# Xử lý Dataset

Link Dataset để Train model: <https://github.com/MWiechmann/enron_spam_data/blob/master/enron_spam_data.zip>

File name: Train\_code\_spam.ipynb: đọc, xử lý dữ liệu trong dataset (remove null column, maping features), phân tích độ dài đoạn message, phân tích phân phối spam/ham theo từ, câu, kí tự

A graph of a distribution of characters

Description automatically generated

A diagram of a number of words

Description automatically generated

A graph of a number of sentences

Description automatically generated

Đoạn funtion clean message :

1. def clean\_mess(text):

2. # Remove special characters and digits

3. text = re.sub(r"[^a-zA-Z]", " ", text)

4.

5. # Remove punctuation

6. text = text.translate(str.maketrans("", "", string.punctuation))

7.

8. # Remove emojis

9. emoji\_pattern = re.compile("["

10. u"\U0001F600-\U0001F64F" # emoticons

11. u"\U0001F300-\U0001F5FF" # symbols & pictographs

12. u"\U0001F680-\U0001F6FF" # transport & map symbols

13. u"\U0001F1E0-\U0001F1FF" # flags (iOS)

14. u"\U00002702-\U000027B0"

15. u"\U000024C2-\U0001F251"

16. "]+", flags=re.UNICODE)

17. text = emoji\_pattern.sub(r'', text)

18.

19. # Convert to lowercase

20. text = text.lower()

21.

22. # Remove stop words

23. stop\_words = set(stopwords.words('english'))

24. tokens = nltk.word\_tokenize(text)

25. tokens = [word for word in tokens if word not in stop\_words]

26.

27. return ' '.join(tokens)

28.

Sử dụng để xóa biểu tượng, emotion, xóa stop words, chuyển về in thường.

Tách Dataset đã được xử lý thành các dữ liệu Train và Test dùng thư viện train\_test\_split và lưu thành file, lưu tf vector

1. # X, Y Variable and convert a collection of raw documents to a matrix

2. tf=TfidfVectorizer(max\_features = 3000)

3.

4. X=tf.fit\_transform(data['convert\_mess'])

5. Y=data[['Label',]]

6.

7. # train and test subsets

8. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y, train\_size=0.8, test\_size=0.2, random\_state=0)

9.

10. # print(type(X\_train))

11.

12. X\_train = csr\_matrix(X\_train)

13. X\_test = csr\_matrix(X\_test)

14.

15. # print(type(X\_train))

16. #print(np.asarray(X\_train))

17.

18. X\_train = np.asarray(X\_train.toarray())

19. X\_test = np.asarray(X\_test.toarray())

20. print(X\_train.shape, X\_test.shape)

21. print(Y\_train.shape, Y\_test.shape)

22.

23. # print(type(X\_train))

24. # print(type(Y\_train))

25.

26. # Save X\_train, X\_test, Y\_train, Y\_test

27. np.save('x\_train.npy',X\_train)

28. np.save('y\_train.npy',Y\_train)

29. np.save('x\_test.npy',X\_test)

30. np.save('y\_test.npy',Y\_test)

31.

* Thử nghiệm với Decision Tree Model

A computer screen shot of a computer program

Description automatically generated

# Phân tích và chọn lọc Model để sử dụng

## Đánh giá bình thường

File name: Select\_Model.ipynb

Sử dụng các model như:

DT: Decision Tree

MNB: Multinomial Navie Bayes

LR: Logistic Regression

SVM: Suppost Vector Machine

RF: Random Forest

NN: Neural Network

* Kết quả của phân tích

Độ chính xác training

A graph of a number of blue bars

Description automatically generated

Độ chính xác từ testing

A graph of blue bars

Description automatically generated

Thời gian training

A diagram of a model

Description automatically generated with medium confidence

Thời gian tesing

A diagram of a model

Description automatically generated

\*\*Thời gian Testing và training lấy từ dataset với tổng là hơn 30000 đoạn message (train 80%, test 20%)\*\*

### Đánh giá cuối

Lựa chọn SVM với độ chính xác cao nhất để xử lý mặc dù thời gian train và test khá lâu (dữ liệu lớn)

Confusion matrix SVM Model

A graph of numbers and spam

Description automatically generated with medium confidence

0.98 phần trăm đoán chính xác HAM

0.021 phần trăm đoán sai HAM

0.0066 phần trăm đoán sai SPAM

0.99 phần trăm đoán đúng SPAM

Lưu lại Model

## Đánh giá nhanh trên nhiều Model

File name: Select\_Model\_v2.ipynb

Các loại model sử dụng

A screen shot of a computer program

Description automatically generated

Đánh giá:

A graph of blue bars

Description automatically generated

A graph of blue bars

Description automatically generated

# Bash file để đọc các Message được lưu

Bash run.sh

1. #!/bin/bash

2.

3. INCOMING\_FOLDER="/opt/zimbra/store/0"

4.

5. COPY\_FOLDER="/home/hlog/Desktop/processing"

6.

7. inotifywait -m -r -e create --format "%w%f" "$INCOMING\_FOLDER" |

8. while read file; do

9. if [ -f "$file" ]; then

10. if [[ $(basename "$file") > "7000-" && $(basename "$file") < \*.msg ]]; then

11. echo "admin notification"

12. else

13. if grep -q "X-Spam-Flag" "$file"; then

14. cp "$file" "$COPY\_FOLDER"

15. fi

16. fi

17. fi

18. done

19.

Dùng inotifywait để chờ file mới được tạo vào INCOMING\_FOLDER rồi xử lý <https://linux.die.net/man/1/inotifywait> : man

Với file name dạng 7000-xxxx.msg là message thông báo của admin nên bỏ qua

Chuyển file bắt được vào COPY\_FOLDER

Bash file dùng ML để phát hiện spam mail

Bash detection.sh

1. #!/bin/bash

2.

3. CAPTURE\_FOLDER="/home/hlog/Desktop/processing"

4.

5. SAVE\_FOLDER="/home/hlog/Desktop/result"

6.

7. inotifywait -m -e create "$CAPTURE\_FOLDER" |

8. while read file; do

9. python3 processing.py "$file" >> "$SAVE\_FOLDER/log.txt"

10. echo "=========================================" >> "$SAVE\_FOLDER/log.txt"

11. done

12.

Python processing.py

1. from joblib import load

2. import sys

3. import numpy as np

4. from sklearn.feature\_extraction.text import TfidfVectorizer

5. import re

6. import string

7. import nltk

8. from nltk.corpus import stopwords

9.

10. #nltk.download('stopwords')

11.

12. arg = sys.argv[1]

13.

14. def clean\_mess(text):

15. # Remove special characters and digits

16. text = re.sub(r"[^a-zA-Z]", " ", text)

17.

18. # Remove punctuation

19. text = text.translate(str.maketrans("", "", string.punctuation))

20.

21. # Remove emojis

22. emoji\_pattern = re.compile("["

23. u"\U0001F600-\U0001F64F" # emoticons

24. u"\U0001F300-\U0001F5FF" # symbols & pictographs

25. u"\U0001F680-\U0001F6FF" # transport & map symbols

26. u"\U0001F1E0-\U0001F1FF" # flags (iOS)

27. u"\U00002702-\U000027B0"

28. u"\U000024C2-\U0001F251"

29. "]+", flags=re.UNICODE)

30. text = emoji\_pattern.sub(r'', text)

31.

32. # Convert to lowercase

33. text = text.lower()

34.

35. # Remove stop words

36. stop\_words = set(stopwords.words('english'))

37. tokens = nltk.word\_tokenize(text)

38. tokens = [word for word in tokens if word not in stop\_words]

39.

40. return ' '.join(tokens)

41.

42. with open(arg, 'r') as f:

43. email = f.read()

44.

45. from\_address = re.search(r'From: (.+)', email).group(1)

46. to\_address = re.search(r'To: (.+)', email).group(1)

47. date\_get = re.search(r'Date: (.+)', email).group(1)

48. subject = re.search(r'Subject: (.+)', email).group(1)

49. message\_id = re.search(r'Message-ID: (.+)', email).group(1)

50.

51. print("From:", from\_address)

52. print("To:", to\_address)

53. print("Date:", date\_get)

54. print("Subject:", subject)

55. print("Message-ID:", message\_id)

56.

57. # doan message

58. # --=\_7919a7cb-4852-47ca-9734-03cf0b12eb9c

59. # Content-Type: text/plain; charset=utf-8

60. # Content-Transfer-Encoding: 7bit

61. #

62. # khong con gi de mat.

63. # dong 2 mesage

64. # dong message

65. #

66. # --=\_7919a7cb-4852-47ca-9734-03cf0b12eb9c

67. # Content-Type: text/html; charset=utf-8

68. # Content-Transfer-Encoding: 7bit

69.

70. pattern = r'Content-Type: text/plain; charset=utf-8\nContent-Transfer-Encoding: 7bit\n\n(.\*?)\n\n--='

71. # "." khop voi bat ki ki tu nao ngoai tru (/n)

72. # "\*" co the xuat hien 0 hoac nhieu lan

73. # "?" co khong hoac 1 ki tu dang sau

74. # ".\*" greedy quantifier khop voi chuoi dau den chuoi cuoi (neu chi co 1 chuoi duy nhat giong voi ".\*?")

75. # ".\*?" reluctant or "non-greedy" quantifier khop voi chuoi dau

76.

77. match = re.search(pattern, email, re.DOTALL)

78. # re.DOTALL khop voi bat ki ki tu nao ke ca dau (.) va dau xuong dong

79.

80. if match:

81. email\_msg = match.group(1).strip()

82. else:

83. email\_msg = ""

84.

85. print("Message:\n", email\_msg)

86.

87. email\_msg = subject + " " + email\_msg

88.

89.

90. ms = [""]

91.

92. #ms.append(clean\_mess(email\_msg))

93. ms = [clean\_mess(email\_msg)]

94. #print(type(ms))

95.

96. # load the vectorizer

97. loaded\_vectorizer = load(open('vectorizer.joblib', 'rb'))

98.

99. # load the model

100. svm\_model = load(open('dump\_svm\_model.joblib', 'rb'))

101.

102.

103. data\_transformed = loaded\_vectorizer.transform(ms).toarray()

104.

105. predictions = svm\_model.predict(data\_transformed)

106.

107. if predictions==0:

108. print("Status Predict: : Ham")

109. else:

110. print("Status Predict: : Spam")

111.

File python đọc và tách những thông tin trong mail dùng regex và thêm model dự đoán spam hoặc ham

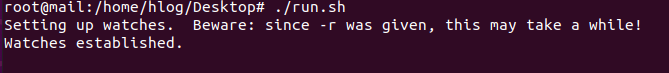
## Kết quả cho ra với test\_msg.msg file

A screen shot of a computer

Description automatically generated

# Kết quả áp dụng vào mail Zimbra

Chạy 2 file run.sh và detection.sh

 A black background with white text

Description automatically generated

Thử nghiệm với hai đoạn mail spam thử nghiệm và cat log.txt:

A computer screen shot of a computer code

Description automatically generated

A computer screen shot of text

Description automatically generated

Thử nghiệm 1 đoạn mail HAM và cat log.txt:

A screen shot of a computer

Description automatically generated

# Yếu điểm của bài nghiên cứu

Chỉ áp dụng phát hiện cho những mail bằng tiếng anh, khi dùng tiếng việt để mail thì mặc định được phát hiện là spam.

Giải quyết: bổ xung các dataset với mail bằng tiếng việt

Chưa xử lý được trước khi gửi đến người nhận (chỉ bắt được mail sau khi đã gửi)