Sentiment analysis based on Machine Learning methods

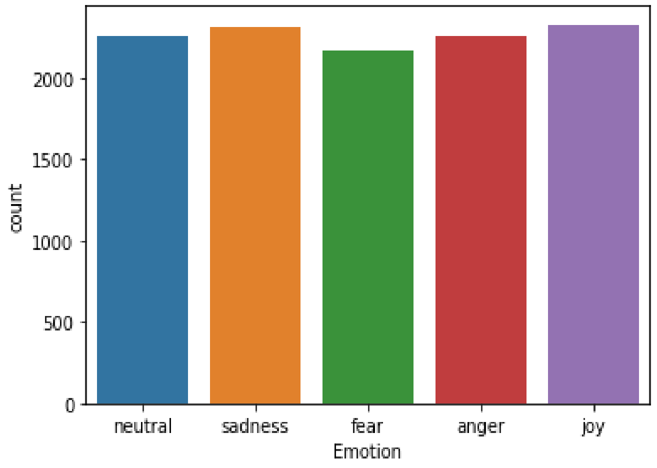
Huynh Anh Kiet
Falcuty of Computer Science
University of Information Technology
Ho Chi Minh City, Vietnam
19521724@gm.uit.edu.vn

Nguyen Thi Nhu Van
Falcuty of Computer Science
University of Information Technology
Ho Chi Minh City, Vietnam
20520855@gm.uit.edu.vn

***Abstract*—Summary form only given. Sentiment analysis is an exciting new field of research in Artificial Intelligence combining Natural Language Processing, Machine Learning and Psychology. Since 2000, due to the proliferation of huge amounts of opinions in electronic form on the web, on social networks and on blogs, automatic means of polarity (positive, negative and neutral) detection in texts flourished in leaps and bounds. Individual and organizations with public interface can no longer afford to be oblivious of sentiments expressed about them in electronic form. In the present tutorial, we will first discuss the foundations of sentiment analysis, covering knowledge based and machine learning based techniques. Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is joy, sadness, anger, fear or neutral.**

I. Introduction

The advancement of technology has resulted in the creation of blogs, forums, and online social networks that allow users to discuss and share their thoughts on any topic. They may, for instance, complain about a product they purchased, debate current issues, or express political opinions. Exploiting such user information is beneficial in analyzing human activity and behavior in a variety of areas. In the field of e-commerce, the founder understands the level of user satisfaction with any product via feedback and plans to improve the item's quality. In addition, analyzing public opinion is critical for governments because it explains human activity and behavior, as well as how they are influenced by the opinions of others. Nevertheless, the sources of data for sentiment analysis (SA) are online social media, whose users generate a growing amount of data. As a result, these types of data sources must be considered as part of the big data approach, as additional issues must be addressed in order to achieve efficient data storage, access, and processing, as well as to ensure the reliability of the obtained results.
The problem of automatic sentiment analysis (SA) is becoming more popular as a research topic. Although SA is an important area with many applications, it is clearly not an easy task with many challenges related to natural language processing (NLP). As a result, methods from the field of computer science, such as Machine Learning or Deep Learning, are introduced to serve the SA problem with a very large amount of data.
The group will learn about Machine Learning methods such as Support Vector Machine and Decision Tree, which is implemented by Huynh Anh Kiet and Nguyen Thi Nhu Van, in this report.
The dataset selected by the group was combined with dailydialog, isear, and emotion-stimulus to produce a balanced dataset with five labels: joy, sadness, anger, fear, and neutral. The majority of the texts are made up of short messages and dialog utterances. The dataset contains 11327 sentences, with 2326 sentences in the joy class, 2317 sentences in the sad class, 2259 and 2254 sentences in the anger and neutral classes, and 2171 sentences in the fear class. The dataset is divided into two parts: a train set of 7934 sentences and a test set of 3393 sentences.



II. Preprocessing data

The goal of preprocessing data is to remove unnecessary information and noise out from the problem (SA). For this dataset, there will be the following processing steps:

A. *Uppercase to lowercase*

For example, if there is ”There” at the beginning of a sentence and the word ”there” is non-zero at the beginning of the sentence, the computer will interpret these as two different words, inadvertently increasing the sentence length.

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

B. *Remove noise*

Remove html markup, urls, hashtags and @names, punctuation and non-ascii digits and whitespace.

C. *Remove stopwords*

Stopword simply comprehends a collection of words that appear in a sentence but convey the meaning of that sentence, such as articles, prepositions, ect. However, in a stopword with the word ”not”, this word can change the meaning of the sentence. For example, if the sentence begins with ”He is not angry”, the sentence could be neutral or angry class. If the word ”not” is omitted, the sentence belongs to the angry class, which has an impact on the outcome. As a result, we will remove the word ”not” from the list.

D. *Punctuation*

Many sentences in the dataset contain the words: I’m, He’s, She’s, or I didn’t. As a result, in order to retain the word ”not,” such sentences must be expanded, a process known as punctuation.

E. *POS tagged*

The goal of using POS tagged is to be able to move words from the past tense to the present tense without the computer recognizing that a word in two different tenses is two different words. Noun, Verb, Adjective, Adverb are four types of words used.

III. Feature Extraction

In NLP, there are numerous ways to obtain features; for this problem, we chose tfidf. as shown below, how to calculate tfidf:

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

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

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

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. The below image after using tfidf as an example.

```
print(X_train_vect[0])
```

```
  (0, 63029)	0.21673999437414293
  (0, 58933)	0.45381381735832493
  (0, 58932)	0.43472982533905613
  (0, 57643)	0.3885652176763226
  (0, 57484)	0.22399539141018943
  (0, 41506)	0.45381381735832493
  (0, 41499)	0.3885652176763226
```

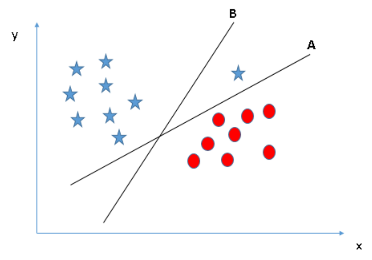
IV. Support Vector Machine

Support Vector Machine (SVM) is a supervised learning model that is commonly used in multi-class classification problems in machine learning. The method’s goal is to find a hyperplane that divides the data points correctly. In other words, the data points from the same layer are on the same side of the hyperplane as the data points from the other layers. However, there are numerous such hyperplanes; the problem is determining the best dividing hyperplane based on the following criteria:

1) The first rule: The first rule of selecting a hyperlane is to select a hyperplane that best separates the two layers. In other words, no data points from one layer are contained within another.

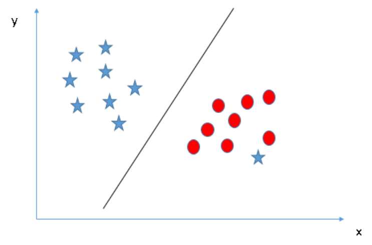
2) The second rule: The second rule is to calculate the maximum distance between the nearest point of a given layer and the hyperplane. The term ”margin” refers to this distance. If you choose the wrong hyperlane with a lower margin, there is a high risk of misclassifying the data later when the data increases.

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

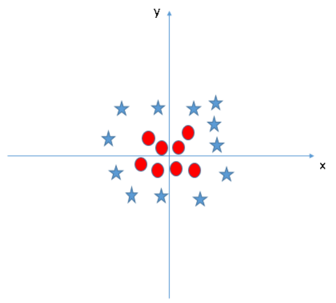


There may be line B because it has a greater margin than line A, but that will not be true because the first rule will be rule number 1, we must choose the hyper-plane to separate the layers. As a result, line A is the correct answer.

4) The fourth rule: Consider the image below, which cannot be divided into two separate layers with a single line to create a section with only stars and another with only round points.

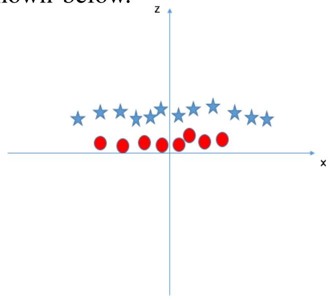


Accepting that a late outer star is treated as a more outer star in this case, the SVM has a feature that allows it to ignore outliers and find the hyper-plane with the greatest boundary. As a result, SVM has a high capacity for accepting exceptions.



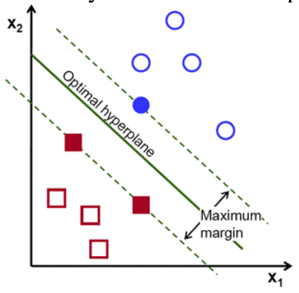
5) *The fifth rule:*

Because there is no relative hyper-plane to divide the layers in the case above, how does SVM split the data into two separate layers? SVM can solve this problem. Simply put, it will be solved by adding a feature, in this case z = x2+ y2. The data will now be transformed along the x and z axes as shown below.



A. *Margin in SVM*

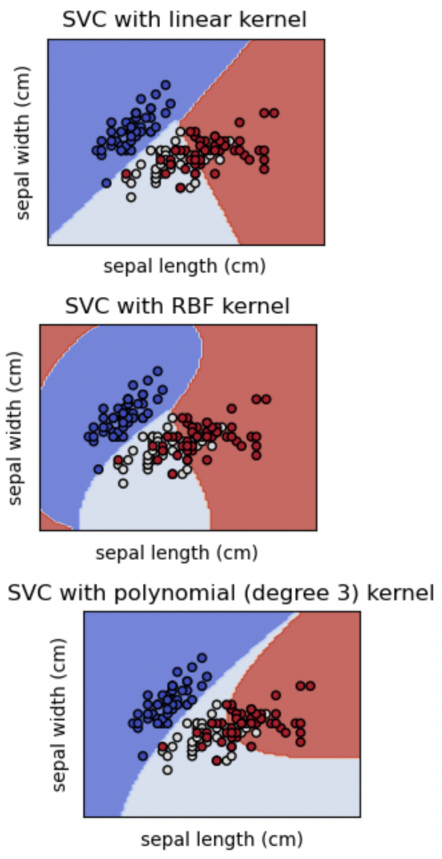
Margin is the distance between the hyperplane to the nearest data points corresponding to the classifiers (figure below). The important thing here is that the SVM method always tries to maximize this margin, thereby obtaining a hyperplane that creates the longest distance from the apples and pears. As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



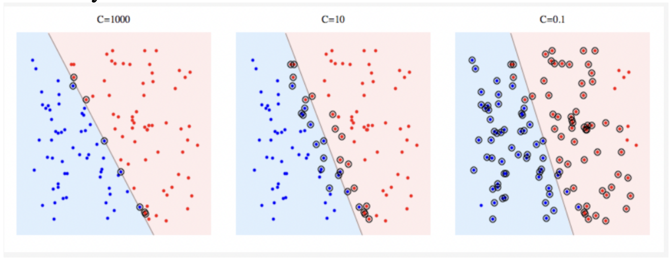
B. *Hyperparameter*

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 gramma.

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.

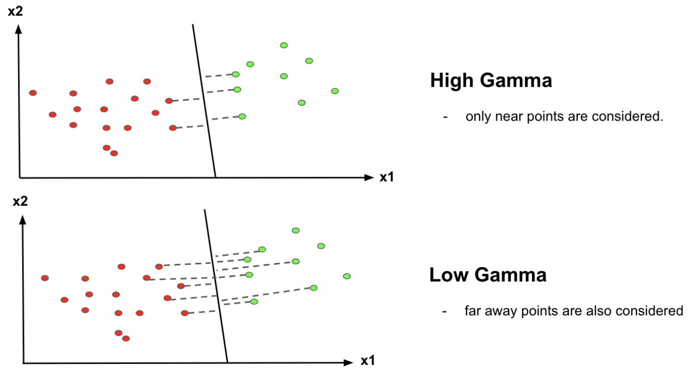


2. C (Regularisation): C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimisation how much error is bearable. This is how you can control the trade-off between decision boundary and misclassification term.



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.


x2
x1
High Gamma
- only near points are considered.
x2
x1
Low Gamma
- far away points are also considered

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

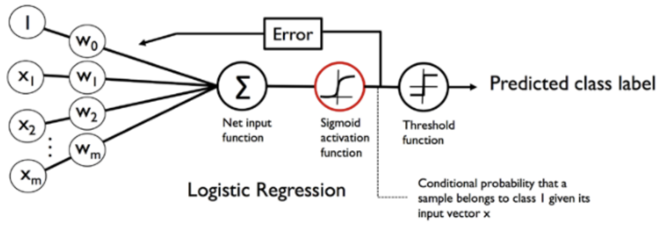
C. SVM Hyperparameter Tuning using GridSearchCV

GridSearch should be used to optimize SVM with three hyperparameters: kernel, C and gramma.
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 classification model that is very easy to implement and performs very well on linearly separable classes. It is one of the most widely used algorithms for classification in industry too, which makes it attractive to play with.

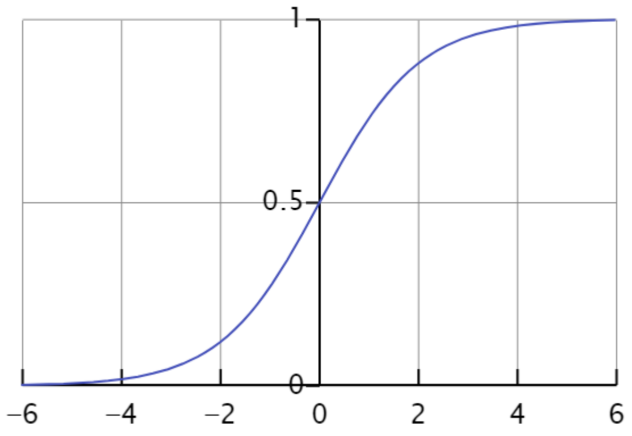
Very simplistically explained, Logistic Regression works as follows: [2]



First we will define the input for our algorithm. The input will be each sample in whatever dataset we are working with. Each sample will consist of several features. For example, if we're working with housing price prediction, the features for each sample could be the size of the house, number of rooms, etc. We'll call the input vector X.

For the algorithm to learn, we need to define variables that we can adjust accordingly to what we want to predict. We will create a vector of weights (W) that the model will adjust in order to predict more accurately. The process of adjusting those weights is what we call learning.

For every input sample, we will perform a dot product of the features by the weights XW. This product is sometimes referred to as net input. This will give us a real number. Since in this particular problem we want to classify (positive/negative), we need to squash this number in the range [0, 1]. This will give us the probability of a positive event. A function that does precisely that is called sigmoid. The sigmoid function looks like this:



What sigmoid is doing is basically transforming big inputs into a value close to 1, and small inputs into a value close to 0. This is exactly what we want.

We will do this for every sample in our training set and compute the errors. To calculate the error we only need to compare our prediction with the true label for each sample. We will sum the square errors of all the samples to get a global prediction error. This will be our cost function.

A cost function is then something we want to minimize. Gradient descent is a method for finding the minimum of a function of multiple variables, such as the one we're dealing with here.

The training process:
In order for logistic regression to learn, we need to repeat the process described before several times. Each one of these times is called an epoch. The number of epochs to run depends on the problem and the training data. It is... yes, another tunable parameter of the algorithm.

The set of all tunable parameters is called hyperparameters of the model.

Like with the learning rate, we need to be careful when choosing the number of epochs: If we train too many epochs, we risk overfitting. This means that our model will "memorize" the training data and will generalize badly when presented new data.

If we train too little, it will fail to find any pattern and the prediction accuracy will be very low. This is known as underfitting.

There are techniques that help prevent overfitting. These regularization techniques are out of the scope of this tutorial, but... guess! It's also something to tune and experiment with.

This is why when training a model you need to set aside a test dataset in order to know the accuracy of your algorithm in unknown data. The test dataset will never be used during training.

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*

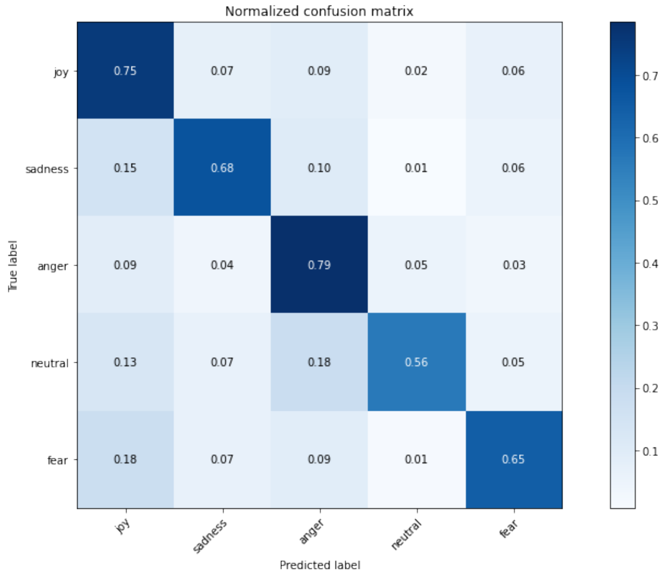
After understanding the hyperparameter which is used in Logistic Regression, we are going to tunning them

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

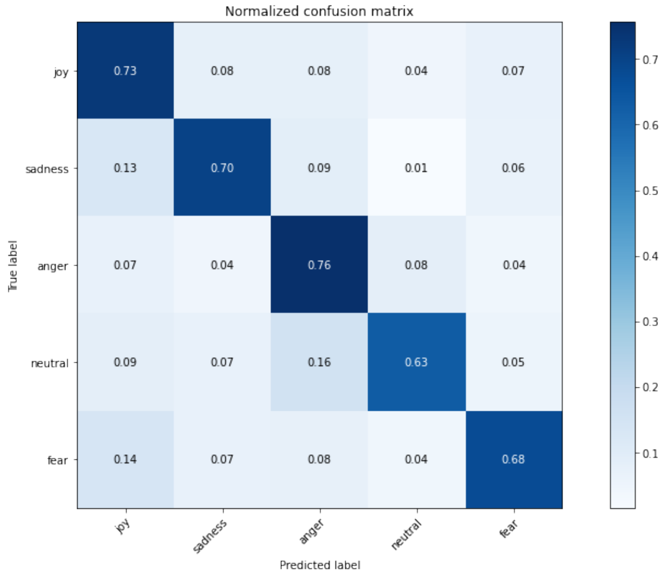
VI. Experiment

A. *Support Vector Machine*

Confusion matrix without Tuning

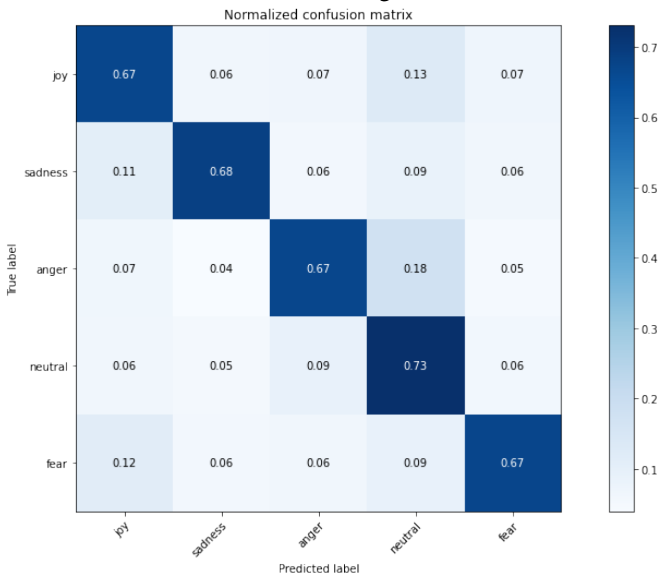


Confusion matrix with Tuning

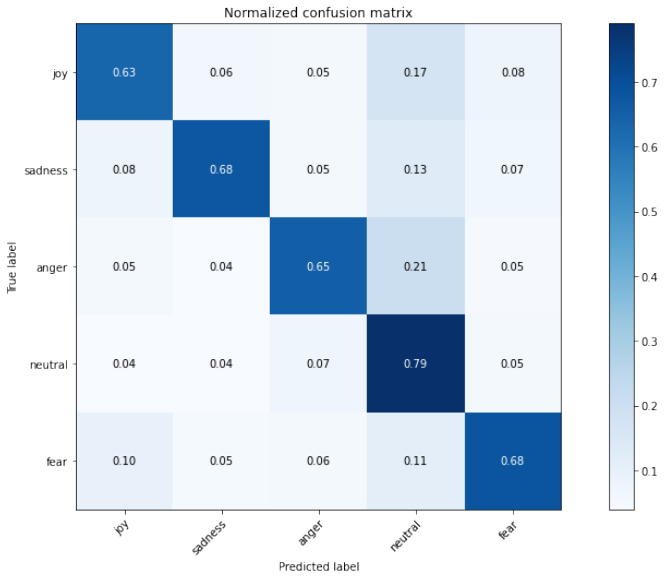


B. *Logistic Regression*

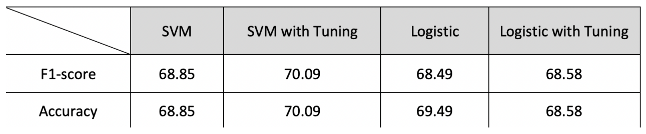
Confusion matrix without Tuning



Confusion matrix with Tuning



C. *Table result*



	SVM	SVM with Tuning	Logistic	Logistic with Tuning
F1-score	68.85	70.09	68.49	68.58
Accuracy	68.85	70.09	69.49	68.58

VII. Conclusions

When using the default function, the overall results are quite good, approaching 0.7. However, when using the GridSearch method to find parameters to optimize accuracy, the results in both machine learning methods did not improve significantly. It is possible to draw conclusions; however, using gridsearch does not always yield results; perhaps the group uses parameter values, or the number of parameters is insufficient, so the results do not increase significantly.

A. *Support Vector Machine*

The result of using SVM without any parameters is 68.85. However, when the GridSearch method is used, the result increases to 70.09 with C = 10, gamma=1, kernel = rbf parameters. It can be seen that the class of sadness rises from 0.68 to 0.7, the class of neutral increases from 0.56 to 0.63, and the fear class grows from 0.65 to 0.68.

B. *Logistic Regression*

As can be seen from the table result above, there is a small increasing in our metrics: accuracy and f1-score. With default, it is approximately 0,9 to 1 percent less comparing to hyperparameter tunning, which means our tunned hyperparameters are not really efficient. There are many reason for this, but the most feasible is the customised parameters we have chosen for tuning are not better than the default parameters much.

References

[1] scikit-learn developers (BSD License), Logistic Regression, 2007 - 2022.
[2] Sebastian Raschka, ”What is the relation between Logistic Regression and Neural Networks and when to use which?”
[3] scikit-learn developers (BSD License), SVC, 2007 - 2022.
[4] Huynh Chi Trung, Introduction into Support Vector Machine, 2020
[5] Clare Liu, SVM Hyperparameter Tuning using GridSearchCV, 2020

00