**GitHub Link:**

https://github.com/dheerajreddykomandla/NLP_TextEmotionDetection

# CSCE 5290
# Natural Language Processing
# Text Emotion Detection
# Increment-1

## 1. Project Title and Members:

- **Project Title:** **Text Emotion Detection**
- **Team Number:** Team 10

**Team Members:**

Dheeraj Reddy Komandla (11526265)
Prem sai Vuppula (11528103)
Rahul Reddy Geereddy (11511846)
Sravan Boinapalli (11553755)

## 2.Goals and Objectives:

- **Motivation:** We opted to work on the Text categorization problem as a team in this project. In which we have decided to work on the Text Classification challenge. Sentiment analysis has recently been used in a wide range of applications. Instead of working on the binary classification task, we as a team decided to challenge ourselves. The key reason for using Text Emotion Detection is that the class labels are connected to the sentiment analysis problem and more resemble the intent classification problem used in conversation bots. According to our investigation, and for the reasons stated above, we decided on the Text Emotion Detection challenge.
- **Significance:** As previously stated, the primary value of this model is that it has a broad range of real-world applications. Consider a chatbot conversation in which emotions play a key part in communication between people from different cultures or areas. Assume we have divided emotions into four categories. They are: joy, anger, sadness, fear.

  We can see the difference between single word and multiple word assertions in the example below.

  Single Words

  - Great
  - Nope
  - Recommend

  Multiple Words Statements

  - This is great!
  - Nope, won't buy again!
  - Definitely recommend this product!

The emotion represented in both statements is the same, but the amount of words varies. As a result, it is critical for a chatbot model to accurately classify the emotion based on which future communication must take place.

Let's try to understand the significance of the model with one more example. If you consider the below statements.

- This is not a friendly product.
- I hate this product; not useful at all.
- I want to return this product, maybe order a replacement.

If you observe the above 3 statements, all are having the sentiment negative. But the meaning, emotion, and the customer requirement for the 3 statements are different.

## Objectives:

Our team has various goals that must be met in order to properly construct an emotion detection model. The following objectives depict the project's workflow.

**1. Data Selection:** As we did some research, we have shortlisted some of the datasets available in the Kaggle. We would like to spend some more time to pick the right dataset with good number of emotion samples.

**2. Text Cleaning:** In any of the NLP tasks, it is important to clean the text dataset. Because the raw dataset may contain lot of missing values, duplicate values etc. We need to handle all the cases. If the text is not cleaned, then the model may give bad results, or it fails.

**3. Text Preprocessing:** It is also an important phase in all NLP related tasks. In this phase, we generally remove the white spaces, remove unnecessary special characters, stop words removal etc.

**4. Text Vectorization:** This phase is also known as text featurization. In this phase we generally convert the text into vectors. There are some popular Vectorization techniques available such as Bag of Words, Tf-IDF, Word2Vec etc. By using these techniques, we will be able to convert the text into n dimensional vector.

**5. Modeling:** Once we converted the text into vector, then we are good to go for modeling. Here we want to try this various machine learning algorithms. Of all the models that are trained, which ever gives the best accuracy, we will choose that model as best model.

**6. Visualization & Evaluation of Results:** In this phase, we basically compare the results of each model and plot the results to understand the performance of the model. Since it is a classification model, we would like to compare the

performance with accuracy. If the dataset is imbalanced, then we will also look into f1-score, AUC score.

## Features:

As discussed earlier we would like to spend some more time on picking the dataset, but in most of the text classification datasets, it contains 2 features.

**1. Text:** This feature mainly contains the raw text. This text represents the either review or statement for which we need to predict the emotion of the statement.

**2. Label:** This feature represents the class label of the given sentence. In most of the datasets we basically have 4 classes. They are:

1) Fear
2) Joy
3) Anger
4) Sadness

There are some import machine learning libraries are required to solve this problem. They are listed below:

1) Pandas
2) SKlearn
3) Matplotlib
4) XGboost

## 4. Related Work (Background):

According to our study, the most often utilized strategy for text categorization problems includes the steps listed below.

1) Text Cleaning

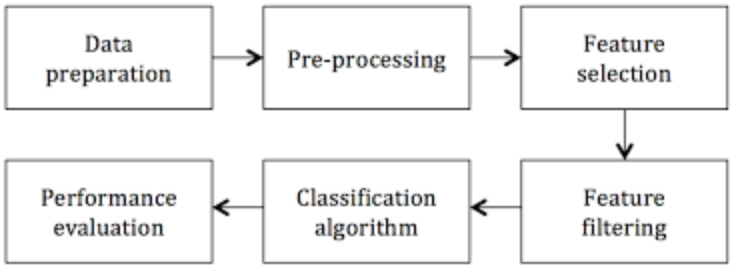
2) Text Preprocessing

3) Text Vectorization

4) Train Test Split

5) Modeling

6) Visualization of Results




Some of the research that we have done are listed below.

https://stackabuse.com/text-classification-with-python-and-scikit-learn/

https://www.datacamp.com/tutorial/naive-bayes-scikit-learn

https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python-and-nltk-c52b92a7c73a

**4. Data Set:** The dataset we are currently using in this project was obtained from the kaggle repository. Kaggle is a repository of a huge number of datasets that are freely available for use in practicing data science challenges. The link to the dataset's kaggle repository is provided below.

https://www.kaggle.com/datasets/anjaneyatripathi/emotion-classification-nlp

We have two columns in the dataset to work with as we solve the Multi-Class text classification problem.

1. Text Column: This column contains the raw text. Our model should be able to take this text as input and predict the mood of the text.

2. Label: This column contains the class label for the relevant input text. The Label column has four different types of feelings. They are Fear, Anger, Joy, and Sadness.

**5. Detail Design of Features:** When it comes to machine learning or deep learning, the models cannot directly interpret text data. As a result, we must convert the text into numerical features. There are numerous text enhancement techniques available. Popular featurization techniques include:

1. Bag of words

2. TF-IDF

3. Word2Vec

In general, every featurization algorithm will turn the text into an N-dimensional array. We are now using the TF-IDF featurization technique in this project.

**TF-IDF Featurization:** TF-IDF stands for term frequency and inverse document frequency. The TF-IDF technique is nothing more than assigning a weight to each unique word in the dataset in relation to the given input text. The formula for calculating the weight of a word in relation to a specific text row is shown in the picture below.

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

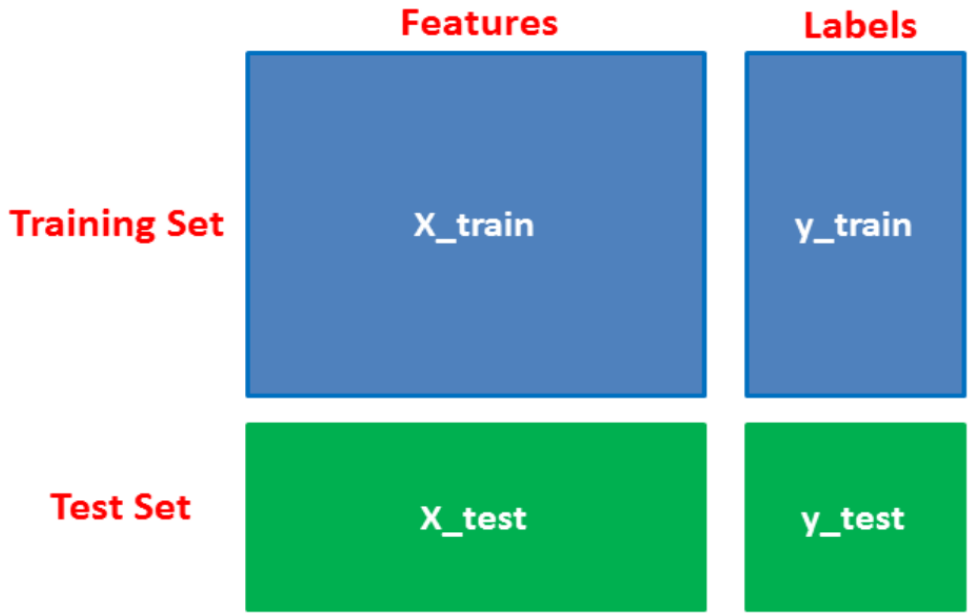
**TF-IDF**
Term x within document y

$tf_{x,y}$ = frequency of x in y
$df_x$ = number of documents containing x
N = total number of documents

The example below shows how text can be transformed into an N-dimensional vector.

| Words/ Documents | going | to | today | i | am | if | is | rain |
|---|---|---|---|---|---|---|---|---|
| Document 1 | 0 | 0.07 | 0.07 | 0 | 0 | 0.17 | 0.17 | 0.17 |
| Document 2 | 0 | 0 | 0.07 | 0.07 | 0.07 | 0 | 0 | 0 |
| Document 3 | 0 | 0.05 | 0 | 0.05 | 0.05 | 0 | 0 | 0 |

**6. Analysis:** So far in this part, we've transformed the given input into an N-dimensional vector representation. We must now perform the Train Test Split process. It is a critical practice in machine learning and deep learning. It facilitates training the model on the training dataset and testing the model on the test dataset. It also aids in determining whether the model is over-fitting. The diagram below depicts the breakdown of the train test split.

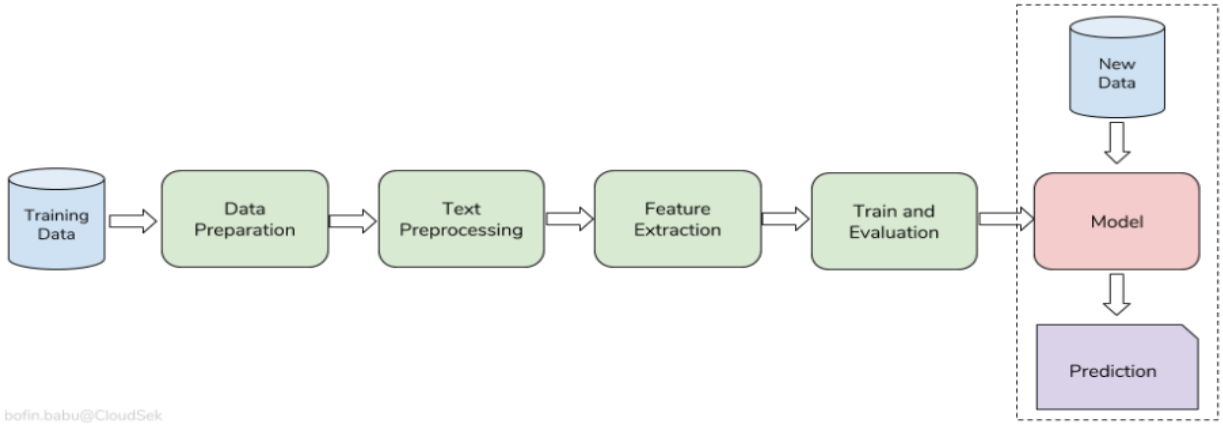


We have a different dataset for testing in our dataset. As a result, we use the split-ted dataset for validation and a second test file to assess the final correctness.

When it comes to modeling, we did some study and discovered that the Naive Bayes model works well for text-related difficulties. As a result, we attempted the Naive Bayes classifier for the emotion categorization task.

Naive Bayes is a probabilistic approach to classification issue solving. It is based on the notions of the Bayes theorem. We used multinomial Naive Bayes in this project, which uses a multinomial distribution for each of the features. To understand more about the Naive Bayes algorithm. Find the link [https://www.javatpoint.com/machine-learning-naive-bayes-classifier](https://www.javatpoint.com/machine-learning-naive-bayes-classifier).

**The Model Workflow:**




---

## 7.Implementation:

The project flow has already been explained. Each phase's implementation is shown in this section. We have included screenshots for each phase of this project.

### 1. Data Preparation:

**Reading the files**

```
In [2]: train_df = pd.read_csv('emotion-labels-train.csv')
        test_df = pd.read_csv('emotion-labels-test.csv')

In [3]: train_df.head()
```

Out[3]:

| | text | label |
|---|---|---|
| 0 | Just got back from seeing @GaryDelaney in Burs... | joy |
| 1 | Oh dear an evening of absolute hilarity I don'... | joy |
| 2 | Been waiting all week for this game ❤❤❤ #ch... | joy |
| 3 | @gardiner_love : Thank you so much, Gloria! Yo... | joy |
| 4 | I feel so blessed to work with the family that... | joy |

```
In [4]: train_df.shape
```

Out[4]: (3613, 2)

**2. Text Processing:**

The function listed below converts the class label to the label-id.

```
def label_encoding(label):
    if label=='fear':
        return 0
    elif label=='anger':
        return 1
    elif label=='joy':
        return 2
    else :
        return 3
```

```
train_df['label'] = train_df['label'].apply(label_encoding)
```

The function below accepts text as input and performs all preparation procedures required to clean the text data.

```
import re
def preprocess(text):
    text = text.replace("n\'t", " not")
    text = text.replace("n't", " not")
    text = text.replace("\'ve", " have")
    text = text.replace("\'m", " am")
    text = text.replace("\'re", " are")
    text = text.replace("\'s", " is")
    text = text.replace("\'ll", " will")
    text = " ".join(text.split())
    text = re.sub('[^A-Za-z0-9]+', ' ', text)
    text = " ".join(text.split())
    return text.lower()
```

```
train_df['text'] = train_df['text'].apply(preprocess)
train_df['text'].head()
```

```
0    just got back from seeing garydelaney in bursl...
1    oh dear an evening of absolute hilarity i do n...
2     been waiting all week for this game cheer friday
3    gardiner love thank you so much gloria you are...
4    i feel so blessed to work with the family that...
Name: text, dtype: object
```

### 3. Feature Extraction:

The code part below discusses feature extraction.

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
tfidf.fit(train_df['text'])
train_tfidf = tfidf.transform(train_df['text'])
test_tfidf = tfidf.transform(test_df['text'])
```

```
train_tfidf.shape,test_tfidf.shape
```

```
((3613, 10187), (3142, 10187))
```

### 4. Train Test Split:

The train test split is implemented using the code below.

**Train Test Split**

```
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(train_tfidf, train_df['label'], test_size=0.2, stratify=train_df['label'],randc
```

**5. Modeling:** Two models were used in this project.

1. Multinomial Naive Bayes

2. Multinomial Naive Bayes with Cross Validation

### 1) Multinomial Naive Bayes:

**Multinomial Naive Bayes Model**

```
from sklearn.naive_bayes import MultinomialNB

clf = MultinomialNB()
clf.fit(X_train,y_train)
```

```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

**2) Multinomial Naive Bayes with Cross Validation:**

## Cross Validation Using Grid SearchCV

```
from sklearn.model_selection import GridSearchCV
import matplotlib.pyplot as plt

params = {'alpha':[0.001,0.01,0.1,1,10,100]}
cv = GridSearchCV(clf, params, cv=10, scoring='accuracy', return_train_score=True)
cv.fit(X_train, y_train)
```

```
GridSearchCV(cv=10, error_score=nan,
             estimator=MultinomialNB(alpha=1.0, class_prior=None,
                                     fit_prior=True),
             iid='deprecated', n_jobs=None,
             param_grid={'alpha': [0.001, 0.01, 0.1, 1, 10, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='accuracy', verbose=0)
```

**8. <u>Preliminary Results:</u>** As previously stated, we have train and validation datasets in addition to the test dataset. We also stated in the preceding section that we have tried two models so far. The accuracy and confusion matrix can be seen in the screenshots below.

1. Multinomial Naive Bayes

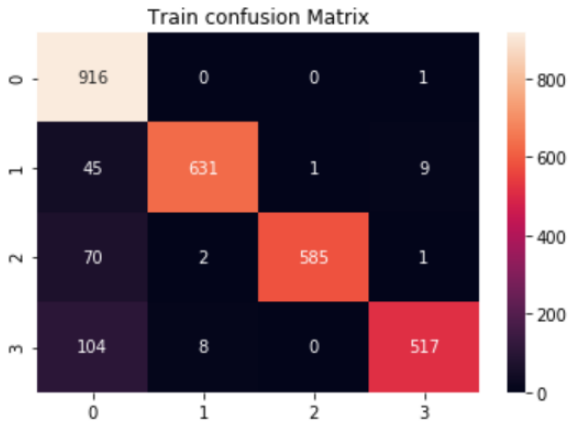
## Accuracy Score

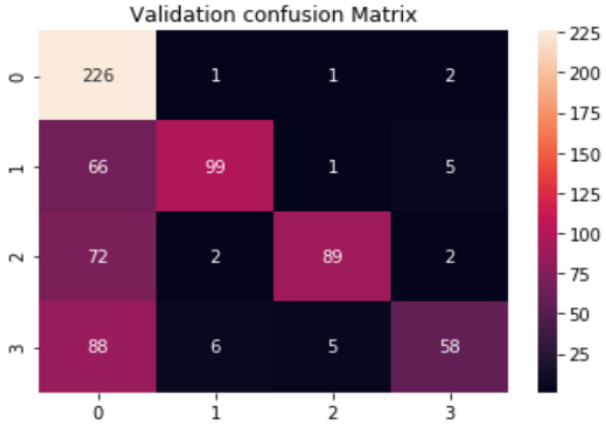
```
from sklearn.metrics import accuracy_score,confusion_matrix,roc_auc_score

print("The Train accuracy score is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy score is : ",accuracy_score(y_val,predicted_val))
```

```
The Train accuracy score is :  0.9166089965397924
The Validation accuracy score is :  0.65283540802213
```

Train and Validation Confusion Matrix:



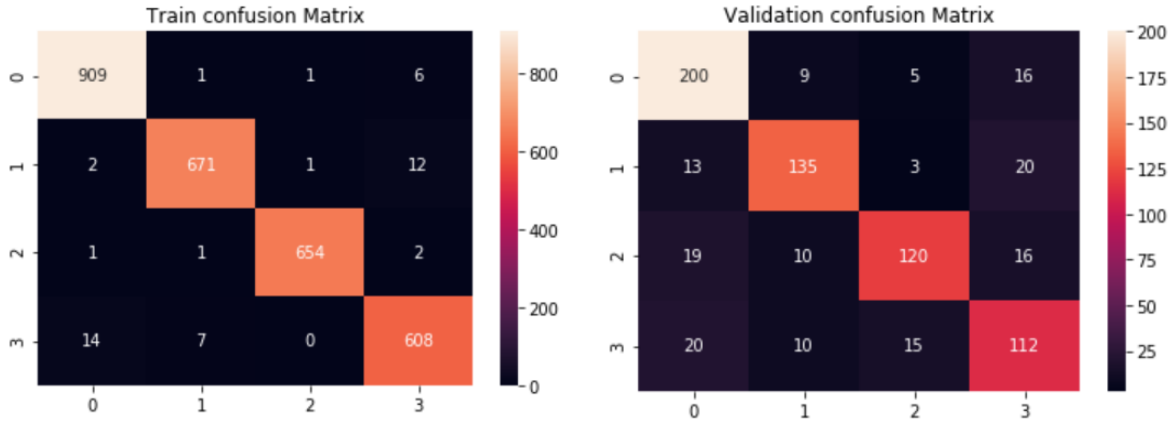
2. Multinomial Naive Bayes with Cross Validation

## Accuracy Score

```
print("The Train accuracy score is : ",accuracy_score(y_train,predicted_train))
print("The Validation accuracy score is : ",accuracy_score(y_val,predicted_val))
```

```
The Train accuracy score is :  0.9833910034602076
The Validation accuracy score is :  0.7842323651452282
```

Train and Validation Confusion Matrix:




## 9. Project Management:

## Work Completed :

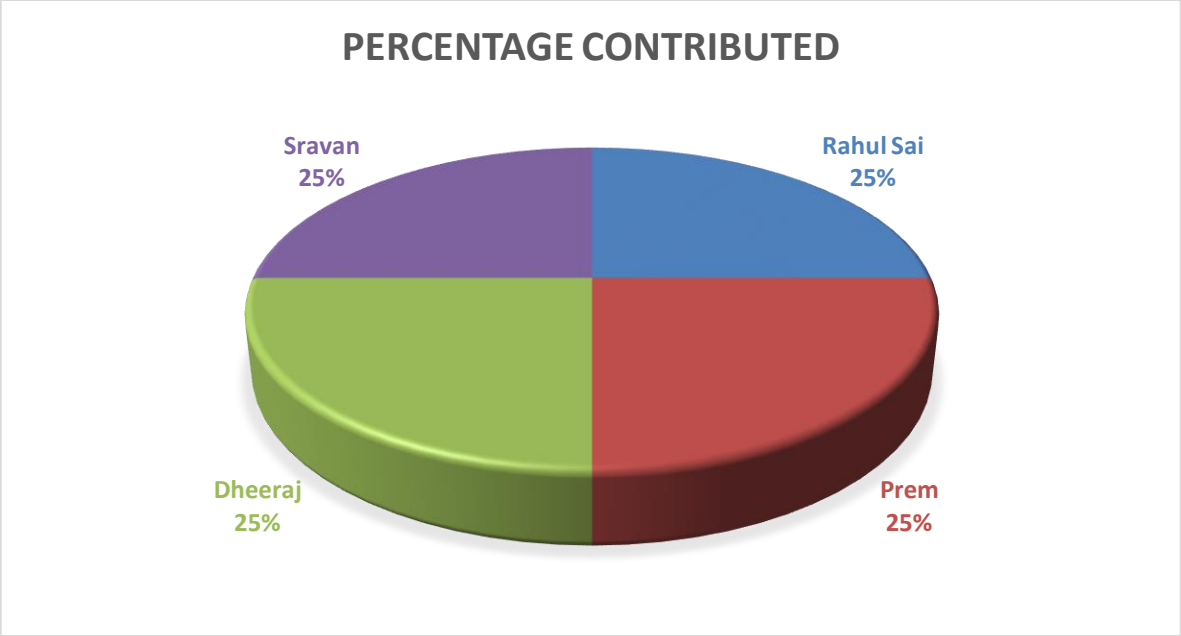
• **Description:** The Text Emotion Detection was chosen primarily because the class labels are connected to the sentiment analysis problem and more closely resemble the intent classification problem utilized in chatbots. Our first step was to get hold of a dataset which we found it on Kaggle which is an ocean of data. Preparing the data to perform analysis is the next step. Thereafter we processed the extracted text data. Before Training the test split data we performed Feature Extraction. Modelling involved Multinomial naïve Bayes Cross Validation model and Multinomial Naïve Bayes Model.

• **Responsibility (Task, Person):**

- Gathering dataset and processing the dataset – Prem Sai, Rahul
- Converting the text data into numerical Features – Rahul, Dheeraj
- Analyzing the numerical data and separating the data – Sravan, Prem Sai
- Text Processing and Feature Extraction – Sravan, Dheeraj

- Train Test Split & Modeling – Rahul, Sravan
- Calculation accuracy score – Dheeraj, Prem

• **Contributions (members/percentage):**




**Work to be completed:**

**• Description:** In the next increment we will be dealing with Combination of Feature vectors received after Feature extraction. Following that we will be performing SVM-based classification which involves which involves classification of linear as well as non-linear data. Finally with the help of the classified data we build a model to predict the emotion of the provided text. The main advantage of this classification is they provide high speed and better performance with just a limited amount of data.

• **Responsibility (Task, Person):**

- Generation of Feature Vectors Extraction – Prem Sai
- SVM Based classification - Rahul
- Prediction of Emotion using the created model – Dheeraj, Sravan

• **Issues/Concerns:** N/A

## 8.References:


1. https://towardsdatascience.com/machine-learning-nlp-text-classification-using-scikit-learn-python and-nltk-c52b92a7c73a
2. https://stackabuse.com/text-classification-with-python-and-scikit-learn/
3. https://www.datacamp.com/tutorial/naive-bayes-scikit-learn
4. https://www.kaggle.com/datasets/anjaneyatripathi/emotion-classification-nlp
5. https://www.javatpoint.com/machine-learning-naive-bayes-classifier