Classifying Emotion in Text-Based Content

Hicran Arnold

NLP - Emotion Classification

- The exponential growth of digital communication has resulted in a massive amount of text-based content that expresses a wide range of emotions. Understanding the emotions convey in this content is essential for businesses, organizations, and individuals to understand their audiences better and gauge public sentiment about their products and services.
- However, manual analysis of these texts is time-consuming and impractical, making it necessary to develop automated methods for emotion classification.
- This project aims to develop a machine learning model for emotion classification in text-based content, which will provide a more efficient and accurate method for analyzing the emotions conveyed in digital communication.

Project Focus Areas

1- Can a machine learning model accurately classify different types of emotions expressed in text-based content, and if so, which model is most effective?

2- What is the expected accuracy of the machine learning model in classifying different types of emotion based on text-based content?

3- How does the performance of the sentiment analysis model vary when using different text preprocessing techniques?

4- What are the most informative features for emotion classification in text, and how does the performance of a sentiment analysis model vary when using different feature selection methods?

Data Preparation

Data Collection

The data set is English Twitter messages with six basic emotions: anger, fear, joy, love, sadness, and surprise. The data includes training, test , and validation text files. The training data set has 16000, and validation and test txt files have 2000 lines of text each. The entire data set is 738 kB. The data set is publicly available and can be found at

PRAVEEN-Kaggle

https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp?select=test.txt

dair-ai/emotion_dataset

https://github.com/dair-ai/emotion_dataset

Data Preparation

For that prep processing I used below techniques:

1. Formatting the data : The data was combination of twit and emotion (sad, angry,etc). So I converted data to a data frame, first column is the twits and the other part is the emotion. Then I created another column in data frame using pyspark StringIndexer method to convert to label to a numeric column

Our emotion labels are : joy, anger, sadness, fear, love, surprize

2. Data Cleaning Process: The process that I follow is removing the tags, none letters, remove words that less than 1 or characters

3. Lemmatize_text: This process expensive to compute since we have to check each word and stem but I wanted to follow this process to see how it effects the accuracy of the models.

4. Removing Stop Words: In this step we want to get rid of the words that does not effect our prediction

5. Tokenizing : We need the number version of the words in the twit

6. Countvectorizer : Counting each words to see how many times it repeated in the twit

7. Idf: This shows us how important a word for that twit

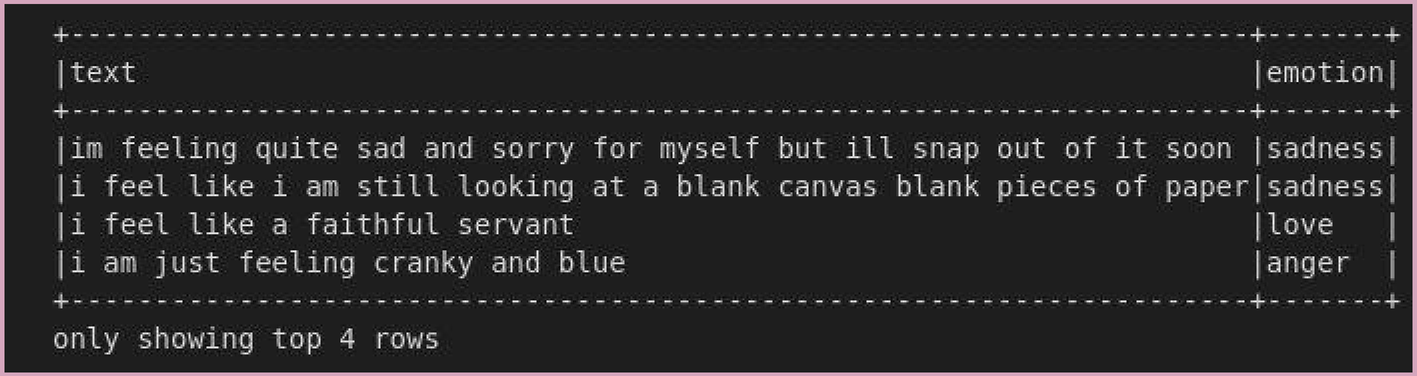
8. ngram: I want to check to see if the context matter, I used ngram=2

9. hashingTF:This is very similar countvectorizer but computationally more effective

10. ChiSqSelector: I want to check to see if less amount of words we still be able to get high accuracy

Data Cleaning

Here is the first formatted of the model



```
+----------------------------------------------------------------------+-------+
|text                                                                  |emotion|
+----------------------------------------------------------------------+-------+
|im feeling quite sad and sorry for myself but ill snap out of it soon |sadness|
|i feel like i am still looking at a blank canvas blank pieces of paper|sadness|
|i feel like a faithful servant                                        |love   |
|i am just feeling cranky and blue                                     |anger  |
+----------------------------------------------------------------------+-------+
only showing top 4 rows
```

Methodology

- I have created different data created four data models

  With cleaned data countVectorizer+iDF

  With cleaned data countVectorizer+iDF+ chi square

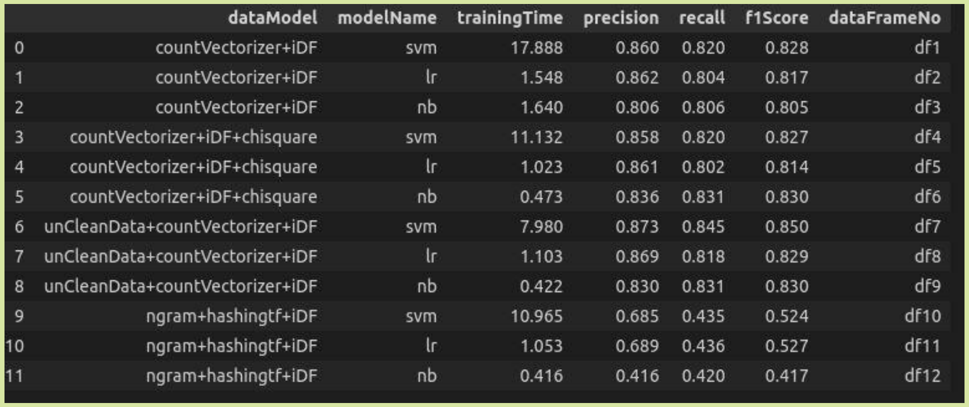
  With uncleaned data + countVectorizer+iDF

  With clean data ngarm+ hashingtf- idf

- I used three machine learning model : SVM, LogisticRegression, Naive Bayes Algorithm

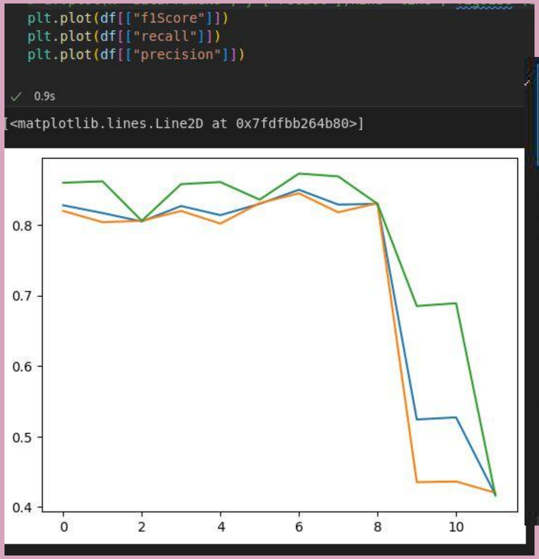
Model Selection

- Since this is a multi class classification I used weighted f1 score, precision and recall to evaluate the model
- I created total 12 models with these 4 different data model
- I saved my results to a df to inspect my result



	dataModel	modelName	trainingTime	precision	recall	f1Score	dataFrameNo
0	countVectorizer+iDF	svm	17.888	0.860	0.820	0.828	df1
1	countVectorizer+iDF	lr	1.548	0.862	0.804	0.817	df2
2	countVectorizer+iDF	nb	1.640	0.806	0.806	0.805	df3
3	countVectorizer+iDF+chisquare	svm	11.132	0.858	0.820	0.827	df4
4	countVectorizer+iDF+chisquare	lr	1.023	0.861	0.802	0.814	df5
5	countVectorizer+iDF+chisquare	nb	0.473	0.836	0.831	0.830	df6
6	unCleanData+countVectorizer+iDF	svm	7.980	0.873	0.845	0.850	df7
7	unCleanData+countVectorizer+iDF	lr	1.103	0.869	0.818	0.829	df8
8	unCleanData+countVectorizer+iDF	nb	0.422	0.830	0.831	0.830	df9
9	ngram+hashingtf+iDF	svm	10.965	0.685	0.435	0.524	df10
10	ngram+hashingtf+iDF	lr	1.053	0.689	0.436	0.527	df11
11	ngram+hashingtf+iDF	nb	0.416	0.416	0.420	0.417	df12

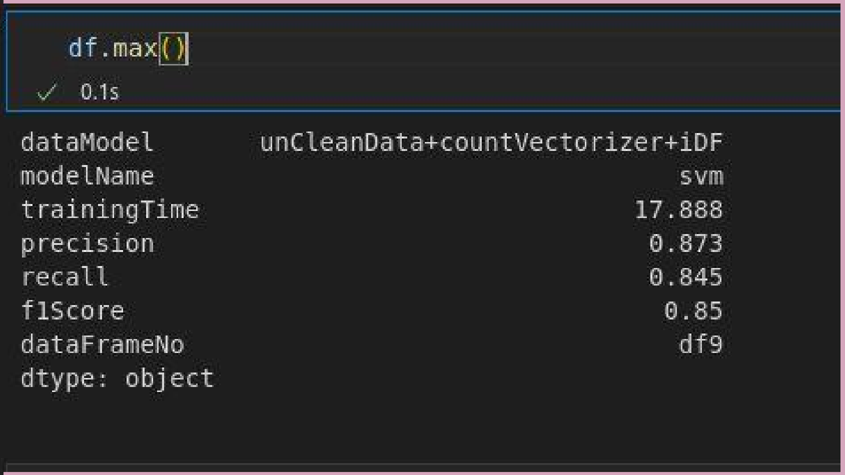
Selecting the best model



```
plt.plot(df[["f1Score"]])
plt.plot(df[["recall"]])
plt.plot(df[["precision"]])
```

✓ 0.9s

```
[<matplotlib.lines.Line2D at 0x7fdfbb264b80>]
```

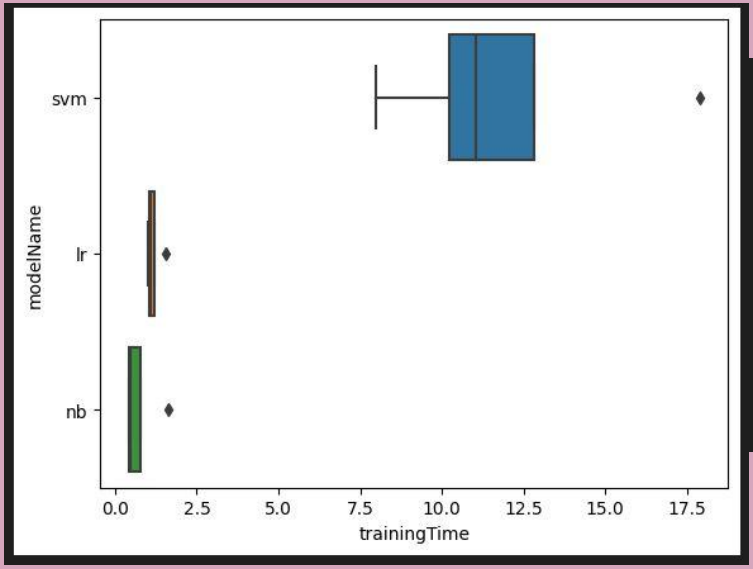


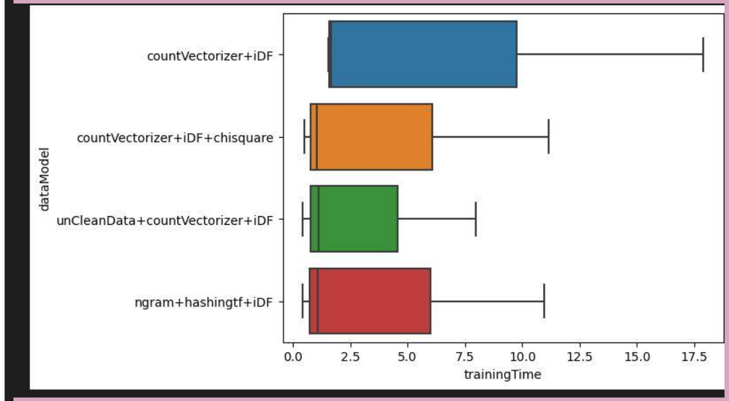
```
df.max()
```

✓ 0.1s

```
dataModel       unCleanData+countVectorizer+iDF
modelName                                   svm
trainingTime                             17.888
precision                                 0.873
recall                                    0.845
f1Score                                    0.85
dataFrameNo                                 df9
dtype: object
```

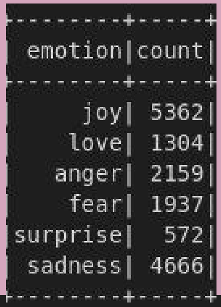
Selecting the best model Training Time



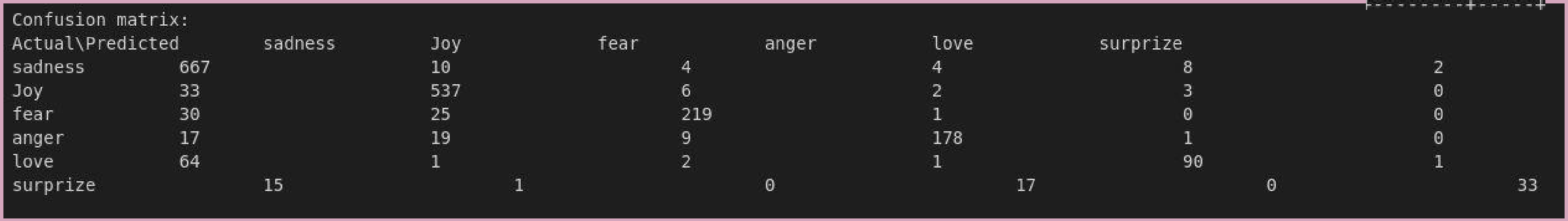


Best Model - Evaluation

We see that our model confused in the sadness a lot but it seems that it is not particularly related to the amount of data

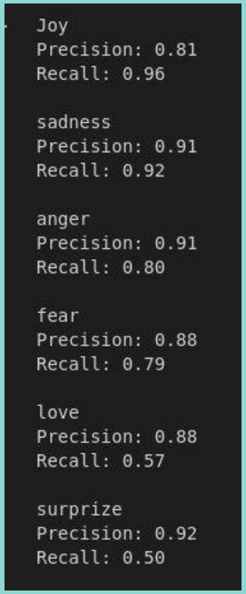


```
+--------+-----+
| emotion|count|
+--------+-----+
|     joy| 5362|
|    love| 1304|
|   anger| 2159|
|    fear| 1937|
|surprise|  572|
| sadness| 4666|
+--------+-----+
```

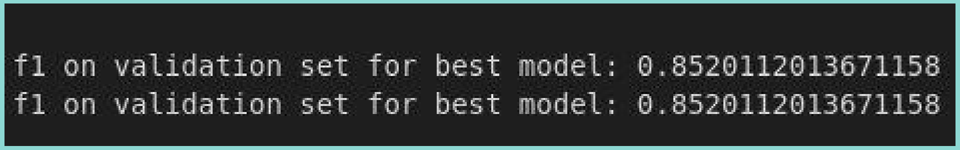


```
Confusion matrix:
Actual\Predicted        sadness         Joy             fear            anger           love            surprize
sadness         667                     10                      4                       4                       8                       2
Joy             33                      537                     6                       2                       3                       0
fear            30                      25                      219                     1                       0                       0
anger           17                      19                      9                       178                     1                       0
love            64                      1                       2                       1                       90                      1
surprize                15                      1                       0                       17                      0                       33
```

Best Model - Evaluation



```
Joy
Precision: 0.81
Recall: 0.96

sadness
Precision: 0.91
Recall: 0.92

anger
Precision: 0.91
Recall: 0.80

fear
Precision: 0.88
Recall: 0.79

love
Precision: 0.88
Recall: 0.57

surprize
Precision: 0.92
Recall: 0.50
```



```
f1 on validation set for best model: 0.8520112013671158
f1 on validation set for best model: 0.8520112013671158
```

Conclusion and Notes

- We achieved a high score accuracy. Therefore it is possible to predict person emotion based on the choose of the words.
- Data cleaning is important part of the nlp but it seems that when it comes to the social media maybe we need emojies and other characters to make sense of the data
- Feature selection decreased model complexity and increased accuracy

Conclusion and Notes

- Surprisingly n-gram did not work on this project.
- Each step in nlp has a very big impact on the result, therefore data model and machine learning model seems inseparable.
- I have tried to implement tensorflow and bert model unfortunately I received installation errors.
- I would like to add neural network and tensorflow library to this project and compare the result with the existing result.