# Sentiment analysis

Nguyen Van Hoang
*Falcuty of Computer Science*
*University of Information Technology*
Ho Chi Minh City, Vietnam
20521346@gm.uit.edu.vn

***Abstract*—Sentiment analysis, a dynamic field of research in Artificial Intelligence, combines Natural Language Processing, Machine Learning, and Psychology to detect polarity (positive, negative, or neutral) in texts. The abundance of opinions in digital formats such as web content, social media, and blogs since 2000 has spurred the development of automated methods for sentiment analysis. It has become essential for individuals and organizations with a public presence to be aware of sentiments expressed about them online. This tutorial provides an overview of sentiment analysis, discussing both knowledge-based and machine learning-based techniques. Sentiment analysis serves as a prevalent text classification tool, examining incoming messages to determine if the underlying sentiment is joy, sadness, anger, fear, or neutral.**

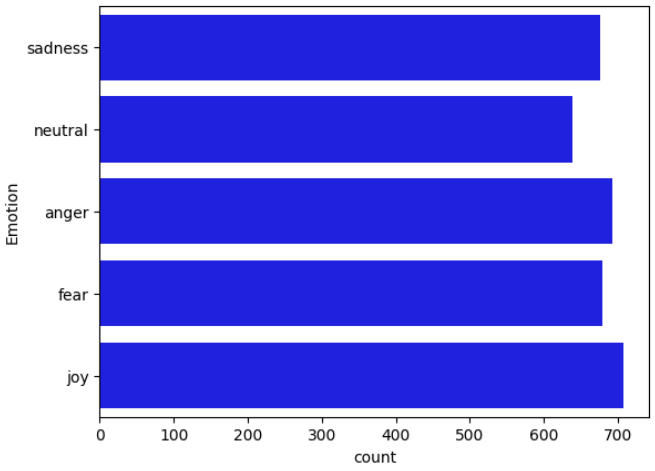
## I. Introduction

The rapid progress of technology has led to the emergence of various online platforms, including blogs, forums, and social networks, enabling users to engage in discussions and share their thoughts on diverse topics. These platforms serve as a space where individuals can voice their grievances about purchased products, engage in debates on current issues, or express their political viewpoints. The utilization of such user-generated information proves valuable in analyzing human behavior and activities across multiple domains. For instance, in the realm of e-commerce, entrepreneurs can gauge user satisfaction levels through feedback, subsequently enhancing product quality. Similarly, governments rely on the analysis of public sentiment to comprehend human behavior and its susceptibility to external opinions. However, sentiment analysis (SA) predominantly relies on data extracted from online social media, where users generate an ever-increasing volume of data. Consequently, these data sources must be regarded within the context of big data, necessitating solutions for efficient data storage, access, processing, and ensuring the reliability of obtained results.

The challenge of automatic sentiment analysis (SA) is gaining traction as a prominent research field. While SA holds significant importance and finds numerous applications, it is undeniably a complex task entangled with challenges associated with natural language processing (NLP). To tackle this, computer science methodologies such as Machine Learning and Deep Learning have been introduced to address the SA problem, leveraging vast amounts of available data.

Identify applicable funding agency here. If none, delete this.

In this report, the group will explore Machine Learning techniques, specifically Support Vector Machine and Decision Tree, implemented by me. I have combined the selected dataset with dailydialog, isear, and emotion-stimulus datasets to create a balanced dataset encompassing five labels: joy, sadness, anger, fear, and neutral. The majority of the texts in this dataset consist of short messages and dialogue utterances, comprising a total of 11,327 sentences. Among these sentences, there are 2,326 sentences in the joy class, 2,317 sentences in the sad class, 2,259 and 2,254 sentences in the anger and neutral classes, respectively, and 2,171 sentences in the fear class. The dataset is divided into two parts: a training set containing 7,934 sentences and a test set containing 3,393 sentences.




## II. Data Preprocessing

The objective of data preprocessing in the context of sentiment analysis (SA) is to eliminate redundant information and noise. To achieve this goal for the given dataset, the following processing steps will be undertaken:

### A. lowercase conversion

For instance, the word ”Artificial” appears at the beginning of a sentence, while the word ”artificial” is used elsewhere, the computer will perceive them as two distinct words. Consequently, this unintended differentiation will inadvertently result in an augmented sentence length.

### B. Remove noise

To eliminate unnecessary elements, eliminate HTML markup, URLs, hashtags, and @names, as well as punctuation, non-ASCII digits, and whitespace.

### C. Remove stopwords

Stopwords refer to a set of words that commonly appear in a sentence but do not contribute significantly to its meaning, including articles, prepositions, and so on. However, the presence of the word "not" in a stopword can alter the meaning of the sentence. For instance, if a sentence begins with "He is not angry," the sentence could belong to either the neutral or angry class. Omitting the word "not" would classify the sentence as angry, significantly influencing the outcome. Therefore, it is necessary to exclude the word "not" from the stopwords list.

### D. Punctuation

Regarding punctuation, several sentences within the dataset include words like "I'm," "He's," "She's," or "I didn't." To ensure the preservation of the word "not" in these cases, it is necessary to expand such sentences through a process called punctuation. The purpose of this expansion is to maintain the intended meaning and context of the sentence.

### E. POS tagged

The objective of utilizing POS tagging is to enable the conversion of words from the past tense to the present tense without the computer perceiving them as distinct words. Nouns, verbs, adjectives, and adverbs are four categories of words that are employed in this process.

## III. Feature Extraction

In the field of natural language processing (NLP), there exist various methods to extract features. In this particular problem, we have opted for tfidf. As demonstrated below, the calculation of tf-idf is outlined.

$$TF(t,d) = \frac{number\ of\ times\ t\ appears\ in\ d}{total\ number\ of\ terms\ in\ d}$$

$$IDF(t) = log\frac{N}{1+df}$$

$$TF-IDF(t,d) = TF(t,d) * IDF(t)$$

TfidfVectorizer() method in sklearn module. There are two parameters used, ngram = 1,2 means unigrams and bigrams, norm = l2: sum of squares of vector elements is 1. The cosine similarity between two vectors is their dot product when l2 norm has been applied and sublineartf = True: used to nomarlize TF value.

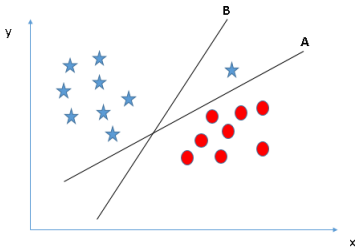
## IV. Support Vector Machine

Support Vector Machine (SVM) is a widely employed supervised learning model in machine learning for tackling multi-class classification tasks. Its primary objective is to discover a hyperplane that can effectively separate the data points. Essentially, the data points belonging to the same category should reside on one side of the hyperplane, while those from different categories should be on the opposite side. Nevertheless, given the existence of multiple possible hyperplanes, the challenge lies in identifying the optimal separating hyperplane based on the following criteria:

*1) The first rule:* : When selecting a hyperplane, it is crucial to choose one that effectively separates the two layers. This means ensuring that no data points from one layer are encompassed within another.

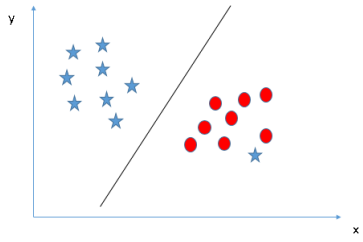
*2) The second rule:* Another important consideration is to calculate the maximum distance, known as the "margin," between the nearest point of a specific layer and the hyperplane. Opting for a hyperplane with a lower margin can significantly increase the risk of misclassifying data as the dataset grows.

*3) The third rule:* Apply the preceding principles to choose the hyper-plane for the following case:

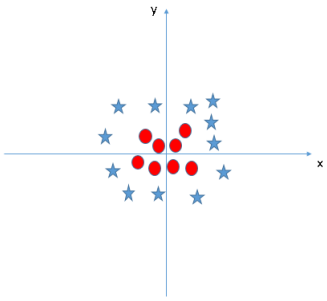



Line B may initially seem preferable due to its larger margin compared to line A. However, this assumption is incorrect as the primary criterion for selection is the first rule, which dictates choosing the hyper-plane to separate the layers. Consequently, line A remains the correct choice.

*4) The fourth rule:* Take into account the image provided below, where it is impossible to partition it into two distinct layers using a single line in order to create a section exclusively containing stars and another section exclusively containing round points.

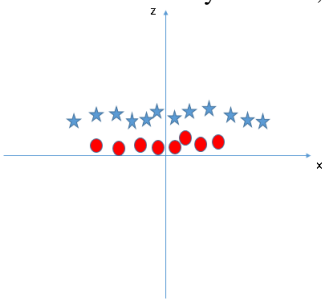



Acknowledging that a late outer star is considered as a star that is further away in this scenario, the SVM possesses a characteristic that enables it to disregard outliers and identify the hyper-plane with the widest margin. Consequently, the SVM exhibits a strong capability to accommodate exceptional cases.



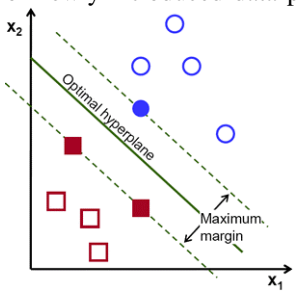

*5) The fifth rule:*

In the aforementioned scenario where no appropriate hyper-plane exists to separate the layers, SVM overcomes this challenge by introducing a solution. It accomplishes this by incorporating an additional feature, specifically z = x2 + y2. By transforming the data along the x and z axes, the problem is effectively resolved, as illustrated below.




### *A. Margin in SVM*

Margin is the distance between the hyperplane to the nearest data points corresponding to the classifiers (figure below). The important thing here is that the SVM method always tries to maximize this margin, thereby obtaining a hyperplane that creates the longest distance from the apples and pears. As a result, SVM can minimize misclassification of newly introduced data points.

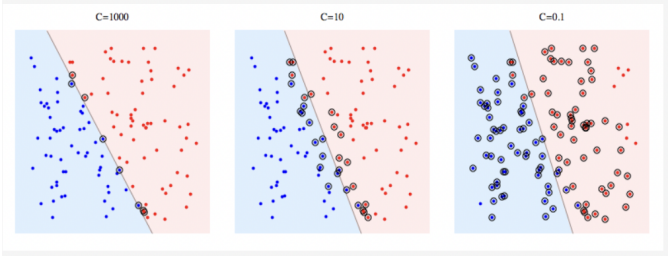



### *B. Hyperparameter*

Some parameters, known as Hyperparameters, cannot be learned directly. Before the actual training begins, they are usually chosen by humans based on some intuition or hit and tried. These parameters demonstrate their significance by improving the model's performance, such as its complexity or learning rate. There are three parameters to consider in this problem: kernel, C, and gamma.

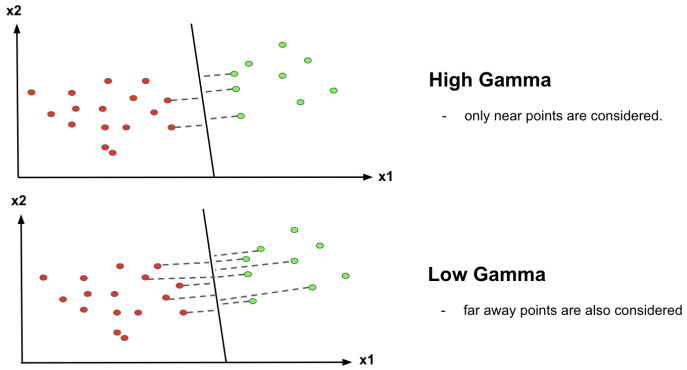
1. Kernel: the main function of the kernel is to take low dimensional input space and transform it into a higher-dimensional space. It is mostly useful in non-linear separation problem. The images below distinguish between the three types of kernels (linear, rbf, polynomial) that the group uses.

2. C (Regularisation): C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimisation how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term.




when C is high it will classify all the data points correctly, also there is a chance to overfit.

3. Gamma: It defines how far influences the calculation of plausible line of separation.




when gamma is higher, nearby points will have high influence; low gamma means far away points also be considered to get the decision boundary.

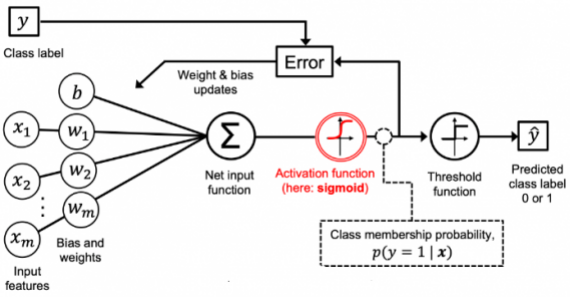
### *C. SVM Hyperparameter Tuning using GridSearchCV*

GridSearch should be used to optimize SVM with three hyperparameters: kernel, C and gamma.
C=0.1, 1, 10, 100, 1000
gamma=1, 0.1, 0.01, 0.001, 0.0001
kernel=linear, poly, rbf
Result when using GridSearchSV:
C = 10, gamma=1, kernel = rbf

## V. Logistic Regression

Logistic regression is a machine learning algorithm used for classification problems. That is, it can be used to predict whether an instance belongs to one class or the other. For example, it could be used to predict whether a person is male or female, based on their height, weight, and other features. It is a supervised learning algorithm that can be used to predict the probability of occurrence of an event. Logistic regression

model learns the relationship between the features and the classes. The logistic regression algorithm is used to map the input data to a probability, unlike linear regression which is used to map the input data to continuous output values. Logistic regression models are used to predict the probability of an event occurring, such as whether or not a customer will purchase a product. The output of the logistic regression model is a probability value between 0 and 1. The output represents the probability that the class of the input data is 1.



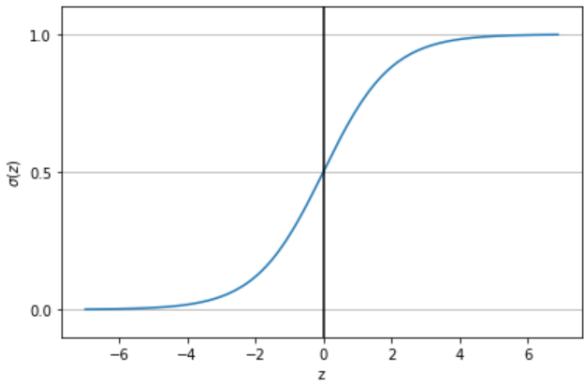

The input data is mapped to a probability using the sigmoid function. The sigmoid function, also called as logistic function, is a mathematical function that maps values (sum of weighted input) from -infinity to +infinity to values between 0 and 1. The sigmoid function that represents the hypothesis is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

The value of z in sigmoid function represents the weighted sum of input values and can be written as the following:

$$z = \theta^T x$$

The following plot is created when the sigmoid function, o(z) is plotted against the net input function output, z. Note that the value of sigmoid function ranges between 0 and 1.




In the above plot, the o(z) approaches 1 when z approaches infinity. Similarly, o(z) approaches 0 when z approaches negative of infinity. Thus, it can be concluded that the value of o(z) ranges from 0 to 1. At z = 0, o(z) takes the value of 0.5. The picture below represents different aspects of a logistic regression model:

Based on the above picture, the following represents some of the key concepts related to logistic regression model:

A set of input features (xi) and related weights (wi) combines together and get added to the bias element (b). This is depicted as new input function in the above diagram. This is same as linear regression function. It is same as "z" shown in equation 1 of the above formula.

The net input is passed to the sigmoid function and the output of the sigmoid function ranges from 0 to 1

The output of o(z) = P(Y = 1— x; w, b). The output represents the probability that a particular data point or example would belong to class 1 given its features x with parameters as weights (w) and the bias (b). Taking the example of IRIS data set, if the goal is to predict whether a flower is IRIS-Versicolor and the value of o(z) = 0.75. This means that the probability that the data points belong to the flower Versicolor is 0.75. When keeping the threshold as 0.5, we can predict that the flower is Versicolor. The predicted probability can be converted to the binary outcome of class 1 or 0 (Versicolor or otherwise in this example) can be represented via the following threshold function.

$$\hat{y} = \begin{cases} 1 & \text{if } \sigma(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The threshold function will become like the following as a function of z (summation of weights and features added with the bias). Refer the sigmoid plot above. For the value of z greater than or equal to 0, one can predict the outcome to be class 1.

$$\hat{y} = \begin{cases} 1 & \text{if } z \geq 0.0 \\ 0 & \text{otherwise} \end{cases}$$

The output of the logistic regression model (sigmoid function output) is always between 0 and 1. If the output is close to 0, it means that the event is less likely to occur. If the output is close to 1, it means that the event is more likely to happen. For example, if the value of logistic regression model (represented using sigmoid function) is 0.8, it represents that the probability that the event will occur is 0.8 given a particular set of parameters learned using cost function optimization.

Based on the threshold function, the class label can said to be 1. For any new value X, the output of the above function will be used for making the prediction.

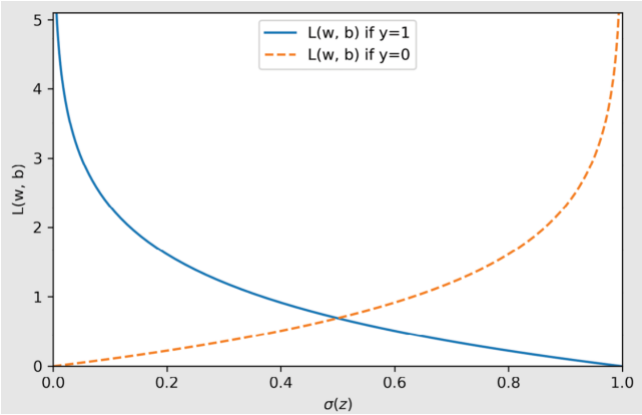
The parameters in logistic regression is learned using the maximum likelihood estimation. The cost function for logistic regression is defined as:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

In above cost function, h represents the output of sigmoid function shown earlier, y represents the class/label of the training data, x represents the training data. Note that for binary classification problems, the first term will be zero for class labeled as as 0 and the second term will be zero for class labeled as 1. The equation below represents this aspect:

$$\text{cost}\,(h_{\boldsymbol{\theta}}(\boldsymbol{x}),\ y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\boldsymbol{x})) & \text{if } y = 0 \end{cases}$$

When the loss function is plotted against hypothesis function (sigmoid), the following plot occurs for y = 0 and y = 1.




In order to fit the parameters, the objective function J(0) would need to be minimized. Gradient descent algorithm (stochastic gradient descent – SGD) can be used for optimizing the objective or cost function. This is how the equation looks like for updating the parameters when executing gradient descent algorithm. Ensuring that gradient descent is running correctly, the value of J(0) is calculated for 0 and checked that it is decreasing on every iteration.

while not converged {

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \text{ for } j = 0, 1, \ldots, n$$

}

Besides stochastic gradient descent algorithm, it is recommended to use advanced algorithms such as some of the following: Conjugate gradient, BFGS, L-BFGS etc. When using scikit-learn for training logistic regression models, these algorithms can be used by mentioning solver parameter such as newton-cg, lbfgs, liblinear, saga, sag, etc.

Logistic regression is similar to linear regression, but the dependent variable in logistic regression is always categorical, while the dependent variable in linear regression is always continuous.

**Using Logistic Regression for Sentiment Analysis:** We are finally ready to train our algorithm. We need to choose the best hyperparameters like the learning rate or regularization strength. We also would like to know if our algorithm performs better or not... To take these methodically, we can use a Grid Search. GridSearchCV is a method of training an algorithm with different variations of parameters to later select the best combination.

Therefore, we should understand clearly about each hyperparameter which is used in Logistic Regression.

### *A. Logistic Regression Hyperparameters*

sklearn.linear-model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit-intercept=True, intercept-scaling=1, class-weight=None, random-state=None, solver='lbfgs', max-iter=100, multi-class='auto', verbose=0, warm-start=False, n-jobs=None, l1-ratio=None)

**Slover** Solver is the algorithm to use in the optimization problem. The choices are 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga', default='lbfgs'. [1]

1) **lbfgs**: relatively performs well compared to other methods and it saves a lot of memory, however, sometimes it may have issues with convergence.
2) **sag**: faster than other solvers for large datasets, when both the number of samples and the number of features are large.
3) **saga**: the solver of choice for sparse multinomial logistic regression and it's also suitable for very large datasets.
4) **newton-cg**: computationally expensive because of the Hessian Matrix.liblinear recommended when you have a high dimension dataset - solving large-scale classification problems.

**Penalty (or regularization)**: intends to reduce model generalization error, and is meant to disincentivize and regulate overfitting. Technique discourages learning a more complex model, so as to avoid the risk of overfitting. The choices are: 'l1', 'l2', 'elasticnet', 'none', default='l2'. However, some penalties may not work with some solvers, see more on sklearn user's guide.

**C (or regularization strength)** must be a positive float. Regularization strength works with the penalty to regulate overfitting. Smaller values specify stronger regularization and high value tells the model to give high weight to the training data.

Logistic regression offers other parameters like: class-weight, dualbool (for sparse datasets when n-samples ¿ n-features), max-iter (may improve convergence with higher iterations), and others. However, these provide less impact.

### B. Tunning

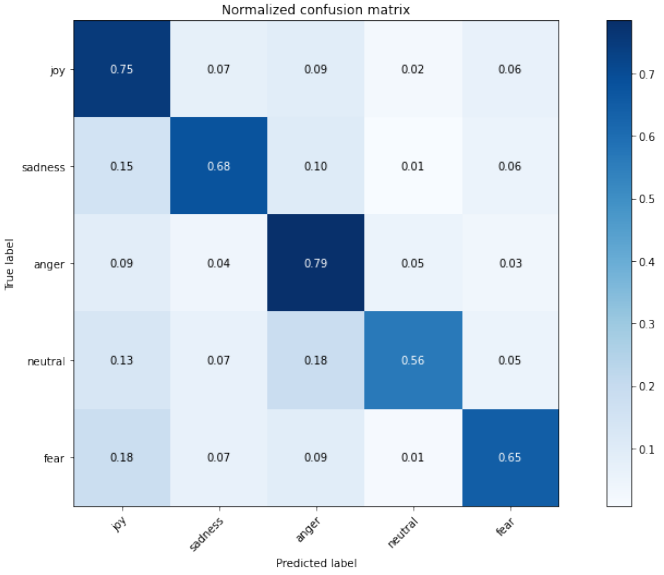
After understanding the hyperparameter which is used in Logistic Regression, we are going to tunning them

```
paramgrid = {'max_iter': [10, 200, 300],
'multi_class': ['auto', 'ovr', 'multinomial'],
'C': [0.01,0.1,1,10,100],
'solver': ['newton-cg','lbfgs','liblinear']}
```

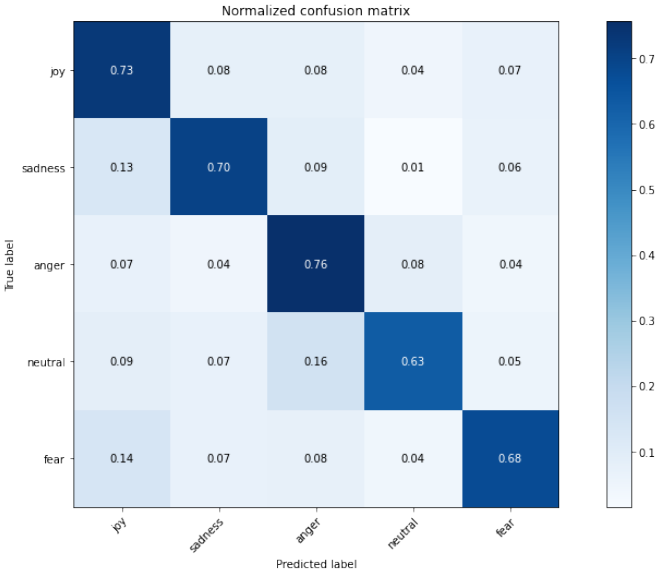
## VI. Experiment

### A. Support Vector Machine

Confusion matrix without Tuning

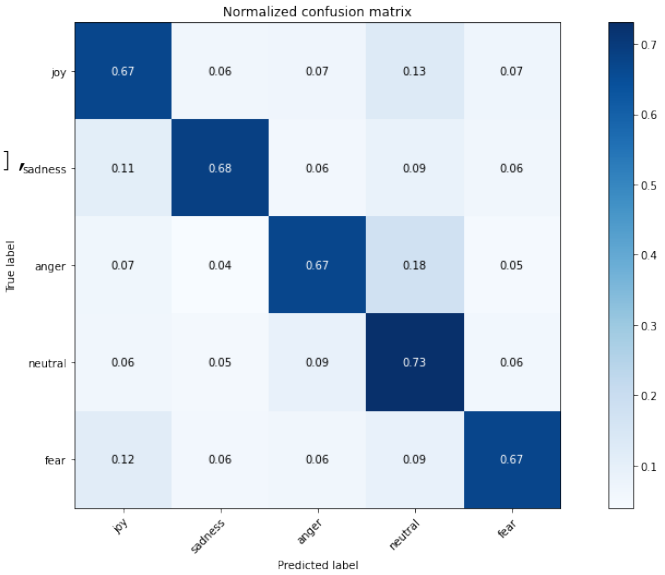



Confusion matrix with Tuning

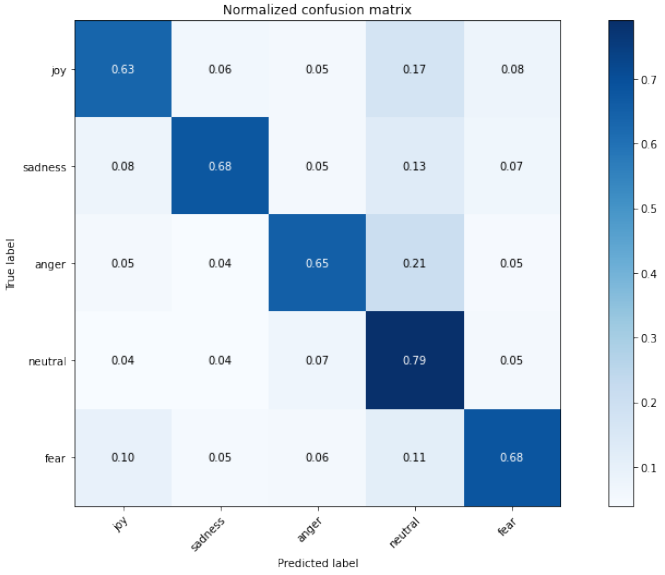



### B. Logistic Regression

Confusion matrix without Tuning




Confusion matrix with Tuning




### C. Table result

| | SVM | SVM with tuning | Logistic | Logistic with tuning |
|---|---|---|---|---|
| F1-score | 68.9 | 70 | 68.5 | 68.6 |
| accuracy | 68.9 | 70 | 69.5 | 68.6 |

## VII. Conclusions

When utilizing the default function, the overall outcomes are fairly satisfactory, nearing 0.7. Nevertheless, employing the GridSearch technique to seek optimized accuracy parameters did not yield significant improvements in the results of both machine learning methods. It is plausible to draw conclusions; however, the utilization of gridsearch does not consistently produce fruitful outcomes. This may

be attributed to the group's parameter value selection or the inadequacy of the number of parameters, resulting in insignificant result enhancements.

### A. Support Vector Machine

In the absence of any parameters, the SVM approach produces a result of 68.85. However, employing the GridSearch method increases the result to 70.09 by utilizing C = 10, gamma = 1, and kernel = rbf parameters. This demonstrates an elevation in the sadness class from 0.68 to 0.7, an increase in the neutral class from 0.56 to 0.63, and a growth in the fear class from 0.65 to 0.68.

### B. Logistic Regression

As apparent from the aforementioned table of results, there is a minor improvement in our metrics: accuracy and f1-score. When compared to hyperparameter tuning, the default setting exhibits an approximately 0.9 to 1 percent lower performance. This suggests that our tuned hyperparameters are not exceptionally efficient. There could be various reasons for this, but the most plausible explanation is that the customized parameters selected for tuning do not surpass the default parameters significantly.

## REFERENCES


[1] Clare Liu, SVM Hyperparameter Tuning using GridSearchCV, 2020
[2] Ajitesh Kumar, Logistic Regression Concepts, Python Example
[3] scikit-learn developers (BSD License), SVC, 2007 - 2022.
[4] Huynh Chi Trung, Introduction into Support Vector Machine, 2020
[5] scikit-learn developers (BSD License), Logistic Regression, 2007 - 2022.