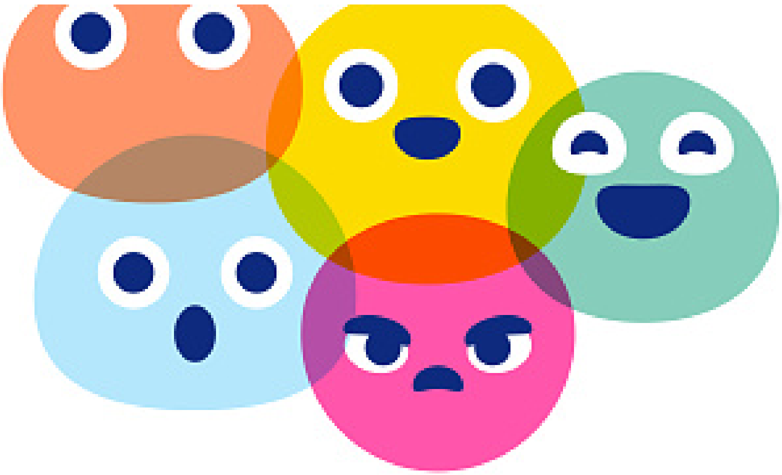


Twitter

Description	Human Emotion Classification Using Twitter Sentiment Analysis
Tech Stack	NLTK Python

▼ Table of Contents

Business Problem

This project aims to solve the business problem of understanding and leveraging user emotions expressed on Twitter, a platform where individuals freely share their views on products, services, events, and various subjects. The objective is to classify the emotions conveyed in text data, determining

Assumptions

- We assume that the Twitter data collected for this project is representative of the broader population's emotions. The assumption is based on the belief that Twitter users express a diverse range of emotions that can be analyzed effectively.

whether they are positive, negative, or belong to a specified set of emotional categories.

- We assume that the data collected adheres to Twitter's terms of service and policies, and that the data collection process respects user privacy and anonymity.

Research Questions

1. Can we accurately figure out people's feelings on Twitter and put them into categories?
2. What makes it hard to understand complicated emotions on platforms like Twitter, and what are the problems we face?
3. How can we use Twitter emotion recognition in practical ways, like helping businesses, learning public opinions, or monitoring mental health?
4. How do we measure and make our system better at aiding businesses, policymakers, and individuals?
5. What can we do to make our system understand more emotions and languages, so it can be used in even more situations?

Hypothesis

1. Emotion classification on social media data will provide actionable insights for businesses, enabling them to improve customer satisfaction and engagement.
2. Natural language processing techniques, such as tokenization and stemming, will significantly improve the performance of the emotion classification system.
3. The sentiment analysis and emotion classification techniques applied to Twitter data will effectively categorize tweets into predefined emotion categories. There is a statistically significant relationship between the features extracted from tweets and the emotional content they represent.

Introduction

In today's digital age, social media platforms have become a rich source of human expressions, notably on Twitter. This project seeks to understand and categorize the emotions conveyed in tweets. The primary goal is to develop a robust sentiment analysis system capable of classifying tweets into predefined emotion categories. This endeavor carries significance across various domains, from enhancing customer experiences and product development to tracking public sentiment during critical events. By deciphering these digital emotions, this project aspires to unlock valuable insights that extend beyond the online world.

Significance and Relevance

The significance of this project lies in its applicability to various domains, including marketing, public opinion analysis, mental health assessment, and sentiment-driven product development. Understanding user emotions in real time allows organizations to respond to customer needs, identify trends, and improve customer satisfaction.

Businesses and organizations seek to leverage this valuable source of information for the following purposes:

- **Customer Feedback Analysis**: To gather insights into customer opinions and feedback regarding products and services. Understanding whether customers express positive, negative, or neutral sentiments is crucial for improving offerings and addressing issues.

- **Brand and Reputation Management**: To monitor and manage the reputation of a brand or organization. This includes identifying and addressing negative sentiment or potential PR crises promptly.
- **Marketing and Product Development**: To tailor marketing campaigns and develop products based on the sentiments and preferences of the target audience. Emotion classification can help identify trends and opportunities.
- **Public Opinion Monitoring**: For political, social, and cultural analysis. Emotion classification on social media can provide insights into public sentiment regarding important events and topics.
- **Mental Health Assessment**: In the healthcare sector, understanding and monitoring emotional expressions on social media can be used for mental health assessment and early intervention.
- **Customer Satisfaction and Engagement**: For businesses that use social media for customer engagement, understanding and responding to customer emotions can enhance satisfaction and loyalty.

Dataset

The dataset used for this sentiment analysis project is sourced from Kaggle.

It comprises 6 columns and 1,600,000 rows of data. The columns include 'target,' 'ids,' 'date,' 'flag,' 'user,' and 'text.'

- 'target' contains labels, with 0 and 4 representing negative and positive emotions, respectively.
- 'ids' consists of unique identifiers assigned to each tweet.
- 'date' indicates the timestamp of when the tweet was posted.
- 'user' contains the Twitter username of the person who posted the tweet.

Finally, the 'text' column contains the actual tweet.

Work Flow

Gather the requirements —→ Load the dataset —→ Perform Exploratory Data Analysis —→ Text Preprocessing —→ Splitting the dataset —→ Model building —→ Model Evaluation

1. Elementary Work

Import Libraries

The first step in the project is to identify and import the necessary python libraries for the analysis. Numpy, Pandas, Matplotlib and Seaborn are the primary requirements for the project. The Natural Language Processing toolkit is also required further in the analysis to extract text features.

```
# utilities
import re
import numpy as np
import pandas as pd

# plotting
import seaborn as sns
import matplotlib.pyplot as plt


# nltk
```

```
import nltk
from nltk.stem import WordNetLemmatizer

# sklearn
from sklearn.linear_model import LogisticReg
!pip install xgboost
from xgboost import XGBClassifier
from sklearn.model_selection import train_tes
from sklearn.feature_extraction.text import Tf
from sklearn.metrics import confusion_matrix

#metrics
from sklearn.metrics import accuracy_score,

#import warnings
import warnings
warnings.filterwarnings('ignore')
```

Load the dataset

The next step is to load the dataset into our workspace. The Pandas ‘read.csv()’ function is used to import the dataset. Then let us have a look at the sample of the data.

```
DATASET_COLUMNS=['target','ids','date','flag
DATASET_ENCODING = "ISO-8859-1"

df = pd.read_csv("E:\\Dataset.csv",encoding=
df.sample(5)
```

	target	ids	date	flag	user	text
532890	0	2196930856	Tue Jun 16 13:07:03 PDT 2009	NO_QUERY	justinjood	@FemProMom jealous IM VERY HUNGRY!
1096819	4	1970345864	Sat May 30 03:29:21 PDT 2009	NO_QUERY	LauraGuthrie	I woke up before my alarm... at 5:45am. Weird....
814713	4	1550907065	Sat Apr 18 07:36:04 PDT 2009	NO_QUERY	evolutionofaman	Good morning, Twitterbugs. I'm feeling very in...
637116	0	2234103159	Thu Jun 18 22:26:05 PDT 2009	NO_QUERY	stephhhaniiieee	@Ricksauce_10 i almost got hit by the ball lo...
566385	0	2206826605	Wed Jun 17 07:06:32 PDT 2009	NO_QUERY	precisionglass	sick of this wind and rain

2. Exploratory Data Analysis

Let us first have a look at the first five rows of the dataset.

```
df.head(5)
```

	target	ids	date	flag	user	text
0	0	1467810369	Mon Apr 06 22:19:45 PDT 2009	NO_QUERY	_TheSpecialOne_	@switchfoot http://twitpic.com/2y1zl - Awww, t...
1	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
2	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
3	0	1467811184	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire
4	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....

The provided code shows the number of rows and columns present in the dataset. Upon

```
df.shape
```

execution, it is found that there are 1600000 rows of data each having 6 attributes.

```
df.dtypes
```

In order to work with the dataset, it is important to know the data types present in it.

```
target      int64
ids         int64
date       object
flag       object
user       object
text       object
dtype: object
```

```
print(df.columns)
```

Similarly, let us look at the attribute names also.

```
Index(['target', 'ids', 'date', 'flag', 'user', 'text'], dtype='object')
```

> Checking for missing values

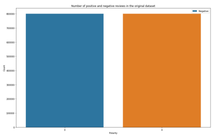
```
print(df.isnull().sum().sum())
```

Presence of missing data can cause interruptions in the text preprocessing. Hence it is important to identify them and fill them with appropriate values. Fortunately, it is observed that there is no missing data in the dataset we are using.

```
plt.figure(figsize=(16,10))
sns.countplot(x='target',data=df)
plt.legend(['Negative','Positive'])
plt.xlabel('Polarity')
plt.ylabel('Count')
plt.title('Number of positive and negative reviews in the original dataset')
plt.show()
```

Data Distribution

Let us now look at the data distribution that is the number of tweets having positive and negative polarity.



There are equal number of positive and negative polarity tweets in the dataset.

3. Text Preprocessing

Now as we have got a summary of the data present with us, we now proceed and clean the data.

This is an important step in any model building project as dirty and sometimes even a large amount of data can make our work difficult.

Hence, it is essential to pre-process the data before proceeding.

This involves removing all the unnecessary urls, hashtags, geographic locations, repeated words(here prepositions, conjunctions, pronouns, nouns which do not actually represent the emotion behind the tweet are removed), numbers, etc.,.

The primary features on focus are "text" and "target," which are determined by the business problem.

```
tweet_data=df[['target','text']]
tweet_data.sample(5)
```

Convert text to lower case

```
tweet_data['text']=tweet_data['text'].str.lowe
df=pd.DataFrame(tweet_data['text'])
df.head(5)
```

The first step in the text preprocessing is to convert all the available text into lower case format. This ensures uniformity in the data.

```
stopwordlist = ['a', 'about', 'above', 'after', 'ag
           'and','any','are', 'as', 'at', 'be', 'becaus
           'being', 'below', 'between','both', 'by',
           'does', 'doing', 'down', 'during', 'each
           'further', 'had', 'has', 'have', 'having',
           'hers', 'herself', 'him', 'himself', 'his', '
           'into','is', 'it', 'its', 'itself', 'just', 'll', 'm',
           'me', 'more', 'most','my', 'myself', 'nov
           'only', 'or', 'other', 'our', 'ours','oursel
             'same', 'she', "shes", 'should', "shou
           't', 'than', 'that', "thatll", 'the', 'their', 't
           'themselves', 'then', 'there', 'these', 't
           'through', 'to', 'too','under', 'until', 'up'
           'we', 'were', 'what', 'when', 'where','w
           'why', 'will', 'with', 'won', 'y', 'you', "yo
           "youve", 'your', 'yours', 'yourself', 'yo
```

Remove stop words

Next, we remove the stop words from the tweets. Stop words are common, non-essential words in a language, like "the" and "in." Removing them in text processing streamlines analysis, focusing on meaningful content. We need to define the stop words first and then remove them from the text.

```
STOPWORDS = set(stopwordlist)

### this function removes the stopwords fror
def remove_stopwords(text):
    return " ".join([word for word in str(text).sp

tweet_data['text'] = tweet_data['text'].apply(l
df=pd.DataFrame(tweet_data['text'])
df.head(5)
```

text
@switchfoot http://twitpic.com/2y1zl - Awww, t...
is upset that he can't update his Facebook by ...
@Kenichan I dived many times for the ball. Man...
my whole body feels itchy and like its on fire
@nationwideclass no, it's not behaving at all....

Before removing stop words

	text
0	@switchfoot http://twitpic.com/2y1zl - awww, t...
1	upset can't update facebook texting it... migh...
2	@kenichan dived many times ball. managed save ...
3	whole body feels itchy like fire
4	@nationwideclass no, it's not behaving all. i'...

After removing stop words

Remove repeated words

In text preprocessing, we remove repeated words to enhance data quality and reduce redundancy, making analysis more efficient

```
def removing_repeating_char(text):
    return re.sub(r'(.)1+', r'1', text)

tweet_data['text'] = tweet_data['text'].apply(l
```

and improving the accuracy of natural language processing tasks.

```
tweet_data['text'].head()
df=pd.DataFrame(tweet_data['text'])
df.head(5)
```

	text
0	@switchfoot http://twitpic.com/21zl - awww, th...
1	upset can't update facebook texting it... migh...
2	@kenichan dived many times ball. managed save ...
3	whole body feels itchy like fire
4	@nationwideclass no, it's not behaving all. i'...

Removing the repeated words

Remove URLs

Urls are often removed in text preprocessing for privacy, security, and analysis purposes. Removing them can protect sensitive information, improve data clarity, and prevent biased insights when analyzing text data.

```
def removing_URLs(text):
    return re.sub('((www.[^s]+)|(https?://[^s]+)

tweet_data['text'] = tweet_data['text'].apply(l
df=pd.DataFrame(tweet_data['text'])
df.head(5)
```

	text
0	@switchfoot s bummer. shoulda got david carr ...
1	upset can't update facebook texting it... migh...
2	@kenichan dived many times ball. managed save ...
3	whole body feels itchy like fire
4	@nationwideclass no, it's not behaving all. i'...

Removing URLs from the text

Removing numerical data

Numericals are often removed in text preprocessing to focus on text-based patterns and semantics. Removing them can simplify analysis and make text data more suitable for natural language processing tasks. Numericals may not contribute to the desired linguistic understanding or patterns being sought in the text and hence need to be removed from the text.

```
def removing_numbers(data):
    return re.sub('[0-9]+', '', data)

tweet_data['text'] = tweet_data['text'].apply(l
df=pd.DataFrame(tweet_data['text'])
df.head(5
```

	text
0	@switchfoot s bummer. shoulda got david carr ...
1	upset can't update facebook texting it... migh...
2	@kenichan dived many times ball. managed save ...
3	whole body feels itchy like fire
4	@nationwideclass no, it's not behaving all. i'...

Removing numbers

Stemming

Stemming is the process of reducing words to their root or base form. It's needed in text preprocessing to standardize variations of words. For example, "jumping" and "jumps" both stem to "jump," helping text analysis focus on core meaning and improving the efficiency of natural language processing tasks. Stemming is especially useful for search engines, information retrieval, and text mining, where reducing words to their common form helps in matching and retrieval.

```
st = nltk.PorterStemmer()
def stemming_on_tweets(data):
    text = [st.stem(word) for word in data]
    return data

tweet_data['text']= tweet_data['text'].apply(la
df=pd.DataFrame(tweet_data['text'])
df.head(5)
```

	text
0	@switchfoot s bummer. shoulda got david carr ...
1	upset can't update facebook texting it... migh...
2	@kenichan dived many times ball. managed save ...
3	whole body feels itchy like fire
4	@nationwideclass no, it's not behaving all. i'...

Stemming

4. Model Building

Identify dependent and independent variables

First, we convert the ‘object’ type data to string type and identify the target variable and the predictor variable. In this case, the predictor variable is the tweet text and the target variable is the polarity of that tweet.

```
X=tweet_data.text.astype(str)
y=tweet_data.target.astype(str)
```

Split the dataset

Then we split the data into train and test sets in the ratio of 3:1.

```
X_train, X_test, y_train, y_test = train_test_spli
```

Vectorization

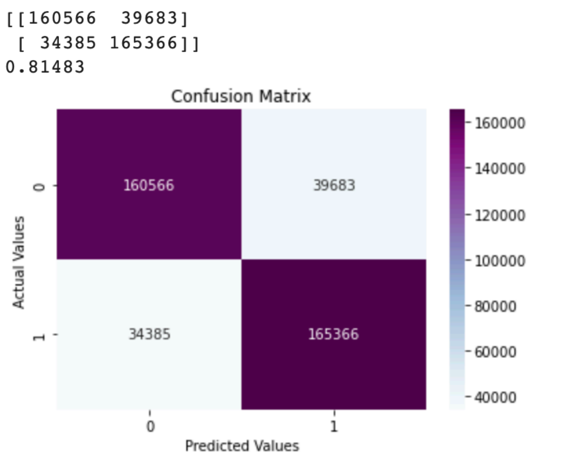
In natural language processing (NLP), vectorization involves transforming text data into numerical vectors, making it suitable for mathematical operations. Here, TF-IDF vectorization is performed to do the same. TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a specific method of vectorization used in NLP. It assigns numerical values to words in a document, with the goal of capturing the importance of each word in the document relative to a larger collection of documents. Upon execution, it is seen that there are 50000 features words.

```
vectorizer = TfidfVectorizer(ngram_range=(1,
vectorizer.fit(X_train)
print('No. of feature_words: ', len(vectorizer.g
```

```
X_train = vectorizer.transform(X_train)
X_test = vectorizer.transform(X_test)
```

Model Building and Evaluation

Finally, we build the model using logistic regression and evaluate it. Using logistic regression, we get around 81% accuracy.



Confusion matrix

```
def model_evaluation(model):

# Predict values for Test dataset
   y_pred = model.predict(X_test)

# Computing and plotting the Confusion matr
   cf_matrix = confusion_matrix(y_test, y_prec
   print(cf_matrix)
   sns.heatmap(cf_matrix,fmt='d',cmap='BuP
   plt.xlabel('Predicted Values')
   plt.ylabel('Actual Values')
   plt.title('Confusion Matrix')
   print(accuracy_score(y_test,y_pred))
```

```
logistic_model = LogisticRegression(C = 2, m
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_tes
model_evaluation(logistic_model)
```

```
def predict_emotion(s):
   s=[s]
   s=remove_stopwords(s)
```

Generalization

Generalizing the functionality of the code is crucial to enhance its usability. In this context, we establish a dedicated function aimed at predicting the polarity of provided text. This step ensures that the code can be easily applied across various scenarios and use cases, making it more versatile and accessible to a broader range of users.

```
    s=removing_repeating_char(s)
    s=removing_URLs(s)
    s=removing_numbers(s)
    s=removing_numbers(s)
    s=stemming_on_tweets(s)
    s=vectorizer.transform([s])
    ans=logistic_model.predict(s)
    if ans=='4':
        return "Sentiment: Positive"
    elif ans=='0':
        return "Sentiment: Negative"
```

```
# random text input
print(predict_emotion('I liked the food at the restaurant'))
```

```
Sentiment: Positive
```

Conclusion

- The project on "Human Emotion Recognition Using Twitter Sentiment Analysis" successfully categorizes a wide range of emotions expressed on Twitter.
- The findings have practical applications in marketing, public opinion research, and mental health monitoring.
- The model can assist businesses in adapting strategies, help policymakers gauge public sentiment, and offer support to individuals.
- Future work can expand the model to encompass more nuanced emotions and languages.
- This project exemplifies the power of data-driven approaches in understanding human sentiment on social media.