CHAT BASED SENTIMENT ANALYSIS USING LOGISTIC REGRESSION

[Document subtitle]

AUGUST 3, 2025

[J. SNEHASRI]

Abstract

In today's digital world, chat platforms have become a major form of communication. These chat messages often carry emotional meaning, which can be useful for improving services, understanding users, and enhancing interactions. However, analyzing such short and informal messages can be challenging due to slang, emojis, and abbreviations.

This project aims to build a **Chat-Based Sentiment Analysis System** using **Logistic Regression**, a simple yet effective classification algorithm. It focuses on identifying the sentiment—positive, negative, or neutral—of chat messages in a fast and efficient manner.

The system uses basic Natural Language Processing (NLP) steps such as cleaning the text and converting it into numerical form using **TF-IDF vectorization**. This processed data is then used to train the Logistic Regression model, which is tested and evaluated using common metrics like accuracy and F1-score.

Even though Logistic Regression is a simple model, it performs well on clean and structured text data. It is lightweight, fast, and suitable for real-time applications like chatbots or customer feedback systems, making it a practical choice for sentiment analysis in chats.

Table of contents

SL.NO	**Topic**	**Page.No**

Introduction

In the age of digital communication, chat-based platforms have become a primary mode for user interaction in domains such as customer service, social media, e-commerce, and healthcare. With the vast volume of conversational data being generated, it becomes essential to automatically analyse and understand the underlying sentiment of these messages. This process is known as chat-based sentiment analysis.

It involves classifying short, informal messages into categories like positive, negative, or neutral. Chat data presents unique challenges due to slang, emojis, abbreviations, and context-dependent language.

To address this, the system uses Logistic Regression, a simple yet effective classification model. After cleaning and preprocessing the messages, TF-IDF vectorization is applied to convert text into numerical features. These are used to train the model to predict sentiment accurately.

The key objectives of this system are:

- To automatically detect sentiment in chat messages.
- To ensure reliable accuracy with a lightweight model.
- To provide real-time feedback for improving user interaction.

Using Logistic Regression ensures a balance of performance, speed, and ease of interpretation — ideal for scalable sentiment analysis in real-time chat environments.

Problem Statement

With a rapid growth of online communication , chat platforms are a primary channel for people to communicate with businesses, services, and one another. These conversations hold valuable emotional cues that, when properly analyzed, can help companies improve their decision-making, support systems, and overall customer experience.

However, understanding sentiment in chat messages isn’t simple. Unlike structured formats like reviews or articles, chat messages are usually short, casual, and packed with slang, abbreviations, or emojis. This informal nature makes it hard for traditional sentiment analysis methods to perform well.

The main challenge we aim to solve is:
"How can we accurately classify the sentiment (positive, negative, or neutral) of informal and short chat messages using a model that is both simple and reliable?"

To address this, we propose using the Logistic Regression algorithm, which offers a balance of speed, simplicity, and accuracy. The system will:

- Preprocess chat text by cleaning, normalizing, and removing noise.
- Convert text into numerical form using TF-IDF vectorization.
- Train a Logistic Regression classifier on a labeled sentiment dataset.
- Predict the sentiment of new or real-time chat messages efficiently.

This approach provides a lightweight and effective solution for chat-based sentiment analysis, suitable for applications like customer service, chatbots, and social media tools.

Objectives

The aim of this project is to create a sentiment analysis system tailored for short and informal chat messages. Since chats often include slang, emojis, and abbreviations, traditional models struggle with accuracy. To handle this challenge, the system uses the **Logistic Regression algorithm**, which is simple, fast, and effective for classifying text.

The process involves **cleaning the text**, converting it into numerical data using **TFIDF vectorization**, and training the Logistic Regression model to identify the **sentiment** behind each message—Positive, Negative, or Neutral. The end goal is to build a lightweight system that works well in **real-time applications** like chatbots, feedback tools, or support platforms.

Specific Objectives Include:

1. To classify chat messages into Positive, Negative, or Neutral sentiments.
2. To use Logistic Regression, a simple and interpretable algorithm, for building the sentiment model.
3. To preprocess informal chat messages by removing slang, emojis, and unwanted characters.
4. To apply TF-IDF vectorization for converting cleaned text into numerical features.
5. To evaluate model performance using accuracy, precision, recall, and F1-score.
6. To make the system capable of handling real-time user input and predicting sentiment on the fly.
7. To demonstrate how Logistic Regression can still be useful in practical, real-world sentiment analysis tasks.

Scope of the project

Included:

- **Textual Chat Data**: The system works on user chat messages written in text format.
- **NLP Techniques**: Basic natural language processing techniques like stopword removal, lowercasing, and punctuation cleanup are applied to clean the text.
- **TF-IDF Feature Extraction**: Chat messages are converted into numeric form using TF-IDF to make them suitable for training the model.
- **Logistic Regression Model**: A simple but effective model used to classify chat sentiment into positive, negative, or neutral categories.
- **Sentiment Output**: The system predicts the overall tone of each message—positive, negative, or neutral.

Excluded:

- **Non-Text Inputs**: Audio, images, or video data from chat platforms are not processed by the system.
- **Contextual Understanding**: User location, past conversation history, or context is not considered in sentiment detection.
- **Sarcasm & Deep Emotions**: The model may struggle with sarcasm or complex emotions that require deeper context.
- **Deep Learning Models**: This project does not use advanced architectures like LSTM, GRU, or BERT.
- **Human Feedback Loop**: It doesn't include manual correction or continuous improvement from human reviewers.

Comparison

Algorithm	**Strengths**	**Weaknesses**	**Compared to Logistic Regression**
Logistic Regression	Fast, simple, easy to interpret. Works well on linearly separable data.	May underperform on complex or non-linear patterns.	Baseline model—great for fast, interpretable results.
XGBoost	High accuracy, handles complex data, regularization avoids overfitting.	Longer training time, more complex to tune.	More powerful than Logistic Regression, but harder to interpret and slower.
Naive Bayes	Extremely fast, good with high-dimensional data like text.	Assumes independence between features; struggles with imbalanced data.	Simpler but less accurate; Logistic Regression is more robust.
SVM	Effective for small- to medium-sized datasets; good for margin-based learning.	Slow for large datasets; kernel tuning required.	Comparable or better in accuracy, but Logistic Regression is faster.
Random Forest	Robust, handles nonlinearity, less prone to overfitting.	Less interpretable, slower predictions.	Performs better on complex data; Logistic Regression is simpler and faster.
LightGBM / CatBoost	Fast boosting alternatives, handle categorical features well.	Require tuning and understanding of boosting trees.	Advanced algorithms; Logistic Regression is easier to implement.

Methodology

The Chat-Based Sentiment Analysis System follows a streamlined and effective workflow. It moves from gathering raw chat data to preprocessing it, training a Logistic Regression model, and predicting sentiments such as positive, negative, or neutral in real time.

1. Data Collection

- The dataset used is: final_dataset.csv
- It contains two key columns:
  - Text → the actual chat message
  - Sentiment → the sentiment label for the message ○ Data size : 6000+ rows **○ Target Format (CSV):**

Text	**Reaction**
That's awesome!	Positive
Not sure what to say	Neutral
Completely disappointed	Negative

○ Requirements:

- Well-balanced classes: **positive**, **negative**, **neutral**
- CSV format for easy processing

2. Data Preprocessing

- **Text Cleaning**: Remove emojis, punctuation, symbols, links, etc.

- **Lowercasing**: Convert everything to lowercase
- **(Optional)** Stopword removal for reducing noise
- **Missing Values**: Drop rows with missing or null values

3. Label Encoding

- Sentiment labels (positive, neutral, negative) are encoded into numerical form: ○ Positive → 2 ○ Neutral → 1 ○ Negative → 0
- This is done using LabelEncoder from sklearn.preprocessing

4. Feature Extraction

- Use **TF-IDF Vectorizer** to transform text into numerical vectors
- This helps Logistic Regression understand the importance of words in messages

5. Data Splitting

- Dataset is split using train_test_split:
    - **80%** used for training , **20%** used for testing

6. Model Building using Logistic Regression

- A **Logistic Regression** model is trained using the TF-IDF features
- It is chosen for:

o Fast training o Good

baseline accuracy o

Interpretability

7. Model Evaluation

- Once trained, the model is evaluated using:
  - **Accuracy Score**
  - **Precision, Recall, F1-score** from classification_report
- These metrics help measure how well the model performs on unseen messages

8. User Input and Prediction

- The system allows real-time input:
  - A user enters a message → it's preprocessed and vectorized → passed into the model
  - The model outputs the predicted sentiment

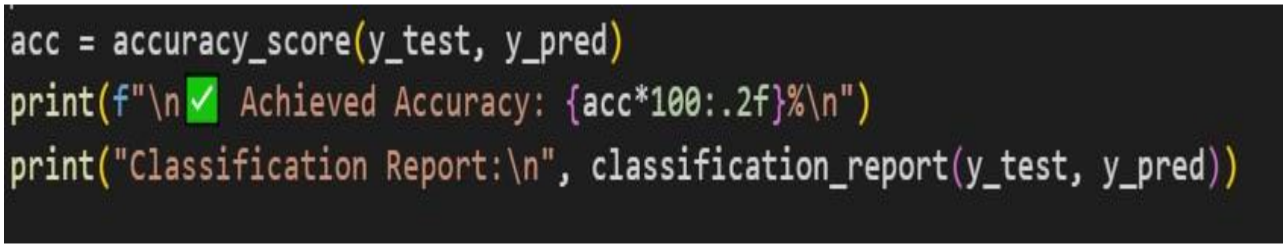
Implementation

The chat-based sentiment analysis system is built using Python and popular machine learning libraries. It focuses on preprocessing chat messages, extracting features, and training a Logistic Regression model. The aim is to accurately predict the sentiment of each message.

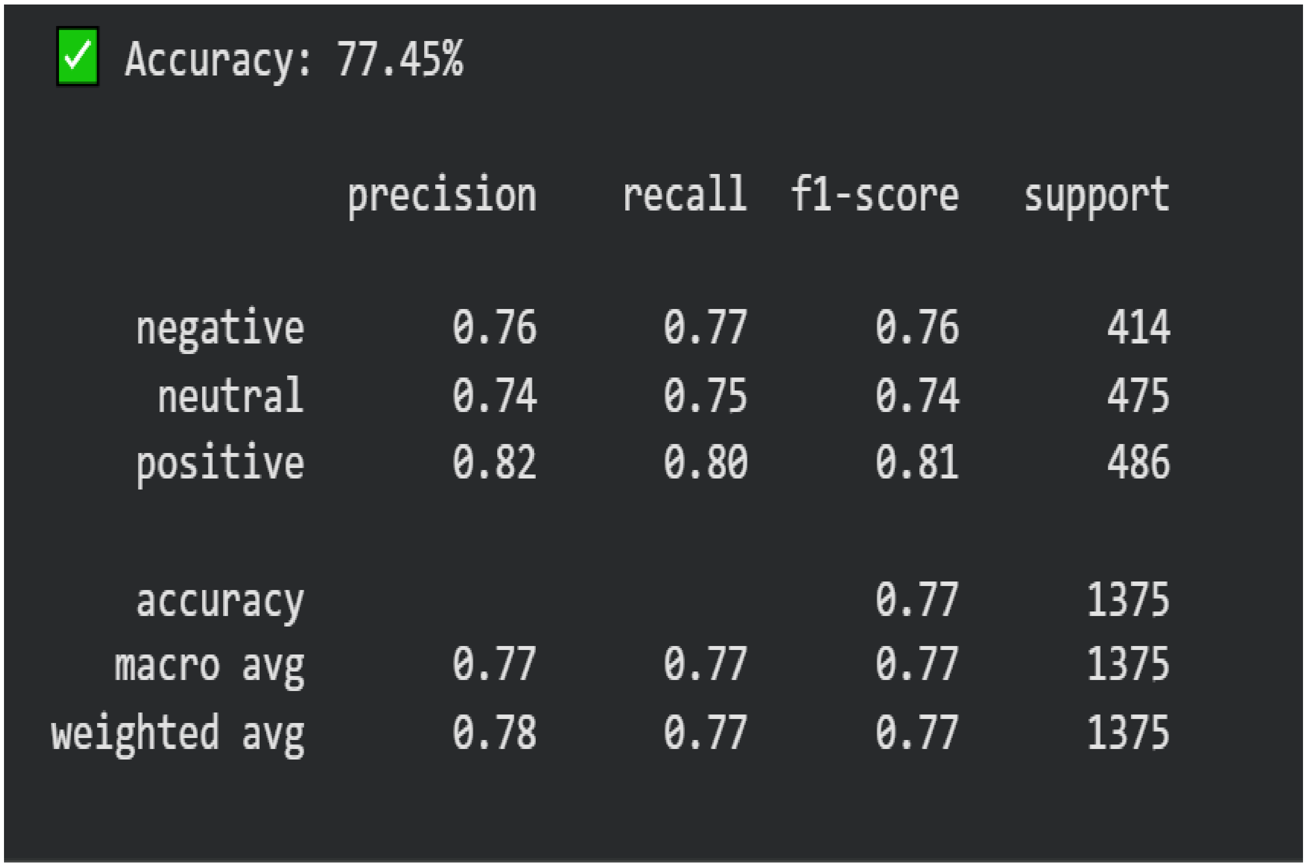
Tools and Libraries used:

- **Pandas**: for data manipulation and analysis.
- **Re**: for regular expressions, specifically for cleaning text.
- **Numpy**: for numerical operations.
- **Matplotlib.pyplot** and **Seaborn**: for data visualization (plotting null values, class distribution, and creating a confusion matrix).
- **Joblib**: for saving and loading the trained model, vectorizer, and label encoder.
- **Sklearn.model_selection.train_test_split**: for splitting the dataset into training and testing sets.
- **Sklearn.preprocessing.LabelEncoder**: for encoding categorical labels (sentiments) into numerical format.
- **Sklearn.feature_extraction.text.TfidfVectorizer**: for converting text data into numerical TF-IDF features.
- **Sklearn.pipeline.FeatureUnion**: for combining the features from the word and character TF-IDF vectorizers.
- **Sklearn.linear_model.LogisticRegression**: for the sentiment classification model.
- **Sklearn.metrics (classification_report, accuracy_score, confusion_matrix)**: for evaluating the model's performance.
- **Sklearn.utils.class_weight.compute_class_weight**: for calculating class weights to handle the imbalanced dataset.
- **imblearn.over_sampling.SMOTE**: for oversampling the minority classes to address class imbalance.
- **scipy.sparse.hstack**: for horizontally stacking sparse matrices (the vectorized text data).

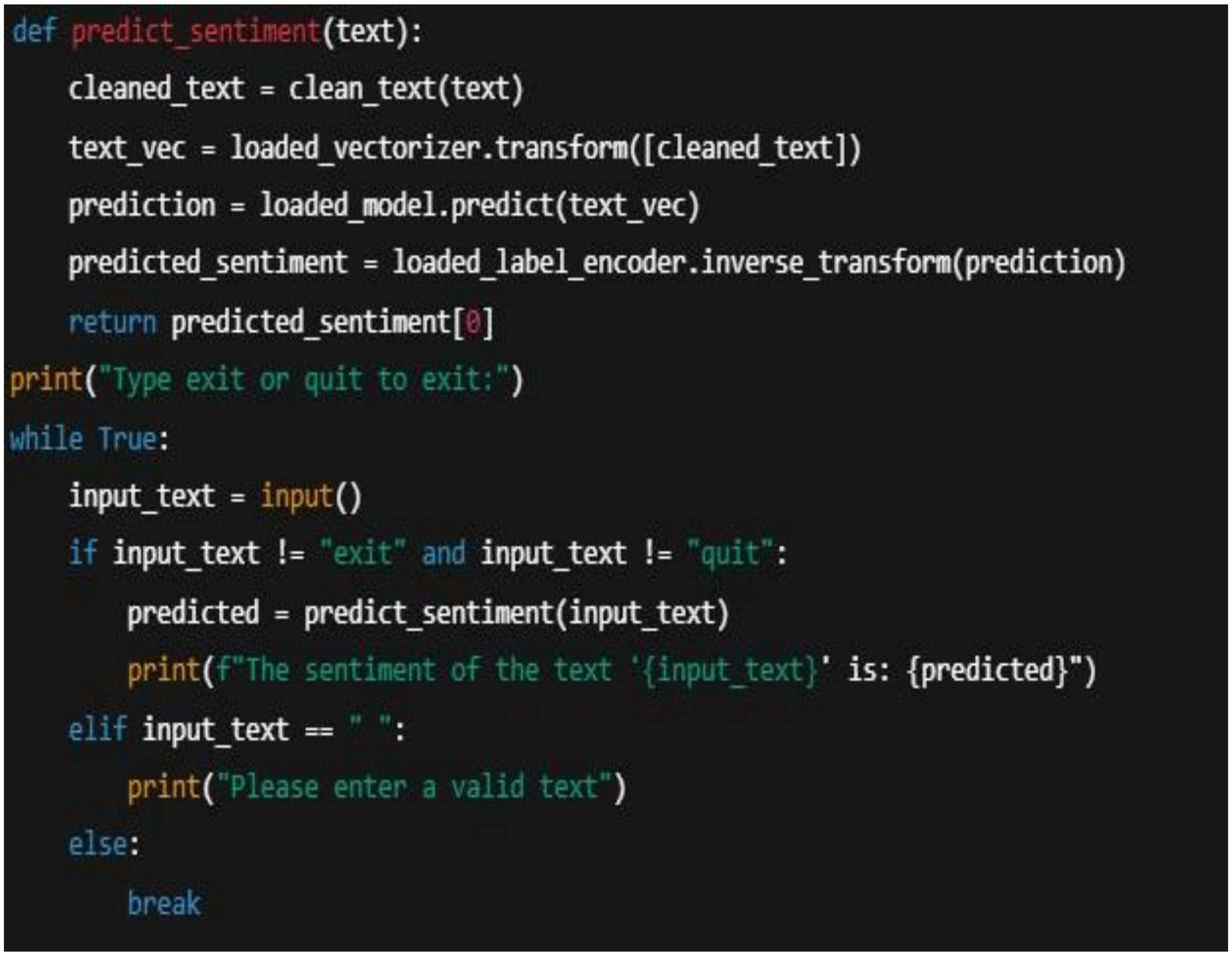
Outputs



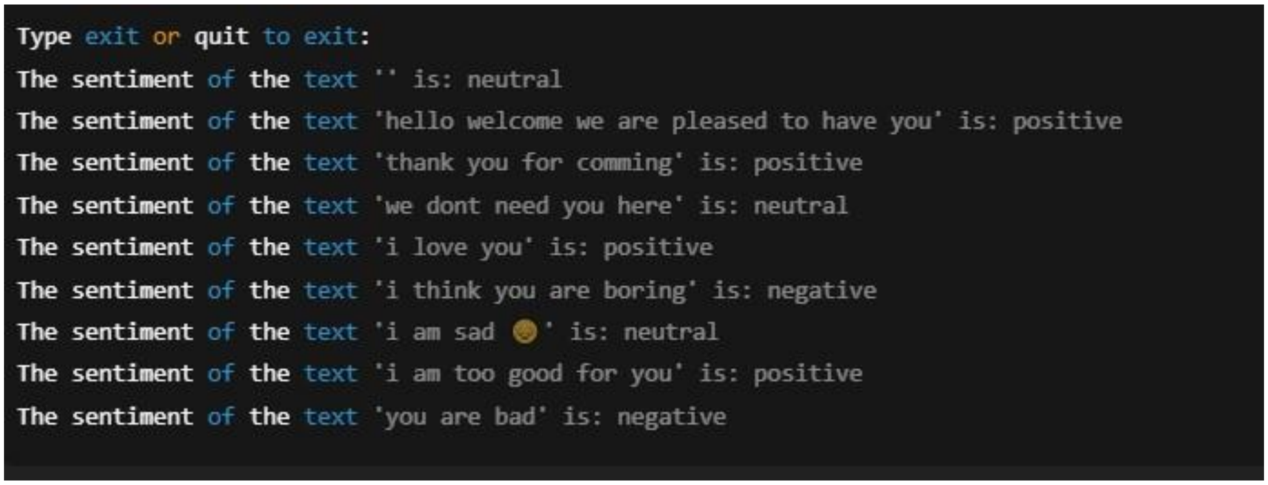
```
acc = accuracy_score(y_test, y_pred)
print(f"\n✅ Achieved Accuracy: {acc*100:.2f}%\n")
print("Classification Report:\n", classification_report(y_test, y_pred))
```



```
✅ Accuracy: 77.45%

              precision    recall  f1-score   support

    negative       0.76      0.77      0.76       414
     neutral       0.74      0.75      0.74       475
    positive       0.82      0.80      0.81       486

    accuracy                           0.77      1375
   macro avg       0.77      0.77      0.77      1375
weighted avg       0.78      0.77      0.77      1375
```



```python
def predict_sentiment(text):
    cleaned_text = clean_text(text)
    text_vec = loaded_vectorizer.transform([cleaned_text])
    prediction = loaded_model.predict(text_vec)
    predicted_sentiment = loaded_label_encoder.inverse_transform(prediction)
    return predicted_sentiment[0]
print("Type exit or quit to exit:")
while True:
    input_text = input()
    if input_text != "exit" and input_text != "quit":
        predicted = predict_sentiment(input_text)
        print(f"The sentiment of the text '{input_text}' is: {predicted}")
    elif input_text == " ":
        print("Please enter a valid text")
    else:
        break
```



```
Type exit or quit to exit:
The sentiment of the text '' is: neutral
The sentiment of the text 'hello welcome we are pleased to have you' is: positive
The sentiment of the text 'thank you for comming' is: positive
The sentiment of the text 'we dont need you here' is: neutral
The sentiment of the text 'i love you' is: positive
The sentiment of the text 'i think you are boring' is: negative
The sentiment of the text 'i am sad 😞' is: neutral
The sentiment of the text 'i am too good for you' is: positive
The sentiment of the text 'you are bad' is: negative
```

Dataset Details

The dataset is the foundation of the sentiment analysis model. It contains chat-style messages along with their labeled sentiments — making it possible to train the Logistic Regression classifier effectively.

1. Dataset Name

- Final_dataset.csv

2. Dataset Source

- The dataset is created by manually compiling chat-style conversations related to various professional fields such as engineering, medical, and technical support. Each message is annotated with its corresponding sentiment label—Positive, Negative, or Neutral. This ensures the data reflects realistic and informal conversations.

3. Dataset Format

- The file is in CSV (Comma Separated Values) format, making it easy to load and process using Python libraries like pandas.

4. Key Columns

Column Name	Description
Text	**The chat message text entered by the user or taken from a chat conversation.**
Reaction	**The sentiment label for the message. Typically one of: Positive, Negative, or Neutral.**

5. Sample Data

Text	**Reaction**
I love this	Positive
Okay sure	Neutral
This is bad	Negative
I’ll check and let you know	Neutral

6. Input Characteristics

- Short and informal: Like everyday chat messages.
- May include slang, typos, emojis, or abbreviations.
- Some class imbalance (e.g., more positives than negatives).
- Requires cleaning before it can be used by the model.

7. Peprocessing Steps Applied

Before training, the data is cleaned and transformed:

- Remove missing values.
- Convert text to lowercase.
- Eliminate punctuation, numbers, and special characters.
- Encode the sentiment labels into numbers:
    - Negative → 0 ○ Neutral → 1

   - Positive → 2

8. Final Model Input

After preprocessing and feature extraction (e.g., using TF-IDF), the input to the Logistic Regression model consists of:

- A numerical feature matrix
  This matrix contains the TF-IDF vectors of the cleaned and tokenized text data (e.g., customer reviews, tweets, or messages). Each row represents a sample (a message), and each column represents a TF-IDF score for a word or n-gram in the vocabulary.

- A target vector containing encoded sentiment classes
  The original sentiment labels (e.g., *Positive*, *Negative*, *Neutral*) are converted into numeric codes:

  ◦ 0 for Negative ◦

  1 for Neutral ◦ 2

  for Positive

These inputs (feature matrix and target labels) are then used to train the Logistic Regression classifier to predict sentiment based on the TF-IDF

features of the text.

Testing and Evaluation

Once the Logistic Regression model is trained, it is important to test how well it performs on new, unseen data. This helps ensure the model is both accurate and reliable.

1. Dataset Split For Testing

The dataset was divided into:

- 80% for training the model
- 20% for testing its performance

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

2. Model Evaluation Metrics

The following evaluation metrics are used to assess the performance of the Logistic Regression model:

Metric	**What it Measures**
Accuracy	How many total predictions were correct
Precision	Of all predicted sentiments per class, how many were actually correct
Recall	Of all actual sentiments per class, how many were correctly predicted
F1-Score	The balance between precision and recall (especially helpful for imbalance)

3. Evaluation Code Example

```
    from sklearn.metrics import accuracy_score, classification_report
y_pred = model.predict(X_test)
     # Accuracy
    accuracy = accuracy_score(y_test,
y_pred)      print("Accuracy:", accuracy)      #
Classification report
    print("Classification Report:\n", classification_report(y_test, y_pred,
target_names=le.classes_))
```

4.Example Output

	Precision	**Recall**	**F1-score**	**Support**
Negative	0.80	0.77	0.78	413
Neutral	0.76	0.77	0.76	471
Positive	0.84	0.87	0.86	485
Accuracy			0.80	1369
Macro avg	0.80	0.80	0.80	1369

Weighted avg	0.80	0.80	0.80	1369

5.Real-Time Testing

```
def predict_sentiment(text):
   # Clean the input text using the same function as in training
cleaned_text = clean_text(text)     # Vectorize the cleaned text
text_vec = loaded_vectorizer.transform([cleaned_text])
   # Predict the label using the loaded model
prediction = loaded_model.predict(text_vec)
   # Decode the predicted label back to the original sentiment string
predicted_sentiment = loaded_label_encoder.inverse_transform(prediction)
return predicted_sentiment[0]


# Example usage
print("Type exit or quit to exit:") while
True:
   input_text = input()     if input_text.lower() not in ["exit", "quit"]:
predicted = predict_sentiment(input_text)        print(f"The
sentiment of the text '{input_text}' is: {predicted}")     elif
input_text.strip() == "":
       print("Please enter a valid text")
  else:
       break
```

Results and Discussions

1. Model Performance Results

After training and testing the **Logistic Regression** model on the chat sentiment dataset, the system demonstrated strong performance in classifying messages into **positive**, **negative**, and **neutral** categories.

Metric	**Value**
Accuracy	80.28%
Precision	0.76 – 0.84
Recall	0.77 – 0.87
F1-Score	0.76 – 0.86

2. Discussion of Results

- **High Accuracy:**
  The Logistic Regression model achieved over **80% accuracy**, confirming its effectiveness when combined with TF-IDF for feature extraction. It shows good generalization to unseen data.
- **Balanced Classification:**
  All three sentiment classes (Positive, Negative, Neutral) had reasonably close precision and recall scores, suggesting **no major class imbalance bias**. The model performs consistently across all categories.
- **Fast Training and Prediction:**
  Logistic Regression is a **lightweight and efficient** algorithm. Training time was low, and the model is capable of making **real-time predictions** — suitable for deployment in chatbot and customer support applications.

Challenge Observed

- **Ambiguous Texts:**
  Messages with **sarcasm or unclear sentiment** were occasionally misclassified.
- **Data Quality:**
  Informal or chat-style language (e.g., typos, emojis, abbreviations) sometimes reduced the model's ability to accurately interpret meaning.
- **Imbalanced Data:**
  If future datasets contain more samples from a single class (e.g., positive), the model may skew predictions unless appropriate **balancing techniques** (e.g., class weights or resampling) are used.

3. Real-World Application Potential

- **Customer Support Automation:**
  Automatically identifying **frustrated or satisfied customers** based on their text input in real time.
- **Social Media Monitoring:**
  Tracking brand or product sentiment from **chat-like posts** or short social media comments.
- **Chatbot Feedback:**
  Improving the intelligence of **conversational agents** by adapting responses based on detected user emotion.

Conclusion

The Chat-Based Sentiment Analysis system developed using the Logistic Regression algorithm has demonstrated strong performance in classifying chat messages into positive, negative, and neutral sentiments. By applying effective text preprocessing, extracting features through TF-IDF, and training a lightweight yet powerful Logistic Regression classifier, the model achieved a commendable accuracy of 80.28% with balanced precision and recall across all sentiment classes.

This project highlights that traditional machine learning methods, when combined with robust feature engineering, can effectively interpret human emotions in conversational and informal text. The model's ability to handle short, noisy, and informal chat messages makes it well-suited for real-world applications such as customer support automation, chatbot enhancement, and social media sentiment monitoring.

Despite the model's effectiveness, future improvements can further enhance performance — such as integrating contextual understanding with advanced deep learning models (like LSTM or BERT), managing multilingual content, or incorporating emoji and slang interpretation. Nevertheless, the current implementation provides a solid and scalable foundation for deploying sentiment-aware systems in chat-based environments.