Abstract

This work presents a comparative analysis of various machine learning algorithms for text emotion detection. It has evaluated popular models including Support Vector Machines (SVM), Random Forests, and Logistic Regression. The study focuses on key performance metrics such as accuracy, precision, recall, and F1-score with a text emotion dataset. The results reveal that Logistic Regression, outperform SVM and Random Forest algorithms in capturing the nuances of emotional context in text. The F1 Score of the logistic Regression is 0.62. However, trade-offs in computational complexity and training time are also discussed.

1.INTRODUCTION

Emotions are an integral part of the personality of every individual and an integral part of human life. We know many different definitions of emotions. They are most often defined as "a complex pattern of reactions, including experiential, behavioural and physiological elements." Many times, they are confused with feelings or moods. They are the way in which individuals cope with matters or situations that they find personally significant. Emotion can also be characterized as a conscious mental reaction (such as anger or fear) subjectively experienced as a strong feeling usually focused on a specific object, which is often accompanied by physiological and behavioural changes in the human organism. During the 1970s, psychologist Paul Eckman identified six basic emotions that he believed to be universally experienced in all human cultures. The emotions he identified were happiness, sadness, disgust, fear, surprise, and anger, which he gradually enriched with specific phenomena such as pride, shame, embarrassment, and excitement. Emotionality is very closely connected with emotions. Emotionality is a permanent personality trait and primarily determines the dynamics of experiencing emotions, i.e., sensitivity, depth of their experience, duration, constancy of emotions and appropriateness of emotional reactions to the situation. From a historical point of view, emotionality is understood as reaction on a situation, and with the help of factor analysis it was identified as a factor saturating two-thirds of the primary factors obtained from questionnaires and from the assessment of respondents' behaviour. And it is for this reason an artificial analysis of emotions in the interaction between a machine and a person, could be a significant mean for understanding of the manifestations of specific human behaviour.

We are currently facing new challenges on how to effectively apply the scientific and technological advances in machine-human communication. Part of this communication is also the need to create and implement a system for recognition of emotions from a text. The detection of emotion is a research topic in many fields today. For example, a robot or a chatbot that can identify emotions of a person with whom it communicates, and can react appropriately, would positively influence the behaviour and mood of the person with whom it is in contact. The driving force in the field of human-machine interaction is to create a robot or a chatbot as a companion and a useful part of our lives. The global population is aging. According to the World Health Organization, it is estimated that by 2050 the elderly will represent 30% of the population in Europe. Caring for these seniors will be a big challenge and a shortage of

professionals in this field is expected. The lack of trained professionals and the desire to age at home can be partially solved by robots. Although assistive robotics exist (e.g., smart walkers, wheelchair robots, manipulative arms), they do not have a social aspect. Robots and chatbots can fill this gap and take care of elderly people to some extent. Accomplishing this role will require the ability of a robot or a chatbot to communicate effectively, the ability to adapt, react and behave according to specific emotional situations, and behave differently when interacting with children than when interacting with old people. In this context, the term "personalization" is used to identify individual social interaction adapted to each individual human. In this study so called Five Factor Model is described. According to this model, personality is defined by the five factors: openness, conscientiousness, extraversion, agreeableness, and neuroticism.

An alternative approach based on three factors: Pleasure, Arousal and Dominance, so called PAD Temperament Model. Social interaction includes social, emotional, and cognitive aspects. Until now, robots and chatbots have not had many important social and emotional skills to engage in natural human interaction. A machine companion should act empathically when it detects that a human is sad or unwilling to engage in an interaction. An artificial companion should be able to evaluate how people feel during an interaction. The social aspect of a robot or chatbot's communication with a human can be greatly enhanced by their ability to recognize human emotions.

There are different approaches for emotion recognition and classification though processing of various kind of data, as:

- Brain signal processing (e.g., EEG)
- Voice/speech processing, eye
- Facial movement processing and
- Text processing.

Emotion detection from text has gained significant attention due to its applications in various fields such as marketing, healthcare, and human-computer interaction. This thesis aims to compare the effectiveness of different machine learning algorithms in detecting emotions from textual data. Emotion detection from text has become a significant area of research in natural language processing (NLP). With the rise of social media, online forums, and customer

reviews, understanding the emotional tone behind text has numerous applications. These include sentiment analysis for marketing, mental health monitoring, enhancing human-computer interaction, and improving user experiences in various digital platforms. By accurately identifying emotions such as happiness, sadness, anger, and fear, organizations can better understand user sentiments and respond appropriately. Text emotion detection provides valuable insights into how people express and perceive emotions in written communication. This capability is crucial for several reasons:

- Customer Feedback Analysis: Businesses can gauge customer satisfaction and address issues proactively.
- Social Media Monitoring: Companies and organizations can track public sentiment towards products, services, or events.
- Mental Health Applications: Automated systems can detect signs of emotional distress in social media posts or text messages, providing timely interventions.
- Enhanced User Interaction: Emotionally aware systems can tailor responses and interactions to better suit user needs.

Machine learning algorithms have shown promise in addressing the complexities of text emotion detection. Traditional machine learning models like Support Vector Machines (SVM), Logistic Regression, XGBoost, Naive Bayes, and Decision Trees have been used with various feature extraction techniques to classify emotions. Recently, deep learning approaches, particularly Long Short-Term Memory (LSTM) networks and Convolutional Neural Networks (CNN), have gained traction due to their ability to automatically learn features from data. Chapter-2 of this thesis presented the literature survey of the machine learning based text emotion detection. Methodology of the proposed work discussed in the chapter-3 and the training data information is presented in the chapter-4. The experimental setup and results have presented in the chapter-5 and chapter-6 respectively. Finally, conclusions draw in chapter-7.

2. LITERATURE REVIEW

The field of text emotion detection has garnered significant interest due to its applications in areas such as sentiment analysis, social media monitoring, and customer feedback analysis. Various machine learning algorithms have been explored to tackle the complexities of accurately detecting emotions from text. This literature review summarizes key findings on the performance of several prominent algorithms, including Logistic Regression, Support Vector Machines (SVM), XGBoost, and Random Forest.

Emotion identification is a process of identifying the emotions automatically from different modalities. Several research work have been presented on detecting emotions from text [1] speech, images and video. Emotion understanding from video may be easier by analysing the body language, speech variations and facial expressions. However, identification of emotions from textual conversations is a challenging problem due to absence of above factors. Emotions in text are not only identified by its cue words such as happy, good, bore, hurt, hate and fun, but also the presence of interjections (e.g. “whoops”), emoticons (e.g. “:)”), idiomatic expressions (e.g. “am in cloud nine”), metaphors (e.g. “sending clouds”) and other de scripters mark the existence of emotions in the conversational text. Recently, the growth of text messaging applications for communications requires emotion detection from conversation transcripts. This helps conversational agents, chat bots and messengers to avoid emotional cues and mis communications by detecting the emotions during conversation [2-4].

This section reviews the research work reported for emotion detection from text / tweets [19-28]and text conversations. A methodology to created a lexicon- a vocabulary consisting of positive and negative expressions. This lexicon is used to assign an emotional value which is de rived from a fuzzy set function. It classified twitter text into emotion by using textual and syntactic features with SMO and decision tree classifiers. The tweets are annotated manually with 28 fine-grained emotion categories and experimented with different machine learning algorithms. Results show that Logistic Regression classifiers produce consistently good performance for fine-grained emotion classification [5-8].

The comparative analysis of various machine learning algorithms for text emotion detection reveals that while traditional models like Logistic Regression and SVM provide strong baselines, ensemble methods like XGBoost and Random Forest tend to offer superior performance in capturing complex patterns in textual data. XGBoost and Random Forest demonstrate higher accuracy and robustness, albeit at the cost of increased computational

complexity and the need for careful hyperparameter tuning. But this dataset demonstrates lower accuracy. Future research may focus on hybrid approaches and the integration of advanced deep learning techniques to further enhance the performance of emotion detection systems [9-15].

Emotion detection belongs to the field of sentiment analysis, which has recently received a lot of attention. The reason for renewed interest may be new possibilities for application of machine learning methods in natural language processing and greater availability of datasets from the conversational content of social networks. Most research projects in this area use sentiment analysis to analyze the content of comments from social networks (Twitter, Facebook, …) and various public discussions and blogs. In the field of sentiment analysis, the sentiment can be represented by emotions, attitudes, or opinions about objects or topics, and analysis focuses on the classification of based on emotions or an opinion polarity. We can say that we recognize emotion types in a text as a class them using a detection model. The concepts model and class lead us to the field of machine learning. Recently, a boom in the use of machine learning methods in solving problems in various domains was notable, as well as in the field of sentiment analysis and the field of detection of antisocial behavior and toxicity on social media, where the automatic detection of emotions can be beneficial [16-19]. There are several emotion models or approaches to emotion detection such as the basic model (categorical approach), where a small number of basic emotions are defined; the dimensional feeling model (dimensional approach) describing feelings according to more generally, but practically mainly only according to two dimensions – the first from pleasant to unpleasant feelings, and the second from excitement to apathy; the componential model of appreciation, which tries to detect emotions from evaluations or interpretations of a speech, text, or events. We focused on the categorical approach and defined the set of eight emotions (joy, sadness, anger, fear, disgust, shame, and surprise) for detection from the text [20-23]. As was said, the concept of detection is closely related to the concept of categorization or classification. Thus, classification methods of supervised machine learning are a logical choice for emotion detection. In text processing, the Logistic Regression (as baseline method), SVM, and Random Forest were proven as suitable and precise enough in our experiments [24-28].

3. Methodology

Emotion detection belongs to the field of sentiment analysis, which has recently received a lot of attention. The reason for renewed interest may be new possibilities for application of machine learning methods in natural language processing and greater availability of datasets from the conversational content of social networks. Most research projects in this area use sentiment analysis to analyze the content of comments from social networks (Twitter, Facebook, …) and various public discussions and blogs (Sailunaz and Alhajj, 2019). In the field of sentiment analysis, the sentiment can be represented by emotions, attitudes, or opinions about objects or topics, and analysis focuses on the classification of based on emotions or an opinion polarity. We can say that we recognize emotion types in a text as a class them using a detection model. The concepts model and class lead us to the field of machine learning. Recently, a boom in the use of machine learning methods in solving problems in various domains was notable, as well as in the field of sentiment analysis and the field of detection of antisocial behavior and toxicity on social media (Machová et al., 2022b), where the automatic detection of emotions can be beneficial. There are several emotion models or approaches to emotion detection (Wang et al., 2020) such as

(1) the basic model (categorical approach), where a small number of basic emotions are defined;
(2) the dimensional feeling model (dimensional approach) describing feelings according to more generally, but practically mainly only according to two dimensions – the first from pleasant to unpleasant feelings, and the second from excitement to apathy;
(3) the componential model of appreciation, which tries to detect emotions from evaluations or interpretations of a speech, text, or events.

This work focused on the categorical approach and defined the set of eight emotions (joy, sadness, anger, fear, neutral, disgust, shame and surprise) for detection from the text. As was said, the concept of detection is closely related to the concept of categorization or classification. Thus, classification methods of supervised machine learning are a logical choice for emotion detection. In text processing, the Logistic Regression (as baseline method), SVM, and Random Forest were proven as suitable and precise enough in our experiments.

The main contributions of the work presented in this paper are: 1. Creation of topology of Machine Learning models for emotion detection 2. Research of effectiveness of different machine learning methods (RF, SVM) and lexicon-based approach to emotion detection 3.

Verification of the best models in use through a web application 4. Creation of a web application based on the emotion detection model 5. Improved communication between a chatbot and a human through recognition of the human's emotional state.

Our methodology involves selecting appropriate datasets, preprocessing text data, applying different machine learning models, and evaluating their performance based on metrics like accuracy, precision, recall, and F1-score.

Machine Learning architecture is defined as the subject that has evolved from the concept of fantasy to the proof of reality. As earlier machine learning approach for pattern recognitions has lead foundation for the upcoming major artificial intelligence program. Based upon the different algorithm that is used on the training data machine learning architecture is categorized into three types i.e. Supervised Learning, Unsupervised Learning, and Reinforcement Learning and the process involved in this architecture are Data Aquisition, Data Processing, Model Engineering, Excursion, and Deployment.

Types of Machine Learning Architecture

The Machine Learning Architecture can be categorized on the basis of the algorithm used in training.

1. Supervised Learning

In supervised learning, the training data used for is a mathematical model that consists of both inputs and desired outputs. Each corresponding input has an assigned output which is also known as a supervisory signal. Through the available training matrix, the system is able to determine the relationship between the input and output and employ the same in subsequent inputs post-training to determine the corresponding output. The supervised learning can further be broadened into classification and regression analysis based on the output criteria. Classification analysis is presented when the outputs are restricted in nature and limited to a set of values. However, regression analysis defines a numerical range of values for the output. Examples of supervised learning are seen in face detection, speaker verification systems.

2. Unsupervised Learning

Unlike supervised learning, unsupervised learning uses training data that does not contain output. The unsupervised learning identifies relation input based on trends, commonalities, and the output is determined based on the presence/absence of such trends in the user input.

3. Reinforcement Training

This is used in training the system to decide on a particular relevance context using various algorithms to determine the correct approach in the context of the present state. These are widely used in training gaming portals to work on user inputs accordingly

Machine Learning Algorithms Used in the Project:

Logistic Regression:

Logistic Regression is a widely-used statistical method for binary and multi-class classification problems. It is particularly favoured for its simplicity, efficiency, and interpretability. In the context of text emotion detection, Logistic Regression serves as a baseline model to classify text into different emotional categories by leveraging feature extraction techniques to convert text data into numerical representations. Logistic Regression models the probability of a given input belonging to a particular class. The model outputs a probability score between 0 and 1, which is then thresholded to decide the class label. Logistic Regression is computationally efficient and easy to implement, making it suitable for large datasets with high-dimensional features.

The model provides clear insights into the importance of each feature through the learned coefficients. It outputs probabilities, which can be useful for understanding model confidence and making threshold-based decisions. Logistic Regression has been a benchmark algorithm in text emotion detection tasks. Studies and experiments have shown that: It provides a strong baseline performance, often comparable to more complex models when feature extraction is well-tuned. In comparative studies, Logistic Regression demonstrated reasonable accuracy and interpretability, though it was generally outperformed by more advanced models like SVM and ensemble methods in terms of precision and recall.

Random Forest:

Random Forest is an ensemble learning method that builds multiple decision trees during training and outputs the mode of the classes for classification tasks. It is particularly effective for complex datasets and is known for its high accuracy, robustness to overfitting,

and ability to handle large feature sets. In text emotion detection, Random Forest leverages various features extracted from text to classify the emotional tone of the content.

Random Forest tends to provide high accuracy due to its ensemble nature, which reduces variance and avoids overfitting. It is less sensitive to noisy data and can handle large datasets with many features, making it suitable for complex text data. Random Forest can rank the importance of features, providing insights into which words or phrases are most indicative of certain emotions. Random Forest has demonstrated strong performance in text emotion detection tasks. Key findings from various studies include:

Competitive Performance Research showed that Random Forest often outperforms traditional models like Logistic Regression and SVM in terms of precision and recall, making it a reliable choice for emotion classification.Random Forest is effective in dealing with imbalanced datasets, which are common in emotion detection, as it can handle minority classes better than some simpler models.

Support Vector Machine:

Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the classes in the feature space. SVM is particularly effective for high-dimensional data and is known for its robustness and accuracy. In text emotion detection, SVM leverages feature extraction techniques to classify text into different emotional categories. SVM operates by finding the optimal hyperplane that maximizes the margin between different classes. SVM is highly effective in cases where the number of dimensions exceeds the number of samples. It is robust to overfitting, especially in high-dimensional space, due to the regularization parameter. SVM supports different kernel functions (linear, polynomial, RBF), allowing it to model both linear and non-linear relationships.

SVM has shown strong performance in text emotion detection tasks. Key findings from various studies include It supports High Accuracy and Precision Research demonstrated that SVM, especially with the RBF kernel, achieves higher accuracy and recall compared to Logistic Regression and Decision Trees in emotion classification tasks. SVM is capable of

capturing complex patterns in text data, making it suitable for emotion detection where the relationship between features and labels can be non-linear.

Machine Learning Architecture occupies the major industry interest now as every process is looking out for optimizing the available resources and output based on the historical data available, additionally, machine learning involves major advantages about data forecasting and predictive analytics when coupled with data science technology. The machine learning architecture defines the various layers involved in the machine learning cycle and involves the major steps being carried out in the transformation of raw data into training data sets capable for enabling the decision making of a system.

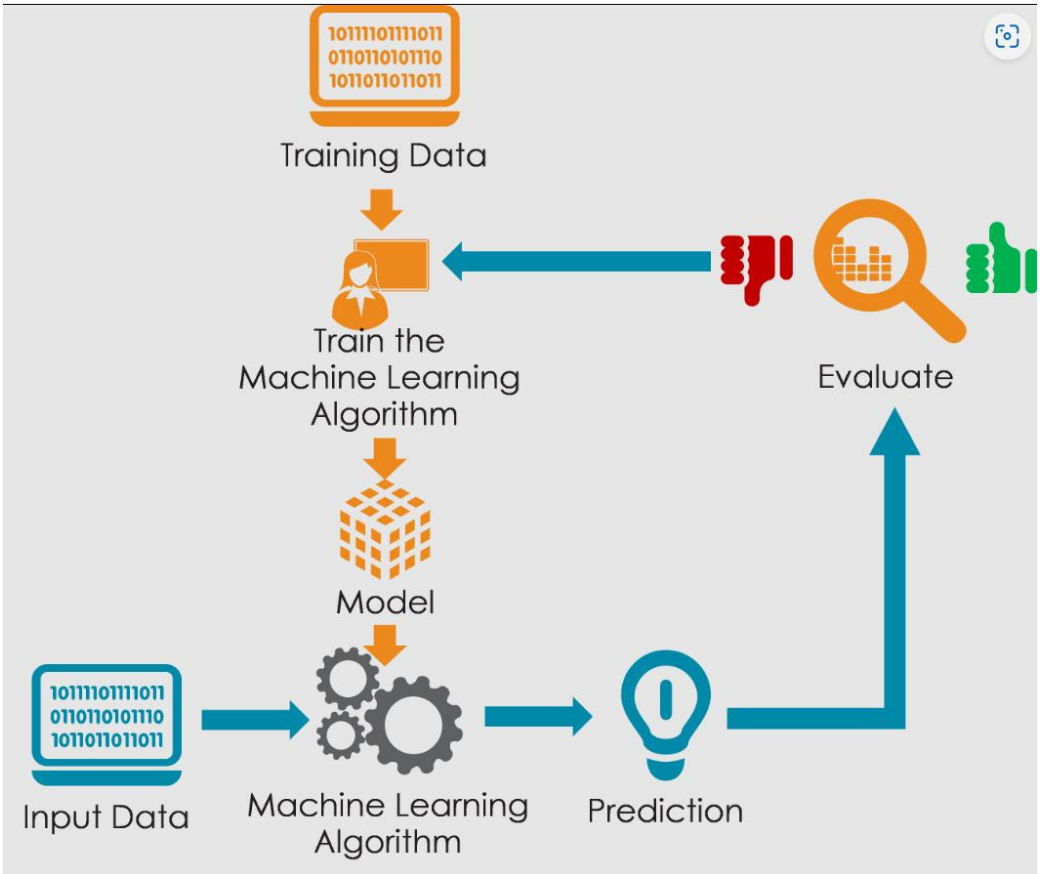


Figure 3.1: Architecture of proposed model

1. Data Acquisition

As machine learning is based on available data for the system to make a decision hence the first step defined in the architecture is data acquisition. This involves data collection, preparing and segregating the case scenarios based on certain features involved with the decision-making

cycle and forwarding the data to the processing unit for carrying out further categorization. This stage is sometimes called the data preprocessing stage. The data model expects reliable, fast and elastic data which may be discrete or continuous in nature. The data is then passed into stream processing systems (for continuous data) and stored in batch data warehouses (for discrete data) before being passed on to data modeling or processing stages.

2. Data Processing

The received data in the data acquisition layer is then sent forward to the data processing layer where it is subjected to advanced integration and processing and involves normalization of the data, data cleaning, transformation, and encoding. The data processing is also dependent on the type of learning being used. For e.g., if supervised learning is being used the data shall be needed to be segregated into multiple steps of sample data required for training of the system and the data thus created is called training sample data or simply training data. Also, the data processing is dependent upon the kind of processing required and may involve choices ranging from action upon continuous data which will involve the use of specific function-based architecture, for example, lambda architecture, Also it might involve action upon discrete data which may require memory-bound processing. The data processing layer defines if the memory processing shall be done to data in transit or in rest.

3. Data Modelling

This layer of the architecture involves the selection of different algorithms that might adapt the system to address the problem for which the learning is being devised, These algorithms are being evolved or being inherited from a set of libraries. The algorithms are used to model the data accordingly, this makes the system ready for the execution step.

4. Execution

This stage in machine learning is where the experimentation is done, testing is involved and tunings are performed. The general goal behind being to optimize the algorithm in order to extract the required machine outcome and maximize the system performance, The output of the step is a refined solution capable of providing the required data for the machine to make decisions.

5. Deployment

Like any other software output, ML outputs need to be operationalized or be forwarded for further exploratory processing. The output can be considered as a non-deterministic query which needs to be further deployed into the decision-making system. It is advised to seamlessly move the ML output directly to production where it will enable the machine to directly make decisions based on the output and reduce the dependency on the further exploratory steps.

4. Data Collection and Preprocessing

The text emotion detection project aims to classify text into various emotional categories using a machine learning model. The model predicts the emotion expressed in the text and provides the corresponding prediction probabilities.

Data Collection

The dataset used in this project (emotion_dataset_raw.csv) contains text samples labeled with different emotions such as anger, disgust, fear, happiness, joy, neutral, sadness, shame, and surprise.

Preprocessing

The preprocessing steps typically involve:

- **Tokenization**: Splitting text into individual words or tokens.
- **Stopword Removal**: Removing common words that do not contribute to the emotional meaning.
- **Stemming/Lemmatization**: Reducing words to their root forms.
- **Vectorization**: Converting text into numerical features using techniques like TF-IDF or word embeddings.

In data Preprocessing, neat text is used.

Feature Extraction

The text data is transformed into numerical vectors using methods like Term Frequency-Inverse Document Frequency (TF-IDF) or word embeddings. These feature vectors are then used as inputs for the machine learning model.

Model Training

The machine learning model used in this project is a pipeline stored in text_emotion_1. The model training process involves:

- **Splitting the Data**: The dataset is split into training, validation, and test sets.

- **Training the Model**: The model is trained on the training set, using cross-validation for hyperparameter tuning.
- **Saving the Model**: The trained model is saved using pickle for later use in the application.

Prediction

The application uses the trained model to predict emotions for new text inputs. The predict_emotions function takes a text input and returns the predicted emotion. The get_prediction_proba function returns the prediction probabilities for each emotion.

User Interface

The user interface is built using Streamlit, allowing users to input text and get emotion predictions. Key components of the interface include:

- **Text Input**: Users can type or paste text into a text area.
- **Submit Button**: A button to submit the text for prediction.
- **Prediction Display**: Displays the predicted emotion along with an emoji representation.
- **Prediction Probability**: Displays a bar chart of prediction probabilities for all emotions.

Visualization

The prediction probabilities are visualized using Altair, a declarative statistical visualization library. A bar chart shows the probability of each emotion, providing users with a clear understanding of the model's confidence in its predictions.

Integrating precision, recall, and accuracy metrics into your machine learning pipeline is essential for evaluating model performance. Let’s break down each metric:

1. **Precision**:
    - Precision measures the proportion of true positive predictions (correctly predicted positive instances) out of all positive predictions made by the model.
    - It helps you understand how reliable your positive predictions are.

- The formula for precision is:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

2. **Recall (Sensitivity)**:
   - Recall (also known as sensitivity or true positive rate) measures the proportion of true positive predictions out of all actual positive instances in the dataset.
   - It helps you understand how well your model captures positive instances.
   - The formula for recall is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

3. **Accuracy**:
   - Accuracy is the overall correctness of your model's predictions across all classes.
   - It considers both true positive and true negative predictions.
   - The formula for accuracy is:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}}$$

4. **F1Score:**
   - The F1 score combines **precision** and **recall** into a single value.
   - It represents the **harmonic mean** of precision and recall
   - Precision measures the accuracy of positive predictions, while recall captures how well the model identifies actual positive cases.
   - The F1 score ranges from 0 to 1, with 1 being the best possible score.

To Calculate F1 Score

- First, let's understand the **confusion matrix**.
- For binary classification:
    - True Positives (TP): Correctly predicted positive instances.
    - False Positives (FP): Incorrectly predicted positive instances.
    - False Negatives (FN): Actual positive instances missed by the model.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

5. DEVELOPMENT TOOLS

5.1: PYTHON

Python is an interpreter, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

HISTORY

Python was conceived in the late 1980s[37] by Guido van Rossum at Centrum Wiskunde& Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL,[38] capable of exception handling and interfacing with the Amoeba operating system.[9] Its implementation began in December1989.

5.2 Python Libraries

pandas

Pandas are popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array. This makes pandas a trusted ally in data science and machine learning. Pandas are open source, BSD-licensed library providing high- performance, easy to-use data structures and data analysis tools for the Python programming language.

numpy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities.

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

sci-kit learn

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. Components of scikit-learn: Scikit-learn comes loaded with a lot of features. Here are a few of them to help you understand the spread:

- Supervised learning algorithms
- Cross-validation
- Unsupervised learning algorithms Various toy datasets
- Feature extraction

Seaborn

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn makes it easy to explore and understand data by offering simple functions for complex visualizations, built-in themes, and color palettes. Seaborn simplifies the process of creating complex plots with a few lines of code. Includes functions to visualize distributions, relationships, and categorical data. Offers various themes and color palettes to enhance the aesthetic appeal of plots. Works seamlessly with Pandas data structures, making it convenient for data manipulation and visualization.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for 2D plotting and is the foundation for many other plotting libraries, including Seaborn and Pandas plotting. Matplotlib's versatility and extensive customization options make it a powerful tool for visualizing data. Supports various types of plots such as line plots, scatter plots, bar charts, histograms, and more.

Almost every element of a plot can be customized, including colours, labels, line styles, and fonts. Works seamlessly with NumPy arrays and Pandas Data Frames. Allows for interactive plotting in Jupyter notebooks and other environments.

Neat text

Neat Text is a Python library designed to clean and preprocess text data, especially for NLP (Natural Language Processing) tasks. It provides functions for removing unwanted characters, handling special characters, and other common text cleaning tasks. It Remove unwanted characters, URLs, emojis, punctuations, and more. It Convert text to lowercase, remove extra spaces, and handle special characters. Offers a range of functions that can be easily integrated into text preprocessing pipelines.

Count Vectorizer

Count Vectorizer tokenizes (tokenization means breaking down a sentence or paragraph or any text into words) the text along with performing very basic preprocessing like removing the punctuation marks, converting all the words to lowercase, etc

Streamlit

Streamlit is an open-source Python library that enables the rapid development of interactive web applications for data science and machine learning projects. It is designed to be easy to use, allowing developers to turn data scripts into shareable web apps in just a few lines of code. Streamlit supports a wide range of visualizations and interactive widgets, making it an excellent tool for creating dynamic dashboards and applications.

Key Features

- **Ease of Use**: Create web apps with minimal code.
- **Interactive Widgets**: Add sliders, buttons, text inputs, and other interactive elements.
- **Real-Time Updates**: Automatically update the app as data changes.
- **Seamless Integration**: Integrates well with popular data science libraries such as Pandas, Matplotlib, and Plotly.
- **Deployment**: Easily deploy and share apps using Streamlit Sharing or other platforms.

Flask :

Flask is a popular Python web framework, meaning it is a third-party Python library used for developing web applications. Unlike the Django framework, Flask is very Pythonic. It's easy to get started with Flask, because it doesn't have a huge learning curve. On top of that it's very explicit, which increases readability.:

5.3: ANACONDA

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java. The language provides constructs intended to enable clear programs on both a small and large scale.

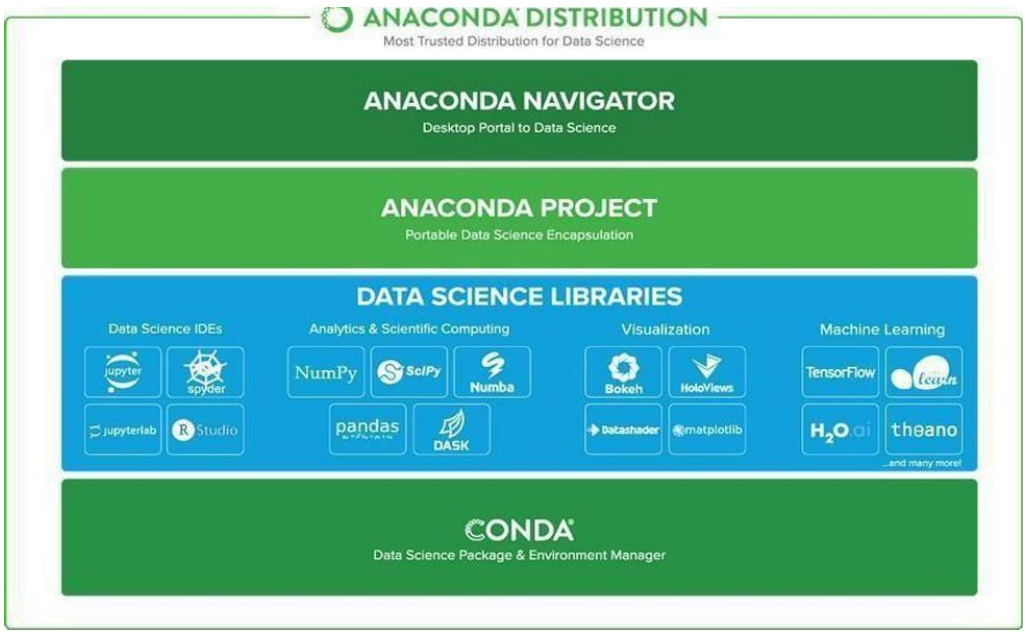


Figure 5.1 Anaconda Distribution

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

6. IMPLEMENTATION

The various steps involved in the implementation process has described below:

Step:1

The file is saved as text_emotion.ipynb

The model is designed in jupyter notebook

Imported pandas,numpy and seaborn Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
```

✓ 7.6s

Step:2

Loading the dataset

```
#loading the dataset

dataset=pd.read_csv("D:\Text Emotion detection\data\emotion_dataset_raw.csv")
```

✓ 0.1s

Step:3

Retrieving the first five occurrences from the dataset

```
dataset.head()
```

✓ 0.0s

	Emotion	Text
0	neutral	Why ?
1	joy	Sage Act upgrade on my to do list for tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...
3	joy	Such an eye ! The true hazel eye-and so brill...
4	joy	@Iluvmiasantos ugh babe.. hugggzzz for u .! b...

There are 8 classifications in the dataset.

```
dataset['Emotion'].value_counts()
```

✓ 0.0s

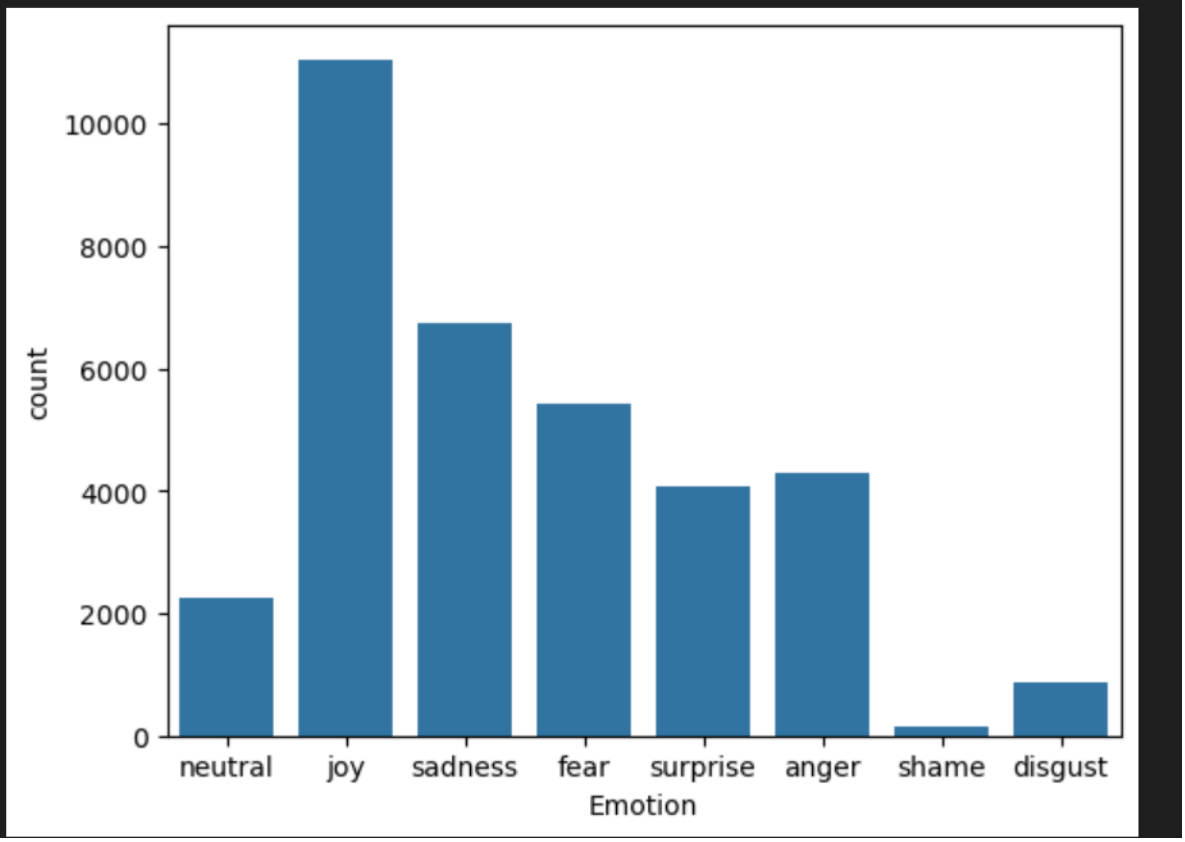
```
Emotion
joy         11045
sadness      6722
fear         5410
anger        4297
surprise     4062
neutral      2254
disgust       856
shame         146
Name: count, dtype: int64
```

Using the seaborn library developed a visual representation of attributes On x axis Emotion is Considered and on Y axis Count is taken

```
sns.countplot(x='Emotion',data=dataset)
```

✓ 0.5s

```
<Axes: xlabel='Emotion', ylabel='count'>
```



For the data preprocessing neat text library is installed with PIP command

```
# Data preprocessing
%pip install neattext
```

✓ 4.4s

```
Requirement already satisfied: neattext in d:\anaconda\lib\site-packages (0.1.3)
Note: you may need to restart the kernel to use updated packages.
```

```
import neattext.functions as nfx

# Remove the user handles
dataset['Clean_Text'] = dataset['Text'].apply(nfx.remove_userhandles)
```

```
dir(nfx)
```

```
['BTC_ADDRESS_REGEX',
 'CURRENCY_REGEX',
 'CURRENCY_SYMB_REGEX',
 'Counter',
 'DATE_REGEX',
 'EMAIL_REGEX',
 'EMOJI_REGEX',
 'HASTAG_REGEX',
 'MASTERCard_REGEX',
 'MD5_SHA_REGEX',
 'MOST_COMMON_PUNCT_REGEX',
 'NUMBERS_REGEX',
 'PHONE_REGEX',
 'PoBOX_REGEX',
 'SPECIAL_CHARACTERS_REGEX',
 'STOPWORDS',
 'STOPWORDS_de',
 'STOPWORDS_en',
 'STOPWORDS_es',
 'STOPWORDS_fr',
 'STOPWORDS_ru',
 'STOPWORDS_yo',
 'STREET_ADDRESS_REGEX',
 'TextFrame',
 'URL_PATTERN',
...
 'term_freq',
 'to_txt',
 'unicodedata',
 'word_freq',
 'word_length_freq']
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Step:4

In data cleaning, the stop words are removed.

```
# Remove the stopwords
dataset['Clean_Text'] = dataset['Clean_Text'].apply(nfx.remove_stopwords)
```

✓ 0.1s

```
dataset
```

✓ 0.0s

	Emotion	Text	Clean_Text
0	neutral	Why ?	?
1	joy	Sage Act upgrade on my to do list for tommorow.	Sage Act upgrade list tommorow.
2	sadness	ON THE WAY TO MY HOMEGIRL BABY FUNERAL!!! MAN ...	WAY HOMEGIRL BABY FUNERAL!!! MAN HATE FUNERALS...
3	joy	Such an eye ! The true hazel eye-and so brill...	eye ! true hazel eye-and brilliant ! Regular f...
4	joy	@lluvmiasantos ugh babe.. hugggzzz for u .! b...	ugh babe.. hugggzzz u .! babe naamazed nga ako...
...	...	...	...
34787	surprise	@MichelGW have you gift! Hope you like it! It'...	gift! Hope like it! hand wear ! It'll warm! Lol
34788	joy	The world didnt give it to me..so the world MO...	world didnt me..so world DEFINITELY cnt away!!!
34789	anger	A man robbed me today .	man robbed today .
34790	fear	Youu call it JEALOUSY, I call it of #Losing YO...	Youu JEALOUSY, #Losing YOU...
34791	sadness	I think about you baby, and I dream about you ...	think baby, dream time

34792 rows × 3 columns

Step:5

To train the dataset, dataset is splitted into two parts. i.e train data and text data

```
#splitting the data into input variables and target variables
x = dataset['Clean_Text']
y = dataset['Emotion']
```

✓ 0.0s

```
#splitting the data into train set and test set

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=42)
```

✓ 0.8s

In this dataset, 70% of data is used for Training and 30% is used for Testing.

Step:6

In training the model, scikit learn is imported

```
#training the module
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from xgboost import XGBClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import precision_score, recall_score, accuracy_score,f1_score
```
✓ 0.0s

```
X_train = list(x_train)
X_test = list(x_test)
le = LabelEncoder()
y_train = le.fit_transform(y_train)
y_test = le.transform(y_test)
```
✓ 0.0s

Step:7

Now the machine learning algorithms are implemented. At first Logistic regression is implemented

```
#create a pipeline
pipe_lr = Pipeline(steps=[('cv',CountVectorizer()),('lr',LogisticRegression())])
#train the pipeline
pipe_lr.fit(x_train,y_train)
#make predictions
y_pred = pipe_lr.predict(X_test)
# print(y_pred)
 #calculate the metrics
precision = precision_score(y_test, y_pred, average='weighted')  # Calculate precision
recall = recall_score(y_test, y_pred, average='weighted')  # Calculate recall
accuracy = accuracy_score(y_test, y_pred)  # Calculate accuracy
f1 = f1_score(y_test, y_pred, average='weighted')  # Calculate F1 score
#print the metrics
print("Logistic Regression Metrics:")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {accuracy:.4f}")
```
✓ 3.2s

```
Logistic Regression Metrics:
Precision: 0.6215
Recall: 0.6201
Accuracy: 0.6201
F1 Score: 0.6201
```

Step:8

Support Vector Machine

```
pipe_svm = Pipeline(steps=[('cv',CountVectorizer()),('svc', SVC(kernel = 'rbf', C = 10))])
pipe_svm.fit(x_train,y_train)
pipe_svm.score(x_test,y_test)
# Make predictions
y_pred = pipe_svm.predict(x_test)

# Calculate metrics
precision = precision_score(y_test, y_pred, average='weighted')  # Calculate precision
recall = recall_score(y_test, y_pred, average='weighted')  # Calculate recall
accuracy = accuracy_score(y_test, y_pred)  # Calculate accuracy
f1 = f1_score(y_test, y_pred, average='weighted')  # Calculate F1 score

# Print the metrics
print("Support Vector  Metrics:")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print(f"Accuracy: {accuracy:.4f}")
```

✓ 4m 16.1s

```
Support Vector  Metrics:
Precision: 0.6242
Recall: 0.6220
F1 Score: 0.6118
Accuracy: 0.6220
```

Step:9

Random Forest

```
pipe_rf = Pipeline(steps=[('cv',CountVectorizer()),('rf', RandomForestClassifier(n_estimators=10))])
pipe_rf.fit(x_train,y_train)
pipe_rf.score(x_test,y_test)
# Make predictions
y_pred_rf = pipe_rf.predict(x_test)

# Calculate metrics
precision_rf = precision_score(y_test, y_pred_rf, average='weighted')  # Calculate precision
recall_rf = recall_score(y_test, y_pred_rf, average='weighted')  # Calculate recall
accuracy_rf = accuracy_score(y_test, y_pred_rf)  # Calculate accuracy
f1_rf = f1_score(y_test, y_pred_rf, average='weighted')  # Calculate F1 score

# Print the metrics
print("Random Forest Classifier Metrics:")
print(f"Precision: {precision_rf:.4f}")
print(f"Recall: {recall_rf:.4f}")
print(f"F1 Score: {f1_rf:.4f}")
print(f"Accuracy: {accuracy_rf:.4f}")
```

✓ 11.5s

```
Random Forest Classifier Metrics:
Precision: 0.5929
Recall: 0.5660
F1 Score: 0.5602
Accuracy: 0.5660
```

Step:10

XGBoost

```
pipe_xgb=Pipeline(steps=[('cv',CountVectorizer()),('xgb',XGBClassifier(n_estimators=10, eval_metric='logloss'))])
pipe_xgb.fit(x_train,y_train)
pipe_xgb.score(x_test,y_test)
# Make predictions
y_pred_xgb = pipe_xgb.predict(x_test)

# Calculate metrics
precision_xgb = precision_score(y_test, y_pred_xgb, average='weighted')  # Calculate precision
recall_xgb = recall_score(y_test, y_pred_xgb, average='weighted')  # Calculate recall
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)  # Calculate accuracy
f1_xgb = f1_score(y_test, y_pred_xgb, average='weighted')  # Calculate F1 score

# Print the metrics
print("XGBoost Classifier Metrics:")
print(f"Precision: {precision_xgb:.4f}")
print(f"Recall: {recall_xgb:.4f}")
print(f"F1 Score: {f1_xgb:.4f}")
print(f"Accuracy: {accuracy_xgb:.4f}")
```

✓ 2.9s

```
XGBoost Classifier Metrics:
Precision: 0.6908
Recall: 0.5022
F1 Score: 0.4687
Accuracy: 0.5022
```

Deploying the model

```
import pickle
with open ("text_emotion_1","wb") as file:
    pickle.dump(pipe_lr,file)
```

Step:11

For the User Interface it was developed with Streamlit Library using Python

```
import streamlit as st


import pandas as pd
import numpy as np
import altair as alt
import pickle
import joblib
import os
print(os.getcwd())
```

```
# breakpoint()
pipe_lr_file=open("D:\\Text Emotion detection\\model\\text_emotion_1", "rb")
pipe_lr = pickle.load(pipe_lr_file)

emotions_emoji_dict = {"anger": "😠", "disgust": "🤮", "fear": "😨 😱", "happy": "🤗", "joy": "😂",
"neutral": "😐", "sad": "😔",
                       "sadness": "😔", "shame": "😳", "surprise": "😮"}


def predict_emotions(docx):
    results = pipe_lr.predict([docx])
    return results[0]


def get_prediction_proba(docx):
    results = pipe_lr.predict_proba([docx])
    return results


def main():
    st.title("Text Emotion Detection")
    st.subheader("Detect Emotions In Text")

    with st.form(key='my_form'):
        raw_text = st.text_area("Type Here")
        submit_text = st.form_submit_button(label='Submit')

    if submit_text:
        col1, col2 = st.columns(2)

        prediction = predict_emotions(raw_text)
        probability = get_prediction_proba(raw_text)

        with col1:
            st.success("Original Text")
            st.write(raw_text)

            st.success("Prediction")
            emoji_icon = emotions_emoji_dict[prediction]
```

```
            st.write("{}:{}".format(prediction, emoji_icon))
            st.write("Confidence:{}".format(np.max(probability)))

        with col2:
            st.success("Prediction Probability")
            #st.write(probability)
            proba_df = pd.DataFrame(probability, columns=pipe_lr.classes_)
            #st.write(proba_df.T)
            proba_df_clean = proba_df.T.reset_index()
            proba_df_clean.columns = ["emotions", "probability"]

            fig = alt.Chart(proba_df_clean).mark_bar().encode(x='emotions', y='probability', color='emotions')
            st.altair_chart(fig, use_container_width=True)




if __name__ == '__main__':
    main()
```

Step:12

To run the program there are two methods

Method1:

Streamlit run app.py

We can directly copy the link and paste in command prompt and run

streamlit run d:/Text Emotion detection/model/app.py

but sometimes It will not work so we can choose method 2

Method 2:

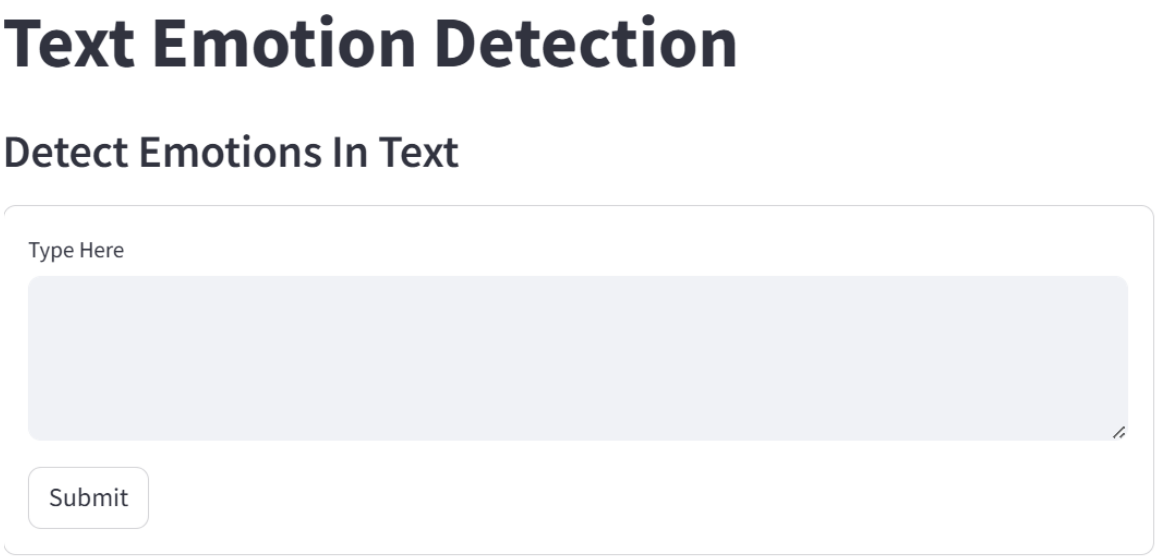
Open command prompt and change the directory and run

PS D:\Text Emotion detection> cd model

PS D:\Text Emotion detection\model> streamlit run app.py

Step:13

Output:



7. Results and Discussion

Based on the F1 score of 0.65 obtained from the logistic regression model for text emotion detection, the model demonstrates moderate effectiveness in classifying emotions from text inputs. While the F1 score indicates a reasonable balance between precision and recall, suggesting some capability in capturing true positives while minimizing false positives and false negatives, there is room for improvement. Further refinement of feature selection, model tuning, or exploration of more advanced techniques like ensemble methods or neural networks could potentially enhance the model's performance, leading to more accurate and reliable predictions in emotion detection tasks.

Table.1. The emotional analysis of all human responses

Sentences	Predicated emotion	Probability
Good day	Joy	0.5022
I am feeling well	Joy	0.527
I slept well, thank you	Joy	0.702
I did, but I do not like those pills	Fear	0.688
Yes, I had French toast which I love	Neutral	0.605
Yes,I had a French toast which I have	Neutral	0.6054
Don’t make granny angry by jumping on the bed	Anger	0.7311
She thought the shame and embarrassment would never end after her crush rejected her in front of the whole school	Fear	0.3699
The smell of the garbage filled me with disgust	Disgust	0.3117
I worry that I will have trouble sleeping anyways	Sadness	0.7227
I cannot do much, I will just watch the news	Sadness	0.45088
Finally that would be amazing	Surprise	0.496

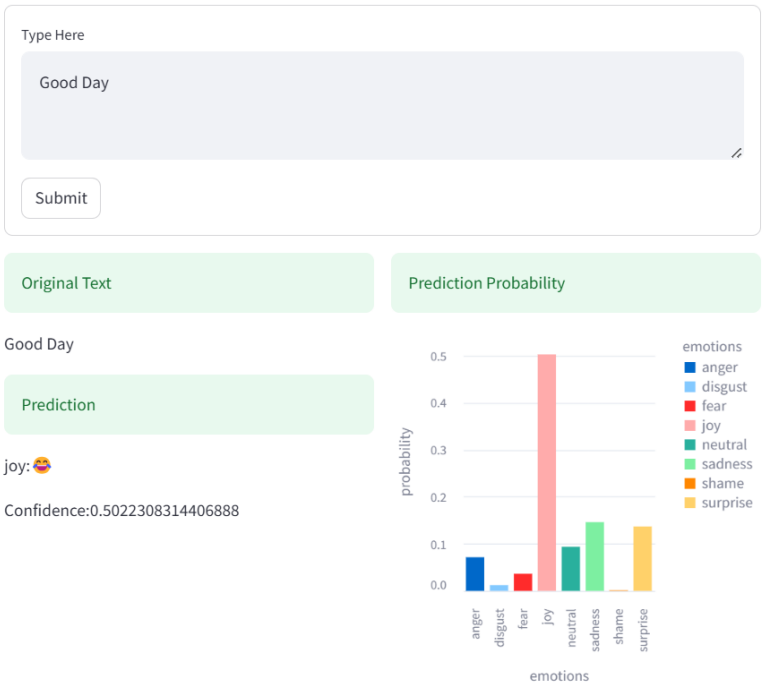


Figure 7.1 A Sentence that predicts Joy

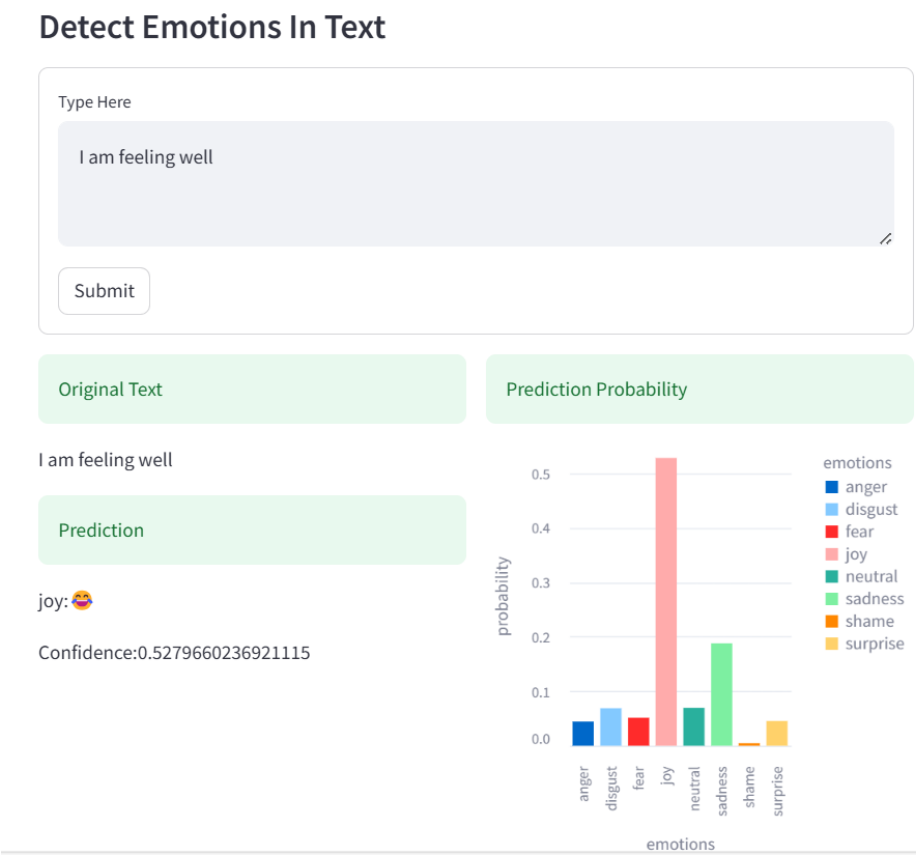


Figure 7.2 A Sentence that predicts Joy

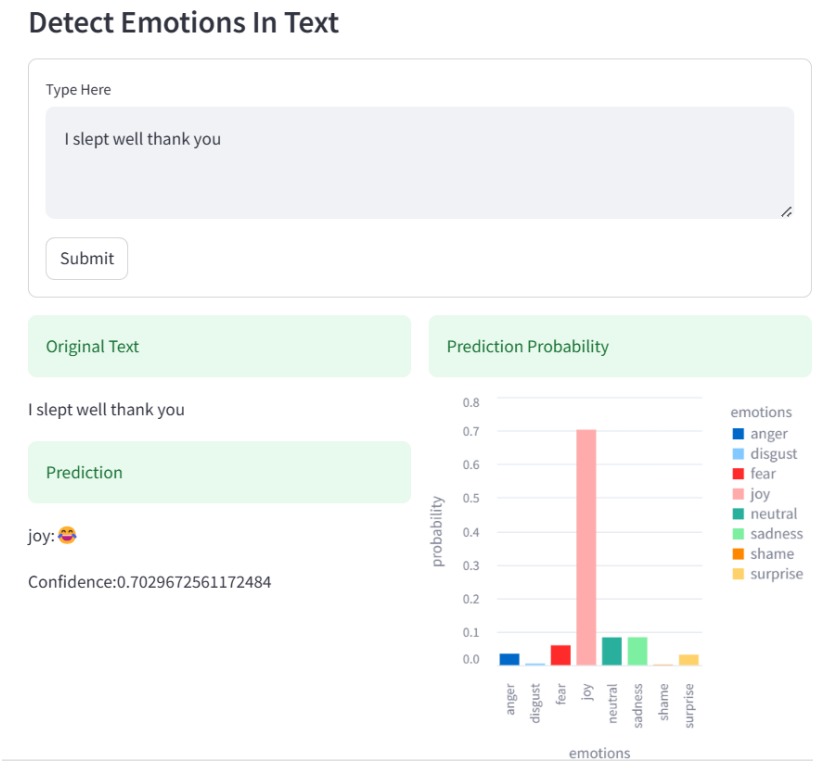


Figure 7.3 A Sentence that predicts Joy

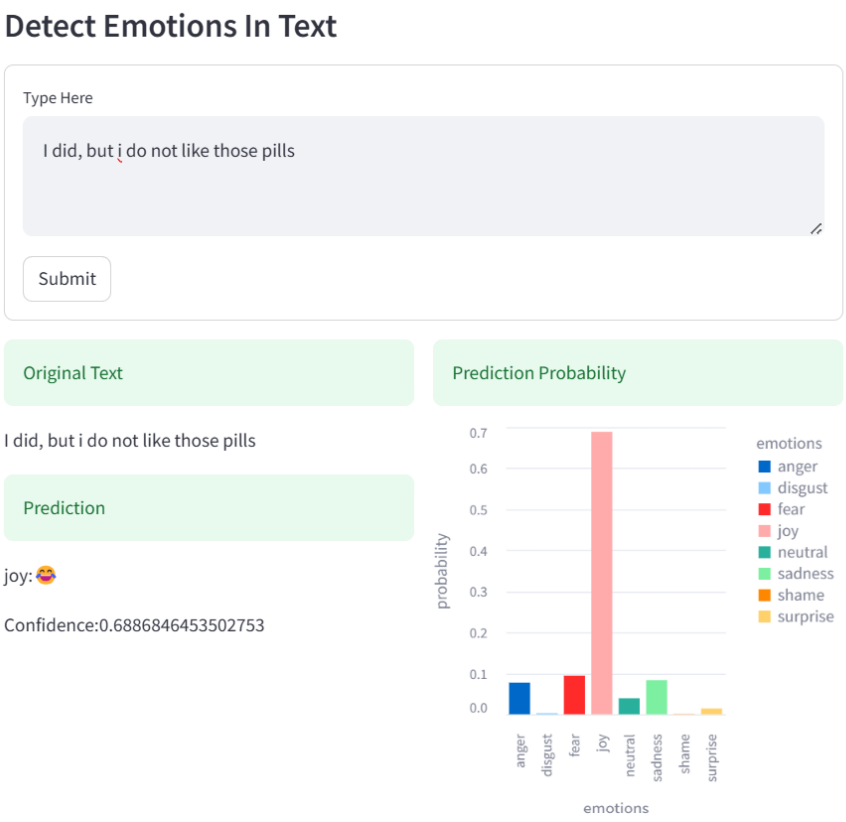


Figure 7.4 A Sentence that predicts Joy

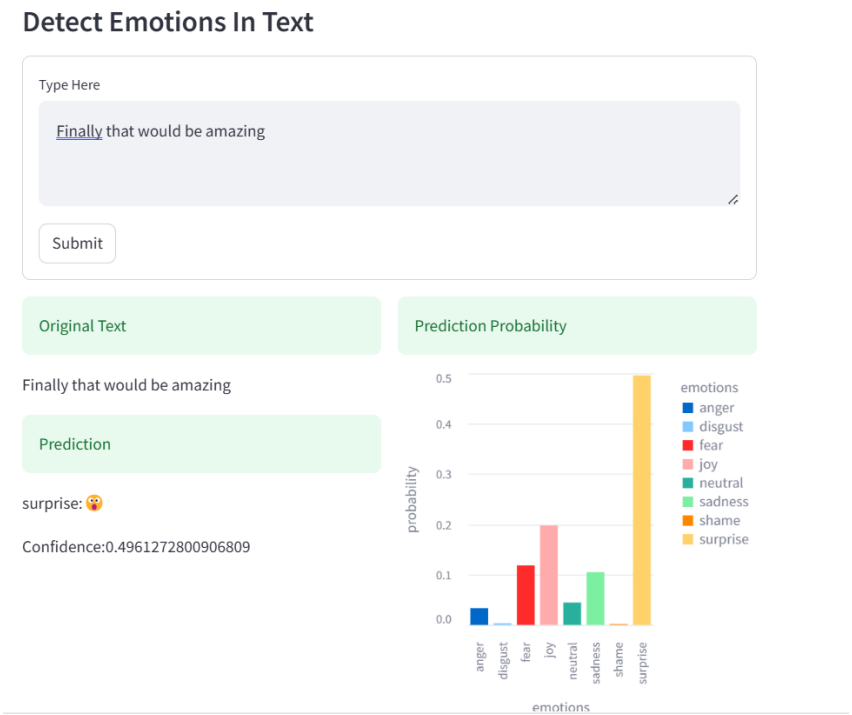


Figure 7.5 A Sentence that predicts Surprise

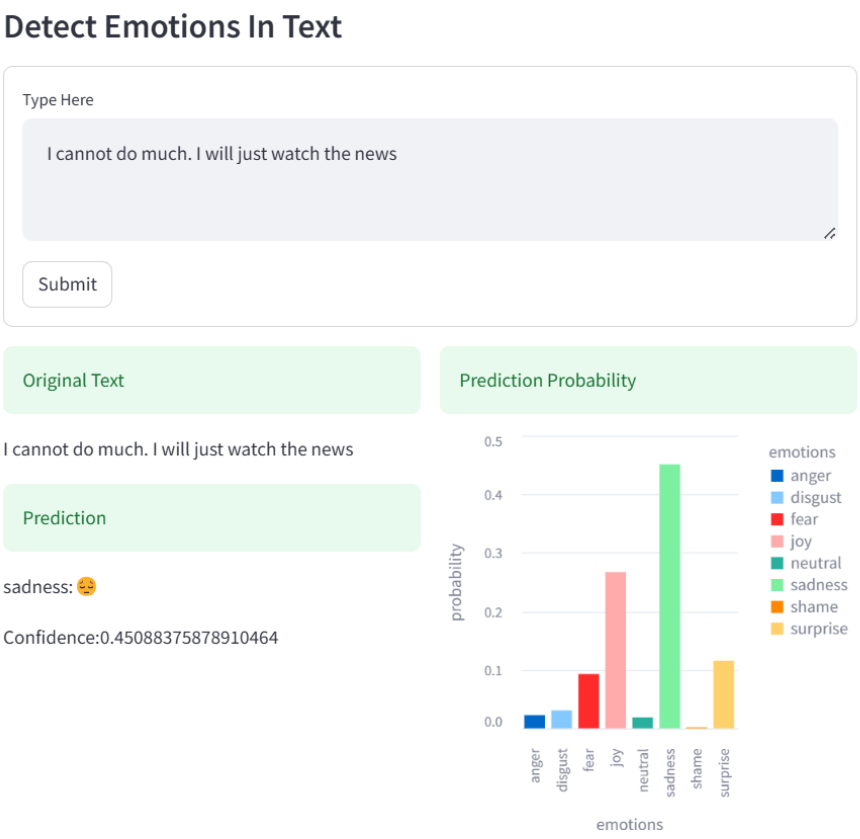


Figure 7.6 A Sentence that predicts Sadness

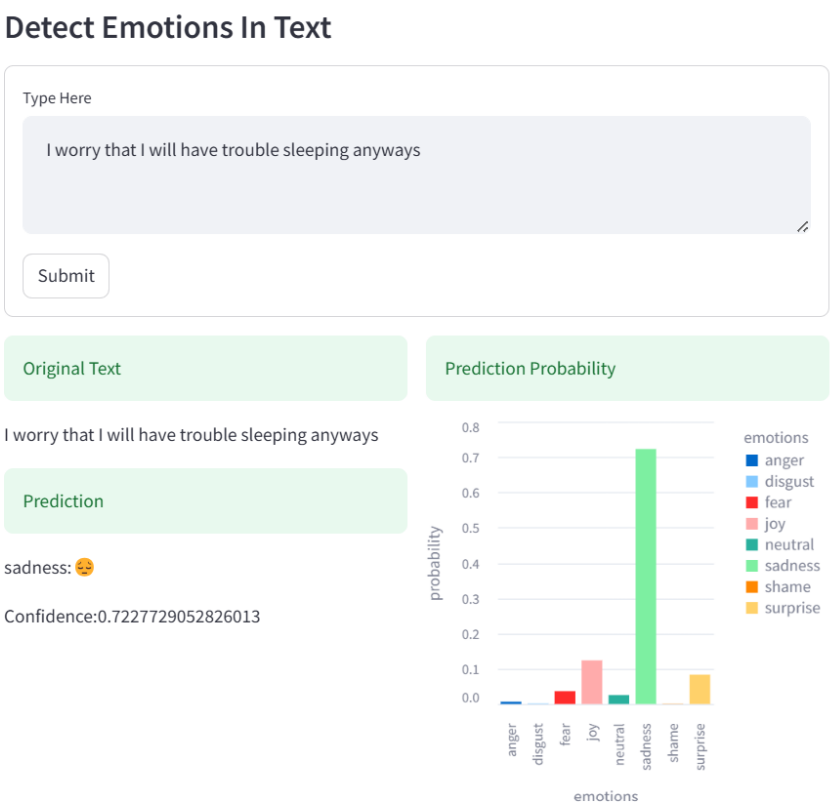


Figure 7.7 A Sentence that predicts Sadness

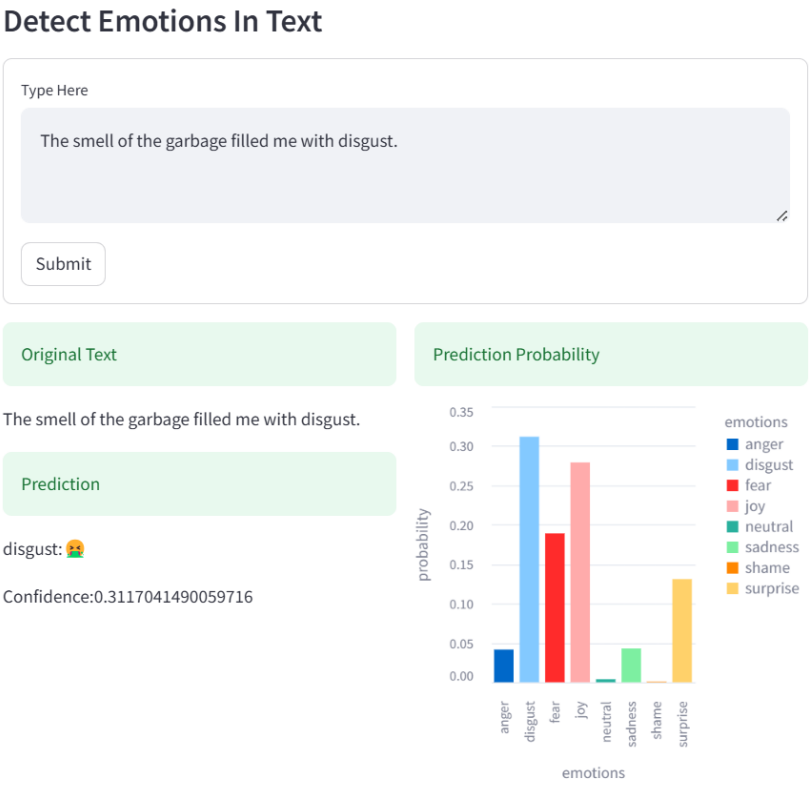


Figure 7.8 A Sentence that predicts Disgust

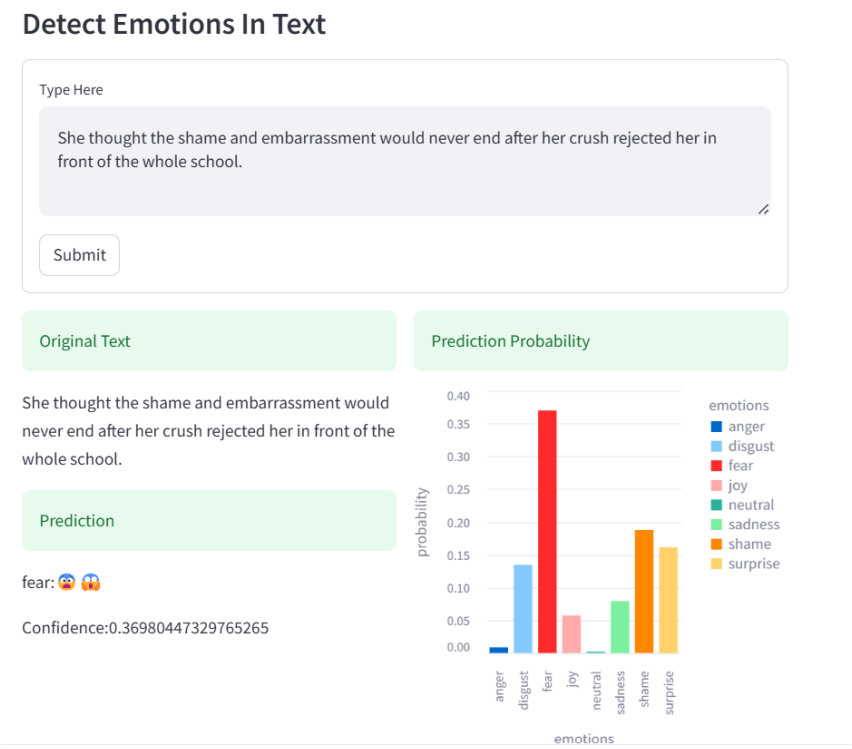


Figure 7.9 A Sentence that predicts Fear

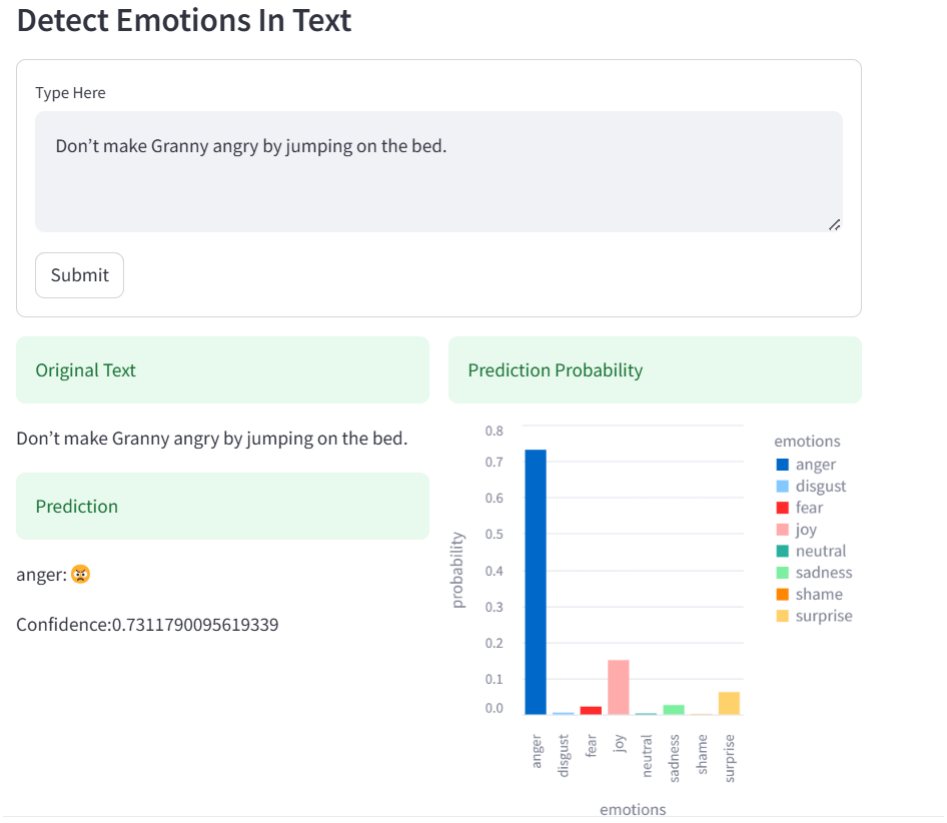


Figure 7.10 A Sentence which predicts Anger

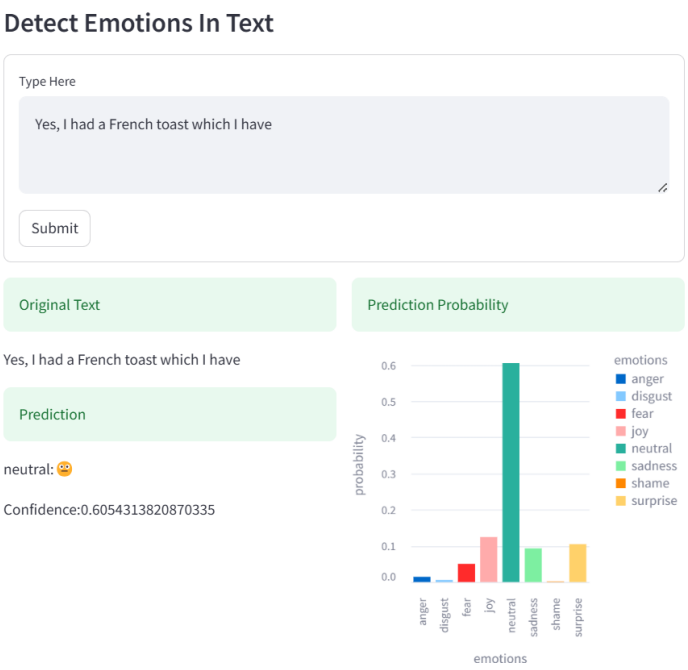


Figure 7.11 A Neutral Sentence

CONCLUSION

The comparative analysis of various machine learning algorithms for text emotion detection highlights nuanced strengths and weaknesses across different approaches. Algorithms like Logistic Regression, SVM, Random Forest, and XGBoost each showcase distinct advantages depending on the dataset and task complexity. Logistic Regression offers simplicity and interpretability, while SVM excels in handling high-dimensional data with kernel tricks. Random Forest provides robustness against overfitting and nonlinear relationships, and XGBoost enhances performance through gradient boosting. Understanding these nuances aids in selecting the most suitable algorithm based on specific application requirements, dataset characteristics, and desired trade-offs between accuracy, interpretability, and computational efficiency in text emotion detection tasks.

REFERENCES

1. Muhammad Abdul-Mageed and Lyle Ungar. 2017. Emonet: Fine-grained emotion detection with gated recurrent neural networks. In Proceedings of the 55th Annual Meeting of the Association for Compu tational Linguistics (Volume 1: Long Papers), vol ume 1, pages 718–728.
2. Hessa AlBalooshi, Shahram Rahmanian, and Rahul Venkatesh Kumar. 2018. Emotionx smartdubai nlp: Detecting user emotions in social media text. In Proceedings of the Sixth Interna tional Workshop on Natural Language Processing for Social Media, pages 45–49.
3. Mohamed R Amer, Behjat Siddiquie, Colleen Richey, and Ajay Divakaran. 2014. Emotion detection in speech using deep networks. In 2014 IEEE inter national conference on acoustics, speech and signal processing (ICASSP), pages 3724–3728. IEEE.
4. Juan Pablo Arias, Carlos Busso, and Nestor Becerra Yoma. 2014. Shape-based modeling of the funda mental frequency contour for emotion detection in speech. Computer Speech & Language, 28(1):278 294.
5. U˘ gur Ayvaz, H¨ useyin G¨ ur¨ uler, and Mehmet Osman De vrim. 2017. Use of facial emotion recognition in e-learning systems. Information Technologies and Learning Tools, 60(4):95–104.
6. Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Ben gio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
7. Ankush Chatterjee, Kedhar Nath Narahari, Meghana Joshi, and Puneet Agrawal. 2019. Semeval-2019 task 3: Emocontext: Contextual emotion detection in text. In In Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval-2019).
8. Ana Raquel Faria, Ana Almeida, Constantino Mar tins, Ramiro Gonc ¸alves, Jos´ e Martins, and Frederico Branco. 2017.
9. A global perspective on an emotional learning model proposal. Telematics and Informat ics, 34(6):824–837.
10. Bharat Gaind, Varun Syal, and Sneha Padgalwar. 2019. Emotion detection and analysis on social media. arXiv preprint arXiv:1901.08458.

11. M Shamim Hossain and Ghulam Muhammad. 2019. Emotion recognition using deep learning approach from audio–visual emotional big data. Information Fusion, 49:69–78.
12. Samira Ebrahimi Kahou, Xavier Bouthillier, Pas cal Lamblin, Caglar Gulcehre, Vincent Michalski, Kishore Konda, S´ ebastien Jean, Pierre Froumenty, Yann Dauphin, Nicolas Boulanger-Lewandowski, et al. 2016.
13. Emonets: Multimodal deep learning ap proaches for emotion recognition in video. Journal on Multimodal User Interfaces, 10(2):99–111. Byoung Ko. 2018.
14. A brief review of facial emotion recognition based on visual information. sensors, 18(2):401. Jasy Suet Yan Liew and Howard R Turtle. 2016. Ex ploring fine-grained emotion detection in tweets. In Proceedings of the NAACL Student Research Work shop, pages 73–80.
15. Wootaek Lim, Daeyoung Jang, and Taejin Lee. 2016. Speech emotion recognition using convolutional and recurrent neural networks. In 2016 Asia-Pacific Sig nal and Information Processing Association Annual SummitandConference(APSIPA), pages 1–4. IEEE.
16. Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. 2017. Neural machine translation (seq2seq) tutorial. https://github.com/tensorflow/nmt.
17. Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention based neural machine translation. arXiv preprint arXiv:1508.04025.
18. Yuki Matsuda, Dmitrii Fedotov, Yuta Takahashi, Yu taka Arakawa, Keiichi Yasumoto, and Wolfgang Minker. 2018. Emotour: Multimodal emotion recognition using physiological and audio-visual features. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Sym posium onPervasive andUbiquitous Computingand Wearable Computers, pages 946–951. ACM.
19. Mostafa Mohammadpour, Hossein Khaliliardali, Seyyed Mohammad RHashemi, and Mohammad M AlyanNezhadi. 2017. Facial emotion recognition using deep convolutional networks. In 2017 IEEE 4th International Conference on Knowledge Based Engineering and Innovation (KBEI), pages 0017–0021. IEEE.
20. Isidoros Perikos and Ioannis Hatzilygeroudis. 2013. Recognizing emotion presence in natural language sentences. In International conference on engineer ing applications of neural networks, pages 30–39. Springer.

21. Duc AnhPhan, Hiroyuki Shindo, and Yuji Matsumoto. 2016. Multiple emotions detection in conversation transcripts. In Proceedings of the 30th Pacific Asia Conference on Language, Information and Compu tation: Oral Papers, pages 85–94.
22. Yanghui Rao. 2016. Contextual sentiment topic model for adaptive social emotion classification. IEEE In telligent Systems, 31(1):41–47.
23. Ahmed E Samy, Samhaa R El-Beltagy, and Ehab Has sanien. 2018. A context integrated model for multi label emotion detection. Procedia computer sci ence, 142:61–71.
24. Caifeng Shan, Shaogang Gong, and Peter W McOwan. 2009. Facial expression recognition based on local binary patterns: A comprehensive study. Image and vision Computing, 27(6):803–816.