một phần là do bộ dữ liệu được thu thập từ các nguồn uy tín, không bị nhiễu và đã được thống kê xử lý tốt.

Độ chính xác đã tăng lên đáng kể (93% so với 83% của bài giải được nhiều bình chọn nhất trên Kaggle cho bộ dữ liệu) và không thể tăng cao hơn dựa trên sự thay đổi các tham số.



Về thời gian thực thi, có thể giảm thời gian thực thi của mô hình bằng việc thay đổi các tham số như đã nêu trên mục tham số nhưng phải đánh đổi bằng độ chính xác của kết quả. Tức giảm thời gian thực thi đồng nghĩa với giảm độ chính xác xuống một mức nhất định.

#### 2.4.2. So sánh thực nghiệm với các thuật toán khác

Để biết hiệu quả của mô hình so với các thuật toán khác, chúng tôi áp dụng bộ dữ liệu tương tự vào các mô hình máy học điển hình khác với bộ tham số mặc định.

Đặc điểm	Decision Tree	Naive Bayes	SVM	Logistic Regression
Độ chính xác (K-Fold)	0.93	0.95	0.95	0.91
Thời gian chạy 1 lần huấn luyện	102s	~0s	720s	6s
Thời gian tổng	507.65s	13.96s	3948.74	43.53s

Link bài code: [DecisionTreeClassifier.ipynb - Colaboratory \(google.com\)](#)

### III. Kết luận

#### 1. Nhận xét chung

*Cây quyết định* là một mô hình máy học có khả năng dự đoán với độ chính xác tương đối so với các thuật toán khác. Cùng với đó là thời gian thực thi của mô hình tương đối ổn định. Mô hình *Cây quyết định* được phổ biến rộng rãi nhờ vào các lý do khác nhau mà chúng tôi phân tích ở phần ưu nhược điểm bên dưới và được ứng dụng khá phổ biến như bài toán phân loại Thư rác, trong Y học (phân loại các loại bệnh), ...

#### 2. Ưu nhược điểm của thuật toán

Đối với mô hình *Cây quyết định*, chúng tôi rút ra được các ưu điểm sau đây:

- Dễ hiểu, dễ hình dung và áp dụng ngay cả trong đời sống. Có thể vẽ mô hình trực quan.
- Có thể giải quyết được cả hai loại bài toán *phân loại* (Classification) và bài toán *hồi quy* (Regression).
- Không cần bộ dữ liệu và chi phí tính toán quá lớn nhưng vẫn có thể hoạt động tốt.
- Phân loại được nhiều lớp khác nhau.
- Là mô hình *hộp trắng* (white box) nên có thể mô hình, quan sát, giải thích và thay đổi các giá trị.

Về nhược điểm, mô hình *Cây quyết định* rất dễ bị *quá khớp* (overfit), vì vậy khi huấn luyện mô hình dữ liệu cần phải được xử lý trước, tránh gây mất cân bằng và nhiễu trong bộ dữ liệu huấn luyện.

### **3. Kết luận**

Từ các đặc điểm trên, mô hình *Cây quyết định* thường được chọn sử dụng để phân tích và mô hình hóa dữ liệu, giải thích các tương tác của các đặc trưng trong bộ dữ liệu. Do đó mô hình *Cây quyết định* sẽ phù hợp với các bài toán phân tích dữ liệu mà vẫn đảm bảo được hiệu suất của mô hình, còn đối với các bài toán phân loại yêu cầu độ chính xác cao thì các mô hình phân loại khác như Support Vector Machine (SVM) thường sẽ được áp dụng phổ biến hơn.

## IV. Tài liệu tham khảo

1. (n.d.). Classification: Basic Concepts, Decision Trees, and Model Evaluation. Retrieved December 7, 2021, from <https://www-users.cse.umn.edu/~kumar001/dmbook/ch4.pdf>
2. Agarwal, S. (n.d.). *A report on Decision Tree, Random Forest and Deep Forest*. Summer Research Fellowship Programme of India's Science Academies 2017. Retrieved December 7, 2021, from <https://edu.authorcafe.com/academies/7920/a-report-on-decision-tree-random-forest-and-deep-forest>
3. Aznar, P. (2020, December 2). *Decision Trees: Gini vs Entropy*. Quantdare. Retrieved December 7, 2021, from <https://quantdare.com/decision-trees-gini-vs-entropy/>
4. *Cây Quyết Định (Decision Tree)*. (2019, June 6). Trí tuệ nhân tạo. Retrieved December 7, 2021, from <https://trituenhantao.io/kien-thuc/decision-tree/>
5. *Decision Trees: Complete Guide to Decision Tree Classifier*. (2019, December 10). Explorium. Retrieved December 7, 2021, from <https://www.explorium.ai/blog/the-complete-guide-to-decision-trees/>
6. *Decision Trees in Machine Learning / by Prashant Gupta*. (2017, May 17). Towards Data Science. Retrieved December 7, 2021, from <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>

7. 1.10. *Decision Trees — scikit-learn 1.0.1 documentation.*  
(n.d.). Scikit-learn. Retrieved December 7, 2021, from  
<https://scikit-learn.org/stable/modules/tree.html>
8. Seif, G. (n.d.). *A Guide to Decision Trees for Machine Learning and Data Science.* Towards Data Science.  
Retrieved December 7, 2021, from  
<https://towardsdatascience.com/a-guide-to-decision-trees-for-machine-learning-and-data-science-fe2607241956>