BÁO CÁO BÀI TẬP LỚN

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

Viện Công nghệ Thông tin và Truyền thông

****

Đề tài: **Phân loại thư sử dụng phương pháp RandomForest+AdaBoost**

|  |  |
| --- | --- |
| Sinh viên thực hiện: | **Đào Hoài Nam - 20173271**  **Nguyễn Khương Duy - 20173072**  **Nguyễn Kiên Trung- 20173421**  **Phạm Thành Đông-20173020** |
| Giáo viên hướng dẫn: | **Nguyễn Nhật Quang** |

Mục lục

[I. Introduce 3](#_Toc37894115)

[II. Detail 3](#_Toc37894116)

[1. Các phương pháp học máy và khai phá dữ liệu 3](#_Toc37894117)

[1.1. Outline 3](#_Toc37894118)

[1.2. Detail meachine learning algorithm and technique 4](#_Toc37894119)

[2. Các kết quả thí nghiệm đánh giá hiệu năng 9](#_Toc37894120)

[2.1. Đánh giá kết quả đạt được 9](#_Toc37894121)

[2.2. Đánh giá kết quả hiệu năng 10](#_Toc37894122)

[3. Các vấn đề gặp phải và cách giải quyết 11](#_Toc37894123)

[3.1. Data processing phase 11](#_Toc37894124)

[3.2. Data Visualize phase 12](#_Toc37894125)

[3.3. Training phase 12](#_Toc37894126)

[4. Các khám phá và các đề cử phát triển trong tương lai 13](#_Toc37894127)

[4.1. So sánh mô hình 13](#_Toc37894128)

[4.2. Đề xuất mới của nhóm 14](#_Toc37894129)

[4.3. Các hướng phát triển tương lai 14](#_Toc37894130)

[5. Tài liệu tham khảo 15](#_Toc37894131)

# Introduce

Đề tài mà nhóm đã giải quyết là bài toán phân loại thư điển tử. Yêu cầu cần đạt được là xác định các thư rác và chuyển vào hộp thư rác, ngược lại những thư bình thường, hợp lệ được gửi vào hộp thư chính để người nhận có thể đọc.

* Đầu vào: Biểu diễn nội dung của một email bằng 1 vector từ khóa
* Đầu ra: Nhãn 1- tương ứng với thư rác, nhãn 0- tương ứng thư thường.
* Tập dữ liệu được sử dụng: tâp dữ liệu thư rác và thư thường của kaggle, gồm có 2 trường text và class. Mỗi trường gồm 5726 diểm dữ liệu.
* Cách biểu diễn các ví dụ học : Học có giám sát.

# Detail

## Các phương pháp học máy và khai phá dữ liệu

### Outline

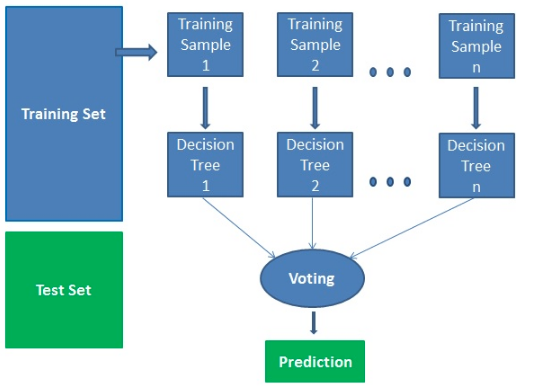
* Tiền xử lí dữ liệu
* Kiểm tra số trường dữ liệu
* Chọn vùng dữ liệu cần
* Kiểm tra có giá trị bị lặp lại
* Khiểm tra có giá trị null
* Xóa, sửa chữa các nhiễu lỗi(missing value, null value, label sai)
* Lowercase tất cả các từ có trong trường text, nội dung của email
* Loại bỏ các từ stopwrods
* Sử dụng kĩ thuật stemming để biến đổi từ về dạng gốc
* Suffle dữ liệu.
* Vector hóa dữ liệu
* Khám phá đăc tính mô tả dữ liệu
* Phân bố lớp của nhãn dữ liệu
* Phân bổ lớp của các tập training set và test set
* Hiển thị hóa dữ liệu
* Hiển thị hóa bằng biểu đồ bar
* Giải thuật và kĩ thuật học máy sử dụng
* RandomForest
* Hyperparameter Tuning by GridSearch
* PipeLine to speed up Tuning
* Adaboost to power up prediction

### Detail meachine learning algorithm and technique

#### Randomforest

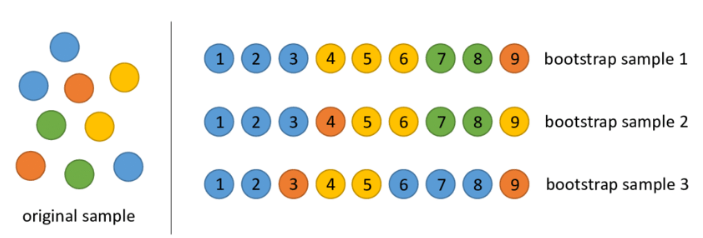
Random Forest là thuật toán học có giám sát. Thuật toán học này được đánh giá là thuật toán linh hoạt và mạnh mẽ. Nằm trong nhóm thuật toán ensembling, Random Forest đưa ra các dự đoán bằng cách dựa trên dự đoán của rất nhiều cây quyết định. Các cây quyết định này có thể có mạnh yếu khác nhau, nhưng nguyên tắc < sự khôn ngoan của đám đông > sẽ cho ta mô hình phân loại chính xác hơn khi sử dụng bất kì mô hình đơn lẻ nào.

* Các bước hoạt động của thuật toán
* Chọn ngẫu nhiên các mẫu từ training set
* Thiết lập cây quyết định cho từng mẫu
* Nhận kết quả dự đoán từ mỗi cây quyết định
* Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng



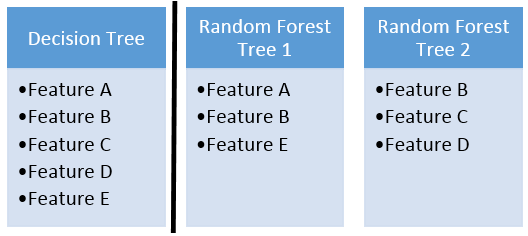
Sự thành công của Random forest phụ thuộc nhiều vào việc sử dụng các cây quyết định không có mối tương quan. Nếu các cây quyết định có độ tương đồng lớn, thì kết quả sau cùng sẽ không khác mấy kết quả của một 1 cây quyết định. Để làm được điều này Random Forest sử dụng boostrapping and feature randomness.

* Boostrapping : randomly lựa chọn các mẫu từ tập training set cùng sự thay thế, sự thay thế ở đây có thể là thay đổi phân bố lớp cho từng cây quyết định, một lớp có thể không thế không xuất hiện hoặc được xuất nhiều trong mẫu.



Fingure [[source](https://www.researchgate.net/publication/322179244_Data_Mining_Accuracy_and_Error_Measures_for_Classification_and_Prediction/figures?lo=1)](https://www.researchgate.net/figure/An-example-of-bootstrap-sampling-Since-objects-are-subsampled-with-replacement-some_fig2_322179244)

* Feature randomness: chọn ngẫu nhiên các thuộc tính cho từng cây quyết định có trong rừng. Số feature tối đa được sử dụng các cây được chỉ định trong parameter **max\_feature**



Fingure [source](https://towardsdatascience.com/decision-tree-and-random-forest-explained-8d20ddabc9dd)

* Ưu điểm
* Random Forest sử dụng một số lượng lớn cây quyết định( 1000 cây trong implement). Chính vì thế mô hình tránh bị vấn đề overfitting
* Random Forest có thể xử lí được cả các giá trị còn thiếu
* Ý tưởng dể hiểu
* Độ chính xác cao, là một trong các thuật toán học máy hiện đại và cho kết quả rất cao trong các bài toán phân loại so với các phương pháp truyền thống(Naïve Bayes, kNN, SVM..)
* Nhược điểm:
* Có thể chậm đưa ra các dự đoán, vì kết quả dự đoán được đưa ra dựa trên quá trình vote của các cây quyết định, tốn nhiều thời gian
* Việc diễn giải quá trình thực hiện của thuật toán không quá dễ hiểu( nhất là với khách hàng không biết gì về công nghệ)

#### Hyperparameter Tuning by GridSearch

Phương pháp phân loại bằng Random Forest(RF) sử dụng các hyperparameters trong quá thọc. Việc tinh chỉnh các hyperparameters này sẽ ảnh hưởng khác nhau đến quá trình học của mô hình. Để tinh chỉnh các hyperparameters này có 2 hướng, đó là dùng Grid search và Random search. Trong quá trình implement, nhóm lựa chọn Grid search để tuning các hyperparameter của mô hình.

Có 4 hyperparameters mà nhóm đã tinh chỉnh đó là:

* max\_depth: Hyperparamter này chỉ định maximum depth của từng cây, default value là None, tức mỗi cây sẽ mở rộng đến khi lá là nguyên chất hoặc cho đến khi tất cả các lá chứa ít mẫu hơn min\_samples\_spit. Nhóm thực hiện Grid search hyperparameter này với 3 giá trị [50, 100, None].
* class\_weight: trọng số của từng lớp, hyperparameter này được tính dựa trên số lượng của từng lớp, default value là ‘balance’, ở chế độ này mô hình sẽ tránh được sự mất cân bằng lớp. Grid search hyperparameter này với 4 giá trị : [{0:0.9, 1:1}, {0:1, 1:0.9}, {0:1, 1:5},"balanced"].
* min\_samples\_leaft: Hyperparameter này chỉ định số lượng mẫu tối thiểu cần có ở 1 nút lá. Grid search hyperparameter này với 2 giá trị [3, 5].
* min\_samples\_split: Hyperparameter này chỉ định số lượng sample tối thiểu để phân chia 1 node nội bộ, Grid search hyperparameter này với 3 giá trị [2, 3, 5].

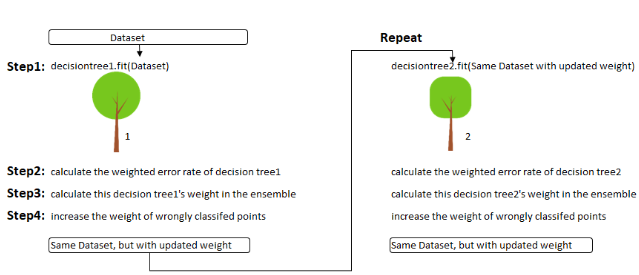
#### Pipeline to speed up Tuning

Grid search tinh chỉnh các hyperparameter, chọn ra các hyperparamter làm một giá trị nào đó đạt giá trị cực đại(nhóm lựa chọn f1-score). Quá trình search này tốn rất nhiều thời gian với các bước lặp lại đi lặp lại. Để giải quyết vấn đề này, nhóm sử dụng Pipe Line . Scikit-learni pipelines là 1 tool nhằm đơn giản hóa quá trình này. Lí do sử dụng pipeline là vì các lợi ích nó mang lại:

* Khiến workflow dễ để đọc và hiểu
* Sắp xếp và thi hành các bước
* Tái sử dụng cáccông việc đã được thực thi.

#### Adaboost to power up prediction

Adaboost là mô hình boosting ensemble. Adaboost làm việc đặc biệt hiệu quả với cây quyết định. Chìa khóa của mô hình boosting là việc học từ các sai lầm trước. Việc học từ sai lầm trước bằng cách gia tăng trọng số cho các điểm dữ liệu bị phân loại sai.



Fingure [source](https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725)

Các bước thực hiện của thuật toán :

* Khởi tạo trong số cho các điểm dữ liệu, tất cả các điểm dữ liệu sẽ có trọng số bằng nhau và bằng 1/n, với n là số điểm dữ liệu.
* Train 1 cây quyết định
* Tính trọng số lỗi err của cây quyết định. Trọng số lỗi err chính là tỉ lệ số điểm dữ liệu sai trên tổng số điểm dữ liệu đúng. Nếu trọng số càng cao thì tỉ lệ lỗi tương ứng càng lớn.
* Tính toán trọng số cây. Trọng số này được tính bới công thức:

the weight error of this tree = learning rate \* log( (1 — err) / err)

Dựa trên công thức này, ta có thể thấy cây có trọng số lỗi cao thì cây đó càng có ít quyền vote sau này, ngược lại cây có trọng số lỗi thấp thì cây đó càng có sức mạnh quyết địn trong giai đoạn vote sau này.

* Cập nhật trọng số lỗi cho các điểm dữ liệu bị dự đoán sai, trọng số mới được tính dựa trên công thức:

new weight = old weight\*

Sau khi cập nhật trọng số cho các điểm dữ liệu sai, trọng số của tất cả các điểm dữ liệu sẽ được chuẩn hóa.

* Lặp lại các bước trên cho đến khi tất cả các cây quyết đinh được duyệt
* Đưa ra dự đoán: ở bước này, adaboost cộng trọng lượng của mỗi cây nhân với dự đoán tương ứng của cây đấy. Rõ ràng là cây có trọng số lỗi ít tức trọng số cây cao sẽ có sức ảnh hưởng lớn hơn các cây khác.

Lưu ý: Ở trong thuật toán Adaboost nhóm đã sử dụng Random Forest là mô hình con trong mô hình adaboost thay vì là Decision Tree như mặc định. Lí do của việc lựa chọn này xuất phát từ một số lí do sau :

* Thứ nhất, tại sao không thử 1 hướng đi mới thay vì cách làm thông thường?
* Thứ 2, về trực quan thì việc dựa trên sức mạnh vote của rừng cây, trong mỗi rừng lại là rất nhiều cây so với việc dựa trên sức mạnh vote của các cây thông thường cho ta cảm giác mạnh hơn
* Thứ 3, nhóm đã thử thực thi và đưa ra đánh giá là đúng như giả thuyết đã đề ra.

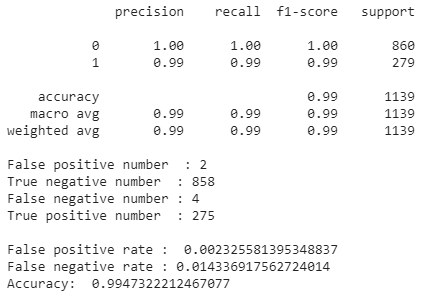
## Các kết quả thí nghiệm đánh giá hiệu năng

Trong phần 2, nhóm sẽ trình bày về kết quả và đánh giá hiệu năng đã đạt được.

### Đánh giá kết quả đạt được

Về kết quả nhóm đánh giá dựa trên chỉ số f1-score. Thay vì đánh giá trên chỉ số precision, đánh giá này được chọn là vì sự mất cân bằng lớp trong tập dữ liệu. Với sự mất cân bằng về lớp, f1-score thường được đưa ra như là chỉ số đánh giá.

Kết quả đạt được như sau:



Accuracy : 99.47%

F1-score lần lượt là : 100% và 99%

False positive rate : 0.23%

False negative rate : 1.4%

Rõ ràng là False positive rate(FPR) tuy thấp chưa đến 0.25% có thể chấp nhận được, nhưng chưa phải là mức “tối ưu” vì nếu như hàm mục tiêu là precision thì FPR có thể xấp xỉ 0% tức là chỉ có 0 hoặc 1 email hợp lệ bị gán cho email cho thường. FNR ở mức chấp nhận được dưới 1.5%.

Tuy kết quả đạt được khá khả quan, nhưng có 3 điểm mà nhóm chưa thấy được sự confidence trong kết quả có được, đó là :

* Liệu rằng mô hình của nhóm có mang tính tống quát hay không? Với các email spam thuộc các domain mới như : buôn bán cần sa, hay chat sex (trong tập dữ liệu train chưa có) thì liệu rằng mô hình có dự đoán đúng hay không, và độ chính xác thế nào?
* Với cùng các domain đã được train trong tập dữ liệu, spammer có thể thay đổi các cách thức trong cấu trúc cũng như từ vựng lẫn cố tình làm sai lệch các từ. Ví dụ : buy for you -> by4y, etc.
* Trong mô hình nhóm đã sử dụng 1000 rừng trong adaboost, có thể nghĩ rằng càng nhiều rừng thì việc quyết định càng chính xác, nhưng liệu rằng điều đó có chính xác không, khi có thể các rừng không khác nhau mấy, như thế training phrase sẽ tốn thời gian và lãng phí tài nguyên không cần thiết.

### Đánh giá kết quả hiệu năng

* Trong quá trình huấn luyện

Một cách tổng thế, quá trình huấn luyện của mạng là lâu so với các mô hình khác( sự so sánh sẽ nói rõ ở phần các khám phá)

Điều này rất dễ lí giải. Trong quá trình huấn luyện nhóm nhận thấy có 4 bước khiến việc huấn luyện tốn thời gian và tài nguyên, đó là:

* Đưa các từ về dạng gốc stemming (1)
* Sử dụng randomforest (2)
* Dùng Grid serach để tinh chỉnh hyperparameter (3)
* Dùng adaboost để cải thiện sự chính xác (4)

1. Sử dụng thư viện nltk để đưa các từ về dạng gốc, quá trình này là bước tiền xử lí, ta cần phải quét qua tất cả các từ và thay thế, với tập dữ liệu sử dụng quá trình này tiêu tốn mất khoảng 3s(GPU Google Colab)
2. Việc sử dụng random forest sẽ lâu hơn việc sử dụng 1 cây quyết đinh thông thường, điều này là rất hiển nhiên, rừng càng nhiều cây quá trình train càng tốn nhiều thời gian
3. Grid search khiến hiệu năng giảm đi rất nhiều( khoảng 5mins) cho việc tinh chỉnh hyperparameters
4. Dùng adaboost khiến hiệu năng giảm đi hơn cả( 10 mins) ở đây nhóm sử dụng 1000 rừng cây quyết định! trong adaboost.

* Trong quá trình dự đoán

Việc đưa ra dự đoán được diễn ra nhanh hơn so với quá trình huấn luyện, cụ thể để đưa ra dự đoán chỉ mất khoảng 5s, một mức chấp nhận được.

## Các vấn đề gặp phải và cách giải quyết

Vấn đề gặp phải trong quá trình thực thi là rất nhiều, nhưng nhóm cũng đưa ra được các giải quyết các vấn đề đó.

Nhóm chia ra thành các phase tương ứng với quá trình implement

* Data processing phase
* Data Visualize phase
* Training phase

### Data processing phase

Ngay từ giai đoạn này, nhóm gặp những vấn đề rất thú vị và cũng đau đầu:

* Khi tải lên dữ liệu google colab( hoặc jupyter notebook) đều gặp cùng 1 lỗi, đó là không thể mã hóa kí tự có trong email.
* Hướng giải quyết: sau khi tìm hiểu, thì nhóm phát hiện ra lỗi xuất hiện là vì UTF-8 không thể mã hóa 1 số kí tự latin! Khi tải dữ liệu lên, thì mặc định các kí tự được mã hóa ở dạng UTF8, do đó để giải quyết nhóm sử dụng mã hóa latin/ latin-1. Vấn đề được giải quyết!
* Khi vấn đề trên được giải quyết, thì nhóm lại gặp vấn đề khác, khi dữ liệu tải lên, có một vài hàng của trường text lại kéo dài sang cả trường class, và lớp class đó không được gán nhãn, mà thay vào đó là dữ liệu dạng string.
* Hướng giải quyết : Xem trong dữ liệu và xóa đi các điểm dữ liệu bị lỗi. Nhóm lựa chọn xóa là vì chỉ có chưa đến 5 điểm dữ liệu, số lượng này là rất nhỏ không đáng kể với tập dữ liệu.
* Sau khi tải dữ liệu lên, và xem 1 vài điểm dữ liệu ở đầu, nhóm nhận thấy từ 2 trường gốc(class, text) đã có thêm tới 10 trường NaN nữa
* Hướng giải quyết : Chọn lấy 2 trường cần đó là text và class

### Data Visualize phase

Việc visualize dữ liệu nhóm gặp chút khó khăn khi trong nhóm không có bạn nào biết nhiều về việc visuaize dữ liệu, để giải quyết vấn đề này nhóm đã tham khảo các bài viết về việc visualize dữ liệu, từ đó lựa chọn biểu đồ bar chart. Biểu đồ bar chart là phù hợp để biểu thị phân bổ lớp trong lớp bài toán phân loại.

### Training phase

Ở phase này, nhóm sẽ trình bày 2 khó khăn chính mà nhóm gặp phải:

* Thời gian trainning
* Độ chính xác

#### Thời gian training

Đây là vấn đề kinh điển với các giải thuật học máy thuộc nhóm Eager learning.

Giải thuật học máy mà nhóm chọn là RandomForest kết hợp Adaboost.

* Giải thuật RandomForest đưa ra dự đoán bằng vote của các cây quyết định. Các cây quyết định càng nhiều và càng khác nhau thì cho dự đoán càng chính xác. Rõ ràng có sự trade-off ở đây. Việc phải tạo ra cây quyết định lấy vote sẽ khiến quá trình train lâu , khi kết hợp tuning parameter còn lâu hơn hơn.
* Giải quyết: Sử dụng PipeLine làm tối ưu hóa quá trình tunning hyperparameter của RandomForest.Lợi ích của dùng PipelIne nhóm đã trinh bày ở phần trên.
* Giải thuật AdaBoost đưa ra dự đoán cũng là trên việc vote(vote khác với Random Forest), training với base model là Decision Tree đã lâu, nhưng nhóm train với base model là Random Forest còn lâu hơn.
* Giải quyết : Thử nghiệm và chọn ra số rừng cây phù hợp để thời gian training không quá lâu mà cũng làm giảm độ chính xác. Nhóm thử nghiệm Adaboost với 1000 rừng mất 10 mins và 5000 rừng gần 40 mins. 2 thử nghiệm cho kết quả giống nhau về các chỉ số. Điều này được lí giải khi số lượng rừng là bão hòa, các rừng giống nhau về các thuộc tính nên nhiều vote của rừng cũng chỉ là 1 vote của rừng gốc.

#### Độ chính xác

Vấn đề trong bài toán này cần đánh giá mức độ FPR và FNR là khác nhau. Tức ta quan tâm đến tỉ lệ precision cao, nhưng tỉ lệ này là không đủ.

* Với thử nghiệm của nhóm khi tỉ lệ precision cao, tức FPR rất nhỏ, xấp xỉ 0%, thì điều này khiến việc các thư rác bị bỏ vào thùng thư hợp lệ xảy ra nhiều hơn. Điều này còn có thể xảy ra vì sự mất cần bằng lớp
* Giải quyết : Thay vì sử dụng độ đo là precison, nhóm sử dụng độ đo f1-score. Việc sử dụng độ đo này nhằm thỏa mãn cả 2 mục đích tối thiểu hóa FNR và FPR trong 1 mức chấp nhận được.
* Làm sao để cải thiện độ chính xác của mô hình?
* Hướng giải quyết: Tinh chỉnh các hyperparameter. Tinh chinh hyperparameter có thể đem lại các kết quả dự đoán và độ chính xác khác nhau, vì thế nhóm sử dụng Gridsearch để tinh chỉnh các hyper parameter nhằm chọn ra các tham số để độ đo f1-score đạt điểm cao nhất trên tập train.
* Thay đổi thresold để phân loại! Việc thay đổi thresold sẽ được trình bày ở mục các khám phá.

## Các khám phá và các đề cử phát triển trong tương lai

Ở phần này nhóm trình bày về các khám phá trong quá trình giải quyết vấn đề này.

Như đã đề cập ở phần trước, nhóm sẽ so sánh một số mô hình học máy truyền thống và mô hình mà nhóm sử dụng để giải quyết bài toán phân loại thư, cũng như nêu ra 1 số cải tiến của nhóm trong tương lại.

### So sánh mô hình

Nhóm đã thử nghiệm 1 số mô hình nhằm chọn ra mô hình phù hợp.

Trong quá trình thực hiện nhóm đã thử nghiệm mô hình phân loại Naïve Bayes. Kết quả so sánh FPR, FNR lần lượt là 1.27% và 0.36%. Kết quả này là chấp nhận được trong thời gian train vừa phải.

Ưu điểm của mô hình này là việc huấn luyện va dự đoán nhanh hơn so với mô hình nhóm đã sử dụng. Tuy nhiên khi số lượng các từ nhiều lên thì việc tính xác suất không còn dễ dàng, với sự phân bố lệch, các email spam ít thì các tập từ khóa có trong spam email có thể là không đủ.

### Đề xuất mới của nhóm

Trong quá trình implement, nhóm đã đề xuất được nhiều ý tưởng hay.

Ban đầu nhóm đề xuất dùng sentiment analysis như là 1 heuristic để phân loại bổ sung cho mô hình, nhưng khi phân tích thì nhóm đã quyết định không dùng đến heuristic này, vì chưa có một thư viện đủ tốt để áp dụng( TextBlob chỉ dùng để phân loại tiêu cực hay tích cực ).

Nhưng nhóm đã đề xuất 1 phát hiện mới, theo nghiên cứu của nhóm là chưa được đề xuất trên các paper nào.

Trong bài toán phân loại thư, thì việc thư thường bị cho vào thư rác sẽ gây khó chịu hơn việc thư rác bị cho thư thường. Nhưng chọn thresold như nào là tốt( trade off chấp nhận được) ?

Qua thực nghiệm, nhóm đề xuất 1 công thức để chọn thresold cho tập test. Đó là

*Test\_thresold = train\_thresold / label\_rate\_train \* label\_rate\_test*

Việc lựa chọn train\_thresold là do thực nghiệm.

### Các hướng phát triển tương lai

* Ý tưởng của nhóm là kết hợp dự đoán của nhiều mô hình.

Đó là việc dự đoán dựa trên đầu ra của mỗi mô hình. Với mô hình random forest tree sau khi kết hơp với Adaboost sẽ có trọng số vote là 0.45, của Naïve Bayes là 0.3, và của SVM là 0.25.

Việc kết hợp này có thể thay đổi trên từng lớp, giả sử mô hình Random Forest dự đoán cho tỉ lệ FPR cao, FNR thấp, ngược lại Naïve Bayes cho FPR thấp , FNR cao thì trọng số của các mô hình cũng thay đổi theo.

* Ý tưởng để phát triển tiếp theo của nhóm đó là sử dụng AutoML, cụ thể là thư viện autoscikietlearn của Scikitlearn để phân lọa có tính chính xác cao hơn, tuy nhiên với automl thì việc giải thích kết quả cho các stakeholder khó khăn hơn hi sử dụng giải thuật như cây quyết định hay Random Forest.
* Một hướng tiếp cận hiện đại nhưng sẽ tốn nhiều thời gian để tìm hiểu nhưng cũng rất thú vị đó là sử dụng mạng mô hình sinh giống mạng GAN. Mô hình này là mô hình dựa trên mạng NN có tính khả quan cao và có tính ứng dụng rộng rãi. Cụ thể sẽ có 1 mạng học các sinh ra email rác và 1 mạng còn lại sẽ cố gắng phân loại thư rác, quá trình học sẽ kết thúc khi đạt trạng thái bão hòa. Tức là khi mô hình phân biệt không còn nhận biết được đâu là thư rác và đâu là thư thường. Với lượng dữ liệu khổng lồ thì việc train cho mạng tạo thư rác sẽ đạt hiệu quả cao.
* Có thể kết hợp các phương pháp truyền thống như lọc thư dựa vào tên miền của địa chỉ emai, thường những tên miền này chứa các kí tự linh tinh hoặc là những tên miền mang tính thương mại.

## Tài liệu tham khảo

Scikit learn document.

Random forest:

<https://towardsdatascience.com/decision-tree-and-random-forest-explained-8d20ddabc9dd>

Hiểu về Random Forest:

<https://towardsdatascience.com/understanding-random-forest-58381e0602d2>

Tối ưu tham số trong Random Forest:

<https://towardsdatascience.com/optimizing-hyperparameters-in-random-forest-classification-ec7741f9d3f6>

Kết hợp Random Forest và PipeLine:

<https://scikit-learn.org/stable/tutorial/statistical_inference/putting_together.html>

Giới thiệu về các mô hình ensemble:

<https://towardsdatascience.com/basic-ensemble-learning-random-forest-adaboost-gradient-boosting-step-by-step-explained-95d49d1e2725>

Pipeline trong scikit learn:

<https://medium.com/vickdata/a-simple-guide-to-scikit-learn-pipelines-4ac0d974bdcf>

Hà Nội 16/4/2020