MANOSPARSH- THE TOUCH OF MIND/HEART

Submitted By

Devanshi Srivastava -241H0660029

Khushi Varshney -241H0660054

Palak Sharma -241H0660070

Pragya Varshney -241H0660074

For the award of degree of

MCA

Under the supervision of

Supervisor

Dr. Pooja Gupta

Department of Computer Science

Banasthali Vidyapith

Banasthali - 304022

Session: 2024-2025

Certificate

Certified that Devanshi Srivastava, Khushi Varshney, Palak Sharma, Pragya Varshney has carried out the project work titled "**MANOSPARSH-THE TOUCH OF MIND/HEART**" from 14-01-2025 to 2-05-2025 for the award of the MCA from Banasthali Vidyapith under my supervision. The thesis embodies result of original work and studies carried out by students themselves and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Dr. Pooja Gupta

Designation:

Place: Banasthali Vidyapith

Date:

Abstract

Understanding the balance between emotional and logical thinking in human communication has long intrigued researchers. The *ManoSparsh* project aims to explore this balance using computational techniques by analyzing textual input. It proposes a dual-model framework that combines sentiment analysis and emotion detection to determine how much of a user's response is driven by emotion (heart) and how much by logic (mind).

The system works exclusively on written text without requiring facial expressions or physiological signals. First, sentiment analysis is used to assess the overall polarity of the input—whether the user expresses positivity, negativity, or neutrality. Then, emotion detection is performed to classify specific emotional states such as joy, fear, sadness, or anger. The insights from both modules are combined to compute a percentage-based output, indicating the logical and emotional contribution within the text.

This study utilizes supervised machine learning algorithms like Naive Bayes, Support Vector Machine (SVM), Random Forest, and Voting Classifier. Tools such as Jupyter Notebook, Google Collab and Hugging Face and libraries like Python’s scikit-learn, pandas, NumPy, and NLTK are used for implementation. *ManoSparsh* offers a new direction for computational psychology and could contribute to applications in behavioral research, mental health support systems, and intelligent dialogue agents.

Acknowledgement

We take this opportunity to express our heartfelt gratitude to all those who supported and guided us throughout the journey of our project, **“Manosparsh”-** a fusion of Sentiment Analysis and Emotion Detection.

First and foremost, we are deeply indebted to our project supervisor **Dr. Pooja Gupta** for her unwavering support, insightful mentorship, and continuous encouragement. Her expertise has been a guiding light through the intricate pathways of machine learning, helping us navigate the complex realms of natural language processing with confidence and clarity.

We extend our sincere thanks to our project coordinator **Dr. Ashok Kumar** for his consistent guidance and support. His valuable suggestions, encouragement, and timely inputs have played a vital role in shaping and on-schedule completion of our project.

Also, our respectful thanks to **Head of the Computer Engineering Department**, Banasthali Vidyapith whose support helped us elevate the quality of our work and contributed to our academic growth.

Last but certainly not least, we are immensely grateful to our friends and college mates. Their collaborative spirit, constant motivation, and unwavering support helped us overcome challenges and celebrate every milestone along the way. This journey would not have been as enriching without their presence and encouragement.

To each and every one who played a part in bringing **Manosparsh** to life — thank you. Your contribution has been invaluable, and we are truly fortunate to have had your support throughout this endeavor.

Table of Contents

Objective

Human decision-making often reflects a conflict between emotional intuition and rational judgment. While advancements in sentiment analysis and emotion detection have enabled machines to interpret affective states from text, they typically operate in isolation and fail to assess the deeper interplay between logic (mind) and emotion (heart). This limitation makes it challenging to computationally evaluate how a person's expression leans-whether it is more practical or emotionally influenced.

ManoSparsh aims to bridge this gap by developing a machine learning-based model that not only detects emotions and sentiments from user-written text but also interprets the relative contribution of emotional versus rational thinking. By fusing the outputs of sentiment analysis and emotion detection, the system computes a percentage-based estimate indicating whether the input is guided more by the heart or the mind. This provides a unique layer of introspective analysis, offering insights into a user's psychological balance during expression.

Following are the objectives:

- To review the techniques of machine learning and identifying datasets to be used.
- Processing of datasets
- Selecting the respective machine learning model to evaluate sentiment analysis and emotion detection
- Implementing the machine learning models for sentiment analysis and emotion detection on the datasets selected.
- To review the techniques of deep learning and understanding the models that are used on large datasets.
- To give a guided and theoretical tool on how the outcomes of sentiment analysis and emotion detection could be combined with contrast with the psychological theories like dual system of sentences.
- To deploy the model using hugging face.

Introduction

Understanding human psychology has always been an arduous task. With the increasing dependency of humans on technological aspects, the research to implement the human psychological theories and concepts holds great importance. Many researches have already been conducted in this field and numerous are going on such as behavioral analysis, symptom monitoring, Diagnostic support etc. This project titled **ManoSparsh: The touch of Mind/Soul** manifests as the transformative bridge between logical and emotional thinking.

There are various methods of communication, one of which is textual input, a prominent and anonymous way to express oneself in the virtual world. The two pillars for understanding the logical and emotional aspect of the sentence are Sentiment Analysis and Emotion Detection.

Sentiment Analysis: It is an application of NLP (Natural Language Processing) in which a textual input is analyzed to be as Positive, Negative or Neutral. There are two battling approaches: one is machine learning and another is deep learning. Although one is the excellent set of the other, one comes out to be more efficient than the other. The models of machine learning themselves compete for the best outcome like Naïve Bayes, SVM (Support Vector Machine), Random Forest, etc. Whereas the models like LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), CNN (Convolutional Neural Network) are popular for the deep learning approach.

Looking into the importance, sentiment analysis is essential in interpreting human emotions. Overall, this analysis identifies feelings stated in comments, reviews, and other text inputs, which helps companies to understand the user experiences better and improve decision-making for customer satisfaction. Sentiment analysis is also essential for analyzing brand reputation, keeping a check on public opinion, and promptly resolving problems. It is an essential tool for utilizing insights from massive amounts of textual data, with applications ranging from content control to mental health monitoring to political analysis. Hence, Sentiment analysis is the foremost task to know the overall sentiment of the text and holds a crucial percentage to determine the logicality or emotionality.

Emotion Detection: It is an integrated field of artificial intelligence, psychology, and linguistics. It refers to the extraction of emotions from the specific text. The simpler the task, the more complex the models need to be to compute the emotions effectively. As a result, machine learning and deep learning models have identified a wide range of emotional states like Joy, Happiness, Sadness, Anger, Surprise, and many other classes.

Speaking of the importance, emotion detection is a major part to determine the actual emotion of the text. It bridges the emotional gap between the humans and the machines which allows the systems to comprehend and respond to human emotions in an efficient manner which is very useful in the recent times when the latter has become the part and parcel of our daily lives.

The product of the above two models can contribute to the implementation of the theoretical research of psychology (Dual-Process Theory of Cognition by: Daniel Kahneman and Amos Tversky) on the duality of logic and emotion in a text can be computed by the machine and enhancing the technology and its efficiency in the real world.

Methodology Used

The project "Manosparsh" is built upon the fusion of two core natural language processing techniques: **Sentiment Analysis and Emotion Detection**. Each module was initially developed as a standalone unit, using distinct machine learning algorithms. However, both shared a common data preprocessing pipeline, ensuring consistency across the framework. The outputs of both models were later integrated under the Manosparsh logic, providing a balanced interpretation of emotional depth and practical thinking from textual data.

Below is the structured outline of the methodology followed:

1. Data Collection

- Dataset Selection and Collection:Relevant datasets were gathered to train and test the two modules. These datasets included real-world textual data such as customer reviews, social media comments, and survey responses.
    - For Sentiment Analysis, the dataset was labeled with sentiments such as positive, negative, and neutral.
    - For Emotion Detection, the dataset was tagged with emotions such as happiness, anger, sadness, surprise, and others.
    - Datasets Used:
        - Twitter Sentiment Analysis dataset (Kaggle)
        - Emotion Dataset (GitHub)

2. Data Preprocessing

A shared preprocessing pipeline was applied to both datasets:

- Data Cleaning: Removed noise, such as special characters, stop words, and non-informative elements.
- Tokenization: Split text into individual tokens (words) to facilitate feature extraction.
- Normalization: Converted all text to lowercase for uniformity.

3. Feature Extraction

- TF-IDF (Term Frequency-Inverse Document Frequency): Used to convert textual data into numerical vectors. TF-IDF effectively highlights words that are important in a document while reducing the weight of commonly occurring words, thus combining the simplicity of Bag of Words with the value of word relevance.

4. Model Training and Testing

- Sentiment Analysis Module
    - Algorithm Used: Multinomial Naive Bayes
    - Training: The model was trained on preprocessed sentiment-labeled data.
    - Testing: Tested using the same feature-extracted data to predict sentiment classes accurately.
- Emotion Detection Module
    - Algorithm Used: Support Vector Machine (SVM) with a non-linear 'rbf' kernel
    - Training: SVM was trained on the labeled emotional data to recognize and separate different emotional states.
    - Testing: Evaluated using the same preprocessing pipeline and optimized to detect nuanced emotional tones in text.

5. Manosparsh Logic: Unified Interpretation

After developing both models individually, Manosparsh was conceptualized to blend the outputs of sentiment and emotion analysis into a cohesive interpretative logic. The integration process involved:

- Execution of both models on the same textual input.
- Extraction of results from each module — i.e., sentiment class and emotional tone.
- Weighted scoring logic: A formula was used to assign scores to both outputs, based on the emotional intensity and sentiment polarity.
- Final Output: A percentage split representing 'Heart vs Mind' orientation — quantifying whether the input reflects more emotional (heart) or practical (mind) reasoning.

This blended model gives Manosparsh its uniqueness — turning textual analysis into a psychologically insightful tool.

6. Model Evaluation

Both modules were rigorously evaluated using standard performance metrics:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

These metrics helped assess each model’s classification effectiveness and guided improvements during model optimization.

Dual Process Theory – A Psychological Perspective

Introduction to the Theory: Dual Process Theory is a foundational concept in cognitive psychology that seeks to explain how individuals process information and make decisions. This theory was extensively explored and popularized by **Daniel Kahneman**, a Nobel Prize-winning psychologist, in his book *"Thinking, Fast and Slow"*. Alongside his collaborator **Amos Tversky**, Kahneman categorized human thinking into **two separate cognitive systems**, each responsible for different types of thought processes.

The Two Systems of Thought:

- **System 1 – Intuitive and Emotional Processing**

System 1 refers to the automatic, fast, and unconscious mode of thinking. It does not require intentional effort and is largely shaped by:

- Emotions
- Intuition
- Past experiences
- Immediate perceptions

This system is **reactive and impulsive**—it often governs our quick judgments, snap decisions, and emotional reactions. For instance, when someone reacts emotionally to a distressing news headline or instinctively smiles at a compliment, **System 1** is at play.

- **System 2 – Analytical and Rational Processing**

System 2, on the other hand, represents the deliberate, effortful, and logical mode of thinking. It is activated when tasks require:

- Conscious thought
- Logical reasoning
- Problem-solving
- Evaluation of consequences

Unlike System 1, this system is slower and more demanding, but it allows individuals to overcome biases and make reasoned decisions. It becomes prominent when solving math problems, planning for the future, or making calculated decisions.

Psychological Significance: Kahneman's dual process model illustrates the **interplay between instinctive emotions and logical reasoning**. It challenges the assumption that people always act rationally, showing instead that **System 1 can often dominate decision-making**, leading to cognitive biases.

This theory has deeply influenced the fields of:

- Behavioural economics
- Cognitive psychology
- Moral psychology
- Human-computer interaction

It also explains how emotional states can sometimes **override rational judgment**, particularly in situations of stress, urgency, or uncertainty.

Relevance to ManoSparsh: In the ManoSparsh project, the logic is built upon the core principle of this psychological framework:

- **Emotional tendencies (System 1)** are captured through the **Emotion Detection Model**, which analyzes underlying emotions like joy, fear, or sadness in a given text.
- **Rational thought (System 2)** is interpreted through **Sentiment Analysis**, identifying a logical tone like positive, neutral, or negative.

By quantifying the influence of both systems—emotion (heart) and logic (mind)—ManoSparsh mimics the cognitive mechanisms suggested in Dual Process Theory. The final outcome (e.g., 70% heart, 30% mind) represents the balance between intuitive and rational processing, just as Kahneman theorized.

Tools and Techniques

Following are the tools and techniques for Sentiment Analysis and Emotion Detection Using Naive Bayes and SVM respectively.

Algorithms and Formulas

- **Naive Bayes Algorithm**
  - **Formula:**

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- $P(C|X)$ = Posterior probability of class C given feature set X.
- $P(X|C)$ = Likelihood of feature set X given class C.
- $P(C)$ = Prior probability of class C.
- $P(X)$ = Probability of feature set X.

  - Used for text classification in Sentiment Analysis by calculating probabilities for each class (positive, negative, and neutral) based on word frequencies.

- **Support Vector Machine (SVM) Algorithm**
  - **Formula:**

$$\text{Minimize: } \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \xi_i$$

Subject to:

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Where:

- w = Weights of the hyperplane.
- b = Bias term.
- C = Regularization parameter.
- ξ_i = Slack variable for misclassification.

  - Non-Linear SVM is used in Emotion Detection for classification of emotions in the textual input as Happiness, Joy, Sadness, Anger, etc.

Software Tools

- **Programming Language**: Python
- **Libraries:** Pandas, NLTK (Natural Language Toolkit), Scikit-learn, Matplotlib/Seaborn
- **Integrated Development Environments (IDEs):** Jupyter Notebook and Google Colab.

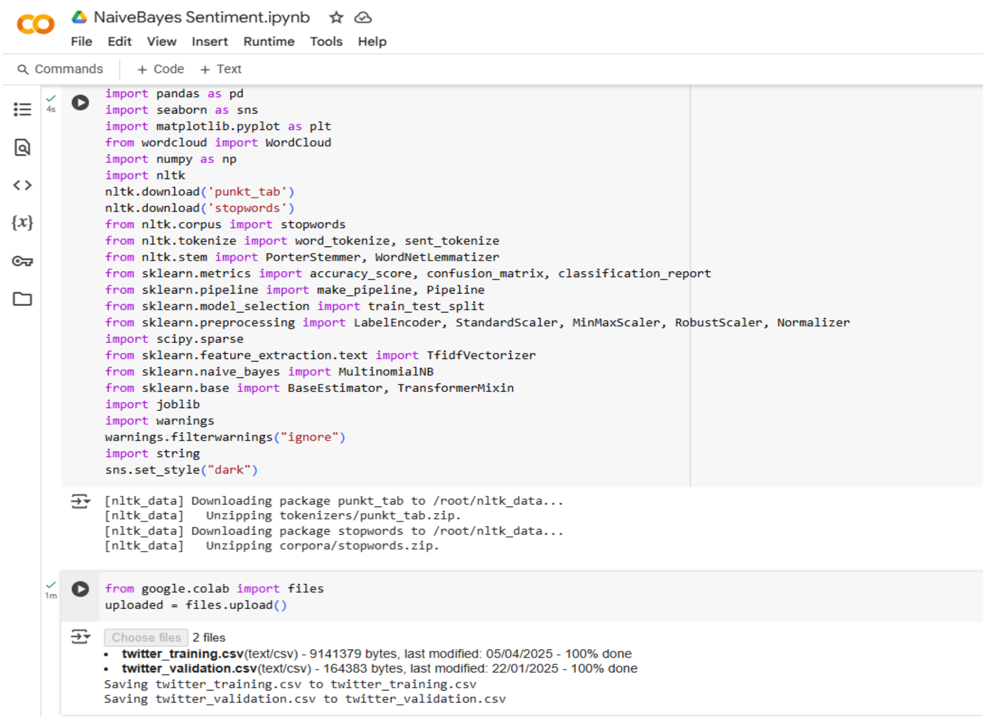
Techniques

- **Data Preprocessing**:
    - **Tokenization**: Splitting text into words.
    - **Stop word Removal**: Eliminating common words like, “and”, “the”, “is”, etc.
    - **Stemming/Lemmatization**: Reducing words to their base form.
    - **Vectorization**: Representing the textual data numerically using TF-IDF (Term Frequency-Inverse Document Frequency) technique.
- **Data Splitting**:
    - Dividing the dataset into training and testing set.
- **Model Training and Testing**:
    - Training the models Naive Bayes and SVM on labeled data for sentiments and emotions respectively.
    - Representing the performance using metrics like accuracy score, precision, recall, F1-score, and confusion matrix.
- **Visualization**:
    - Visualizing the comparison between the training and testing accuracy of both the modules using bar-graph.
- **Deployment:** Using hugging face for the deployment of our project- “Manosparsh”

Coding

Sentiment Analysis Model:

- Importing libraries and Datasets:


NaiveBayes Sentiment.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

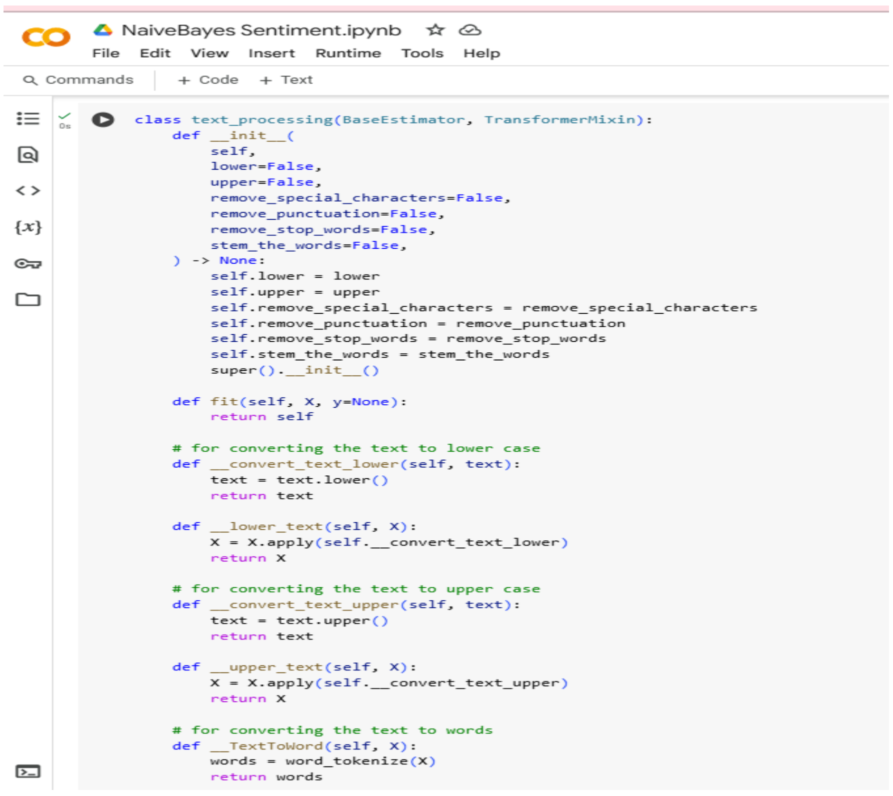
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import numpy as np
import nltk
nltk.download('punkt_tab')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, RobustScaler, Normalizer
import scipy.sparse
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.base import BaseEstimator, TransformerMixin
import joblib
import warnings
warnings.filterwarnings("ignore")
import string
sns.set_style("dark")
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

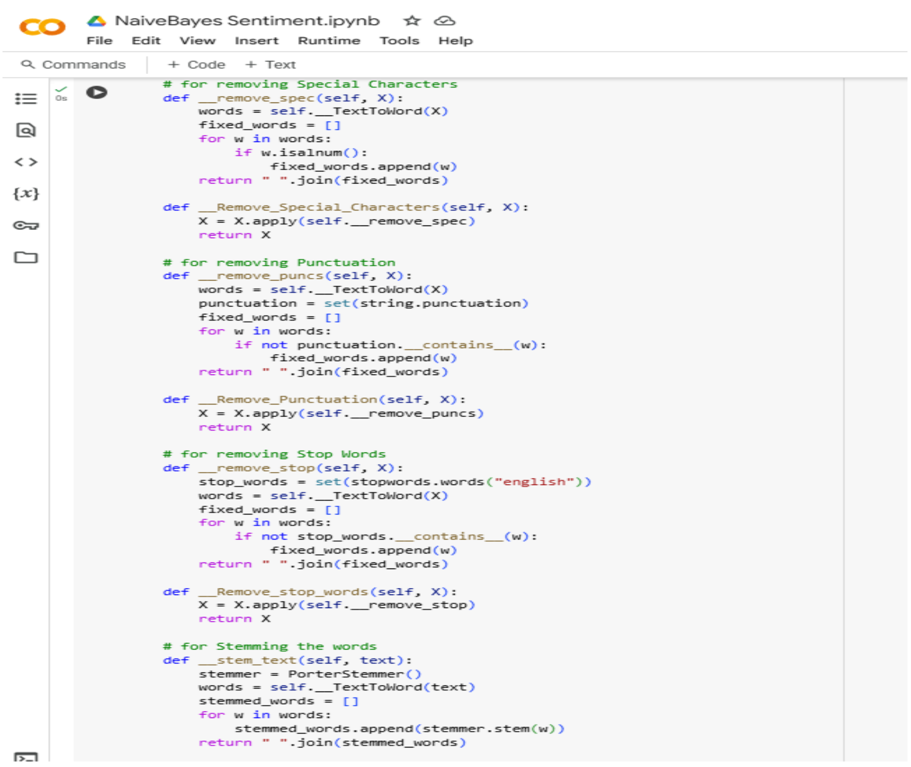
```python
from google.colab import files
uploaded = files.upload()
```

```
Choose files 2 files
• twitter_training.csv(text/csv) - 9141379 bytes, last modified: 05/04/2025 - 100% done
• twitter_validation.csv(text/csv) - 164383 bytes, last modified: 22/01/2025 - 100% done
Saving twitter_training.csv to twitter_training.csv
Saving twitter_validation.csv to twitter_validation.csv
```

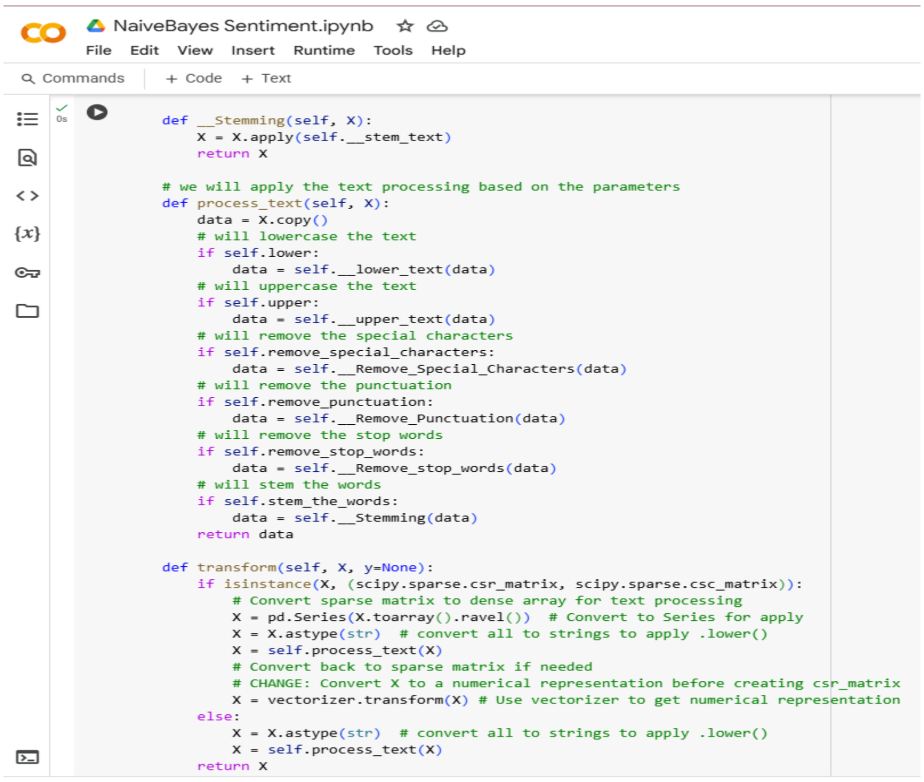
- Text Processing:


NaiveBayes Sentiment.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```python
class text_processing(BaseEstimator, TransformerMixin):
    def __init__(
        self,
        lower=False,
        upper=False,
        remove_special_characters=False,
        remove_punctuation=False,
        remove_stop_words=False,
        stem_the_words=False,
    ) -> None:
        self.lower = lower
        self.upper = upper
        self.remove_special_characters = remove_special_characters
        self.remove_punctuation = remove_punctuation
        self.remove_stop_words = remove_stop_words
        self.stem_the_words = stem_the_words
        super().__init__()

    def fit(self, X, y=None):
        return self

    # for converting the text to lower case
    def __convert_text_lower(self, text):
        text = text.lower()
        return text

    def __lower_text(self, X):
        X = X.apply(self.__convert_text_lower)
        return X

    # for converting the text to upper case
    def __convert_text_upper(self, text):
        text = text.upper()
        return text

    def __upper_text(self, X):
        X = X.apply(self.__convert_text_upper)
        return X

    # for converting the text to words
    def __TextToWord(self, X):
        words = word_tokenize(X)
        return words
```

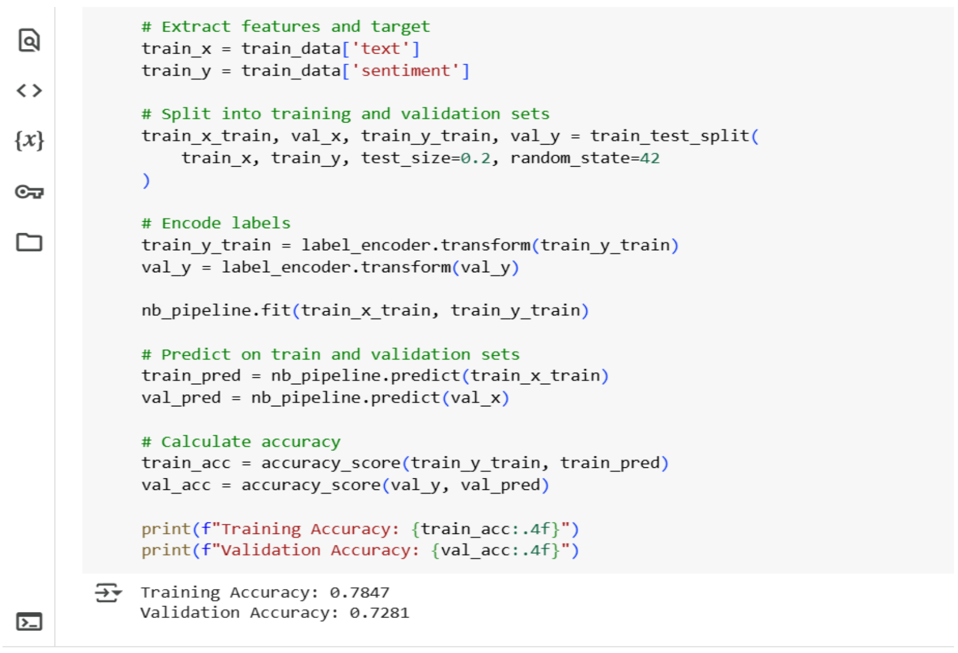

NaiveBayes Sentiment.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```python
    # for removing Special Characters
    def __remove_spec(self, X):
        words = self.__TextToWord(X)
        fixed_words = []
        for w in words:
            if w.isalnum():
                fixed_words.append(w)
        return " ".join(fixed_words)

    def __Remove_Special_Characters(self, X):
        X = X.apply(self.__remove_spec)
        return X

    # for removing Punctuation
    def __remove_puncs(self, X):
        words = self.__TextToWord(X)
        punctuation = set(string.punctuation)
        fixed_words = []
        for w in words:
            if not punctuation.__contains__(w):
                fixed_words.append(w)
        return " ".join(fixed_words)

    def __Remove_Punctuation(self, X):
        X = X.apply(self.__remove_puncs)
        return X

    # for removing Stop Words
    def __remove_stop(self, X):
        stop_words = set(stopwords.words("english"))
        words = self.__TextToWord(X)
        fixed_words = []
        for w in words:
            if not stop_words.__contains__(w):
                fixed_words.append(w)
        return " ".join(fixed_words)

    def __Remove_stop_words(self, X):
        X = X.apply(self.__remove_stop)
        return X

    # for Stemming the words
    def __stem_text(self, text):
        stemmer = PorterStemmer()
        words = self.__TextToWord(text)
        stemmed_words = []
        for w in words:
            stemmed_words.append(stemmer.stem(w))
        return " ".join(stemmed_words)
```


NaiveBayes Sentiment.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

```python
    def __Stemming(self, X):
        X = X.apply(self.__stem_text)
        return X

    # we will apply the text processing based on the parameters
    def process_text(self, X):
        data = X.copy()
        # will lowercase the text
        if self.lower:
            data = self.__lower_text(data)
        # will uppercase the text
        if self.upper:
            data = self.__upper_text(data)
        # will remove the special characters
        if self.remove_special_characters:
            data = self.__Remove_Special_Characters(data)
        # will remove the punctuation
        if self.remove_punctuation:
            data = self.__Remove_Punctuation(data)
        # will remove the stop words
        if self.remove_stop_words:
            data = self.__Remove_stop_words(data)
        # will stem the words
        if self.stem_the_words:
            data = self.__Stemming(data)
        return data

    def transform(self, X, y=None):
        if isinstance(X, (scipy.sparse.csr_matrix, scipy.sparse.csc_matrix)):
            # Convert sparse matrix to dense array for text processing
            X = pd.Series(X.toarray().ravel())  # Convert to Series for apply
            X = X.astype(str)  # convert all to strings to apply .lower()
            X = self.process_text(X)
            # Convert back to sparse matrix if needed
            # CHANGE: Convert X to a numerical representation before creating csr_matrix
            X = vectorizer.transform(X) # Use vectorizer to get numerical representation
        else:
            X = X.astype(str)  # convert all to strings to apply .lower()
            X = self.process_text(X)
        return X
```

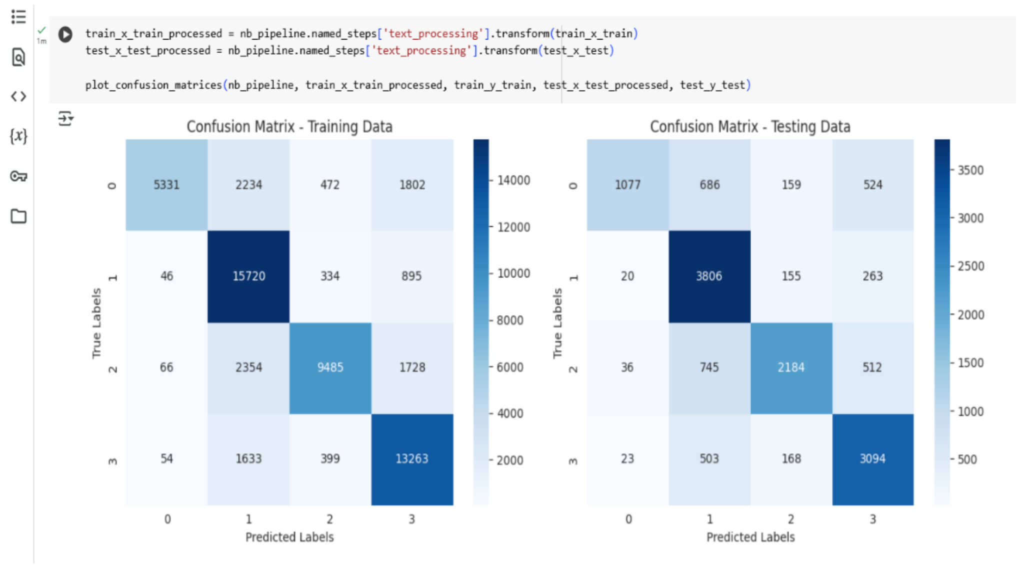
- Feature Extraction and Model Training



```
# Extract features and target
train_x = train_data['text']
train_y = train_data['sentiment']

# Split into training and validation sets
train_x_train, val_x, train_y_train, val_y = train_test_split(
    train_x, train_y, test_size=0.2, random_state=42
)

# Encode labels
train_y_train = label_encoder.transform(train_y_train)
val_y = label_encoder.transform(val_y)

nb_pipeline.fit(train_x_train, train_y_train)

# Predict on train and validation sets
train_pred = nb_pipeline.predict(train_x_train)
val_pred = nb_pipeline.predict(val_x)

# Calculate accuracy
train_acc = accuracy_score(train_y_train, train_pred)
val_acc = accuracy_score(val_y, val_pred)

print(f"Training Accuracy: {train_acc:.4f}")
print(f"Validation Accuracy: {val_acc:.4f}")
```

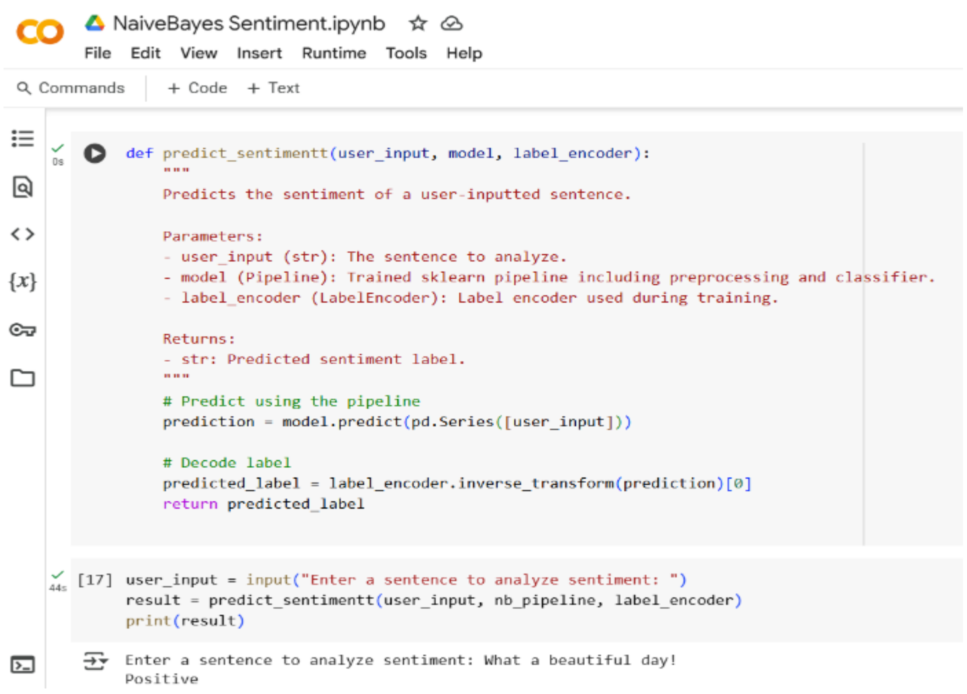
```
Training Accuracy: 0.7847
Validation Accuracy: 0.7281
```

- Confusion Matrix



```
train_x_train_processed = nb_pipeline.named_steps['text_processing'].transform(train_x_train)
test_x_test_processed = nb_pipeline.named_steps['text_processing'].transform(test_x_test)

plot_confusion_matrices(nb_pipeline, train_x_train_processed, train_y_train, test_x_test_processed, test_y_test)
```

- Testing the model


NaiveBayes Sentiment.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text

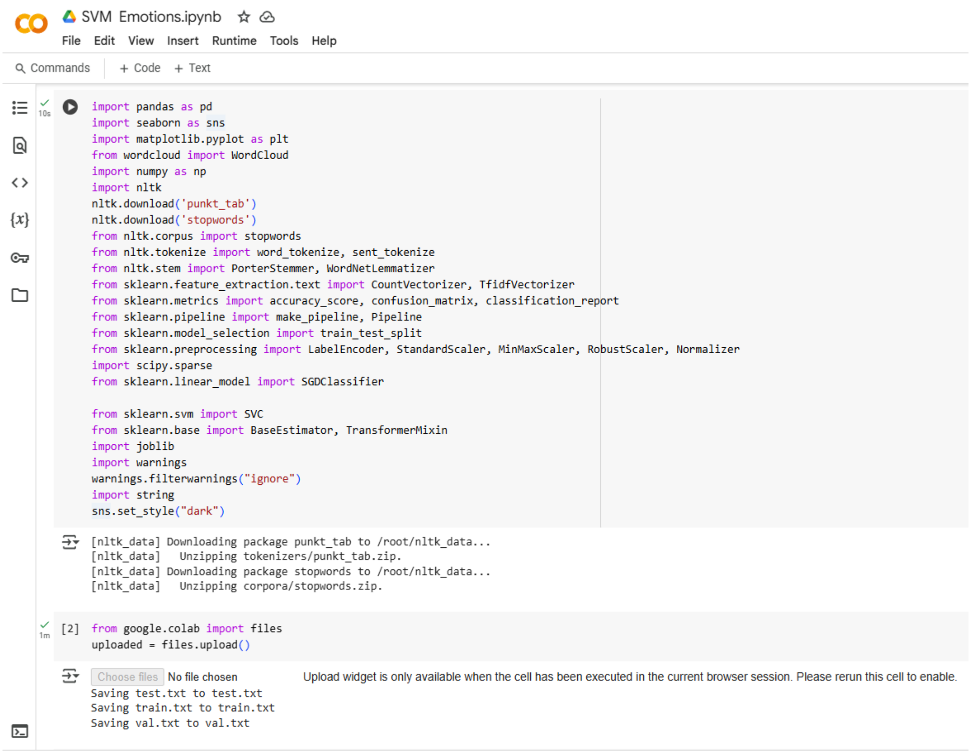
```
def predict_sentimentt(user_input, model, label_encoder):
    """
    Predicts the sentiment of a user-inputted sentence.

    Parameters:
    - user_input (str): The sentence to analyze.
    - model (Pipeline): Trained sklearn pipeline including preprocessing and classifier.
    - label_encoder (LabelEncoder): Label encoder used during training.

    Returns:
    - str: Predicted sentiment label.
    """
    # Predict using the pipeline
    prediction = model.predict(pd.Series([user_input]))

    # Decode label
    predicted_label = label_encoder.inverse_transform(prediction)[0]
    return predicted_label
```

```
[17] user_input = input("Enter a sentence to analyze sentiment: ")
     result = predict_sentimentt(user_input, nb_pipeline, label_encoder)
     print(result)
```

```
Enter a sentence to analyze sentiment: What a beautiful day!
Positive
```

Emotion Detection Model:

- Importing libraries and Datasets:



```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import numpy as np
import nltk
nltk.download('punkt_tab')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.pipeline import make_pipeline, Pipeline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, RobustScaler, Normalizer
import scipy.sparse
from sklearn.linear_model import SGDClassifier

from sklearn.svm import SVC
from sklearn.base import BaseEstimator, TransformerMixin
import joblib
import warnings
warnings.filterwarnings("ignore")
import string
sns.set_style("dark")
```

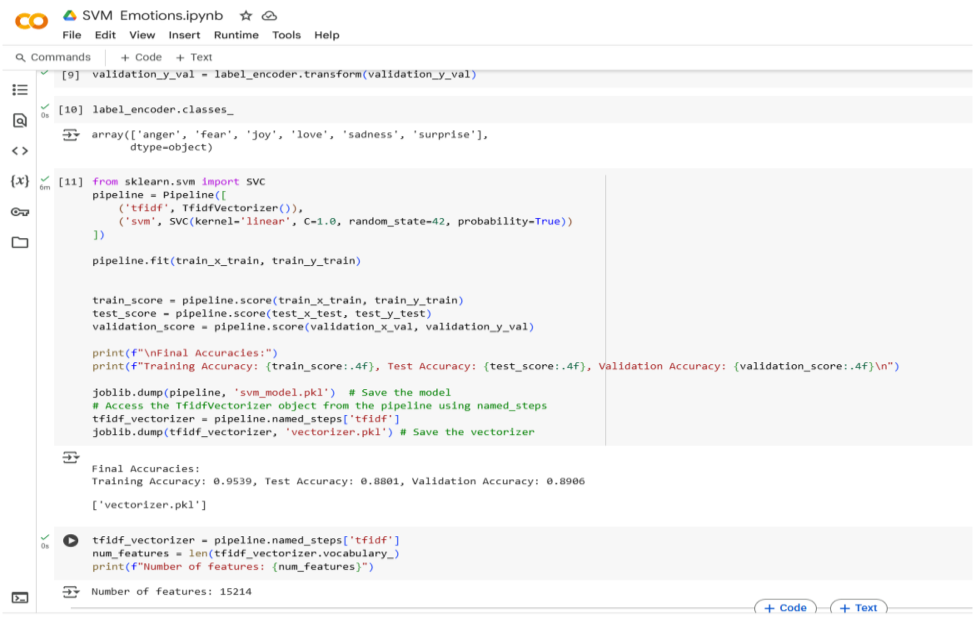
```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[2] from google.colab import files
    uploaded = files.upload()
```

```
Choose files No file chosen        Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving test.txt to test.txt
Saving train.txt to train.txt
Saving val.txt to val.txt
```

- Text Processing: The text processing techniques are common for both the models: Sentiment Analysis and Emotion Detection.

- Feature Extraction and Model Training:



```
[9] validation_y_val = label_encoder.transform(validation_y_val)
```

```
[10] label_encoder.classes_
```

```
array(['anger', 'fear', 'joy', 'love', 'sadness', 'surprise'],
      dtype=object)
```

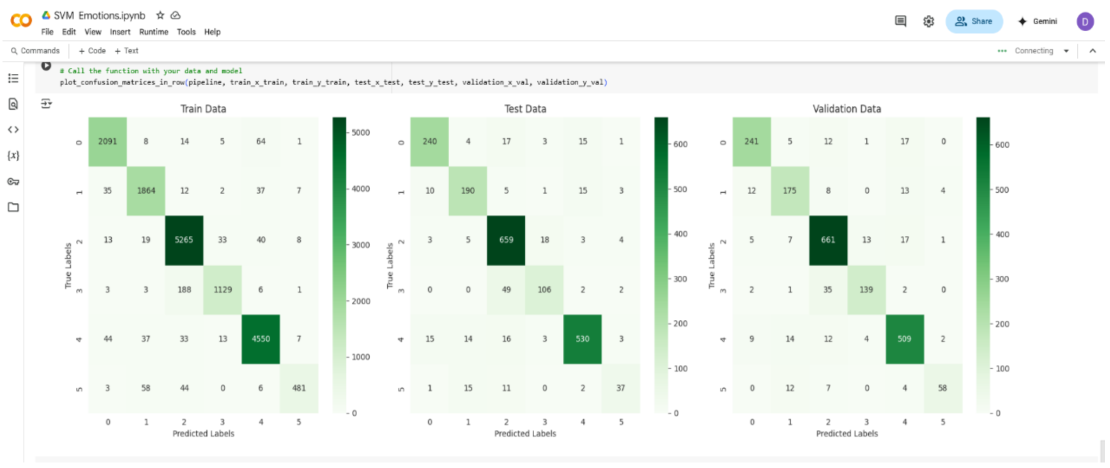
```
[11] from sklearn.svm import SVC
     pipeline = Pipeline([
         ('tfidf', TfidfVectorizer()),
         ('svm', SVC(kernel='linear', C=1.0, random_state=42, probability=True))
     ])

     pipeline.fit(train_x_train, train_y_train)


     train_score = pipeline.score(train_x_train, train_y_train)
     test_score = pipeline.score(test_x_test, test_y_test)
     validation_score = pipeline.score(validation_x_val, validation_y_val)

     print(f"\nFinal Accuracies:")
     print(f"Training Accuracy: {train_score:.4f}, Test Accuracy: {test_score:.4f}, Validation Accuracy: {validation_score:.4f}\n")

     joblib.dump(pipeline, 'svm_model.pkl')  # Save the model
     # Access the TfidfVectorizer object from the pipeline using named_steps
     tfidf_vectorizer = pipeline.named_steps['tfidf']
     joblib.dump(tfidf_vectorizer, 'vectorizer.pkl') # Save the vectorizer
```

```

Final Accuracies:
Training Accuracy: 0.9539, Test Accuracy: 0.8801, Validation Accuracy: 0.8906

['vectorizer.pkl']
```

```
tfidf_vectorizer = pipeline.named_steps['tfidf']
num_features = len(tfidf_vectorizer.vocabulary_)
print(f"Number of features: {num_features}")
```

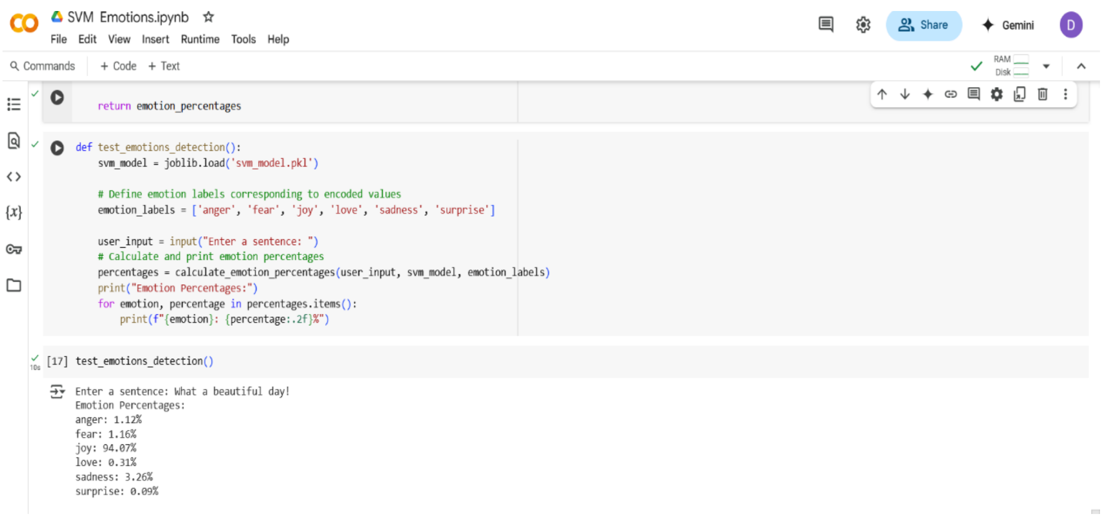
```
Number of features: 15214
```

- Confusion Matrix:



```
# Call the function with your data and model
plot_confusion_matrices_in_row(pipeline, train_x_train, train_y_train, test_x_test, test_y_test, validation_x_val, validation_y_val)
```

- Testing the Model:



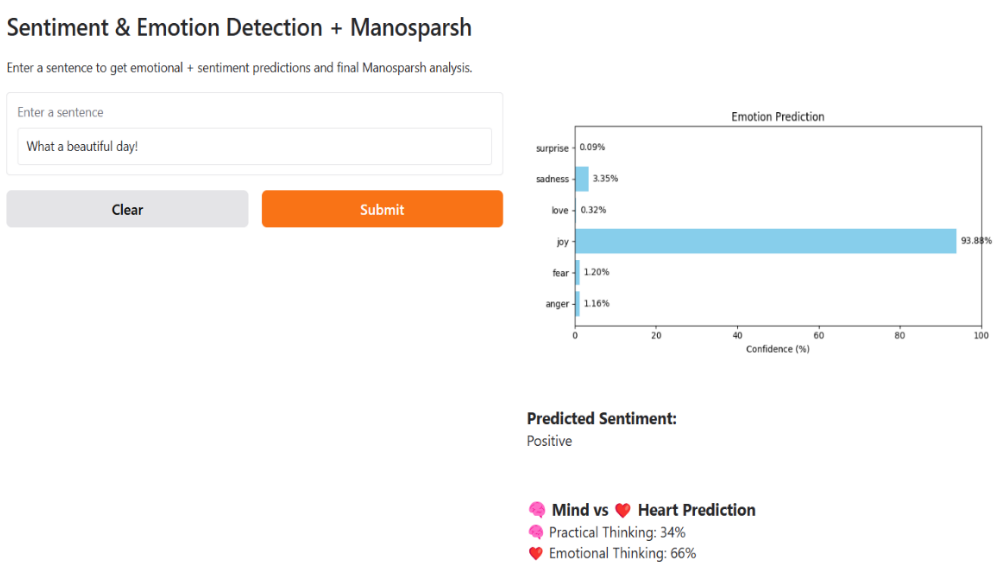
```
return emotion_percentages

def test_emotions_detection():
    svm_model = joblib.load('svm_model.pkl')

    # Define emotion labels corresponding to encoded values
    emotion_labels = ['anger', 'fear', 'joy', 'love', 'sadness', 'surprise']

    user_input = input("Enter a sentence: ")
    # Calculate and print emotion percentages
    percentages = calculate_emotion_percentages(user_input, svm_model, emotion_labels)
    print("Emotion Percentages:")
    for emotion, percentage in percentages.items():
        print(f"{emotion}: {percentage:.2f}%")
```

```
[17] test_emotions_detection()
```

```
Enter a sentence: What a beautiful day!
Emotion Percentages:
anger: 1.12%
fear: 1.16%
joy: 94.07%
love: 0.31%
sadness: 3.26%
surprise: 0.09%
```

Manosparsh:

- Deployment: **User Interfaces**



Sentiment & Emotion Detection + Manosparsh

Enter a sentence to get emotional + sentiment predictions and final Manosparsh analysis.

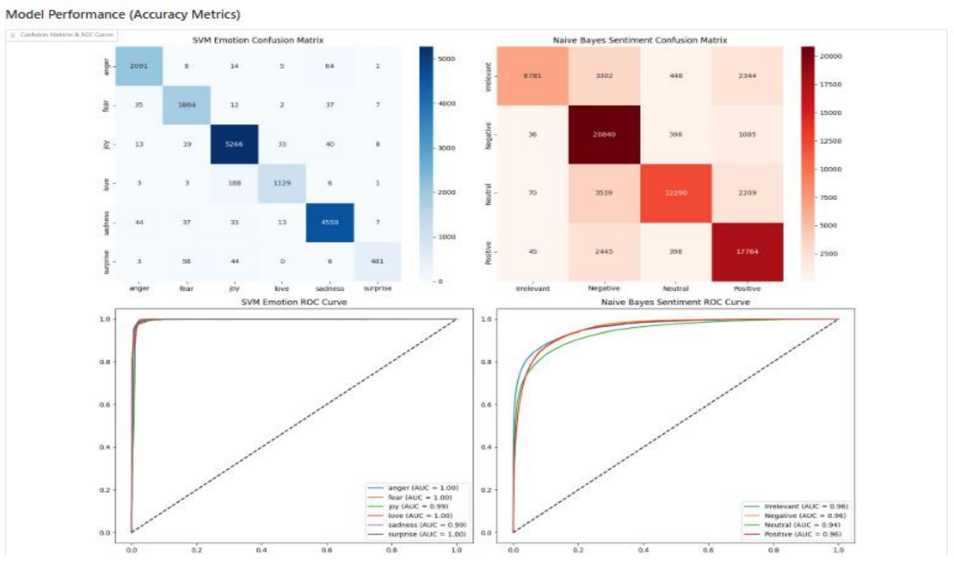
Enter a sentence

What a beautiful day!

Predicted Sentiment:
Positive

🧠 **Mind vs ❤️ Heart Prediction**
🧠 Practical Thinking: 34%
❤️ Emotional Thinking: 66%

- Model Performance:



SVM Emotion Detection - Classification Report

```
              precision    recall  f1-score   support

       anger       0.96      0.96      0.96      2183
        fear       0.94      0.95      0.94      1957
         joy       0.95      0.98      0.96      5379
        love       0.96      0.85      0.90      1330
     sadness       0.97      0.97      0.97      4684
    surprise       0.95      0.81      0.88       592

    accuracy                           0.95     16125
   macro avg       0.95      0.92      0.93     16125
weighted avg       0.95      0.95      0.95     16125
```

Naive Bayes Sentiment Analysis - Classification Report

```
              precision    recall  f1-score   support

  Irrelevant       0.98      0.53      0.68     12875
    Negative       0.69      0.93      0.79     22359
     Neutral       0.91      0.68      0.78     18108
    Positive       0.76      0.86      0.81     20652

    accuracy                           0.78     73994
   macro avg       0.83      0.75      0.77     73994
weighted avg       0.81      0.78      0.77     73994
```

Appendices

Appendix A: Technologies and Tools Utilized

- Programming Language: Python
- Development Environment: Google Colab and Jupyter Notebook
- Python Libraries:
  - scikit-learn – Used for training and testing machine learning models
  - Gradio – Helps in building an interactive user interface for model interaction
  - pandas and numpy – For data processing and manipulation
  - matplotlib, seaborn – To visualize results graphically
  - joblib – For saving and loading machine learning models
- Version Control and Deployment Platform:
  - GitHub – For source code and version control
  - Hugging Face Spaces – For online deployment using Gradio

Appendix B: Dataset Summary

- Emotion Recognition Dataset:
  - Format: Plain text file (.txt)
  - Contains text samples with associated emotional labels
  - Target emotions: joy, sadness, anger, fear, love, surprise, neutral
- Sentiment Analysis Dataset:
  - Source: Public dataset based on Twitter data
  - Format: CSV (.csv)
  - Relevant columns: sentence (text) and sentiment label (sentiment)
  - Sentiment categories: positive, negative, and irrelevant

Appendix C: Project Directory Structure

```
ManoSparsh/
├── app.py
├── README.md
├── SVM_Emotions.zip
├── nb_s.zip
├── extracted_files/
│   ├── SVM_Emotions/
│   │   ├── train.txt
│   │   └── SVM_Emotions_model.pkl
│   └── nb_s/
│       ├── twitter_training.csv
│       └── nb_s_model.pkl
```

Appendix D: Application Features

- Single Sentence Analysis:
  - User can input any sentence in a text box
  - Outputs include:
    - Bar graph displaying emotional probabilities
    - Sentiment classification
    - Analysis of decision tendency as either emotionally or rationally driven

- Batch Analysis:
    - Accepts a CSV file with multiple rows of text
    - Returns a detailed table showing:
    - Emotion probabilities
    - Sentiment outcome
    - Heart vs. Mind percentage (ManoSparsh analysis)

Appendix E: Example ManoSparsh Output

User Input: *"I know I should walk away, but I still hope they'll change."*

- Identified Emotion: Sadness (78.2%)
- Sentiment: Negative
- Decision Analysis (ManoSparsh):
    - 💝 Emotional Response: 88%
    - □ Rational Response: 12%

Citations

Sentiment analysis

Paper Title	Authors	Dataset Used	Model Used	Conclusion	Limitation
Techniques and applications for sentiment analysis	Feldman, R. (2013)	Generalized	Naive Byes and SVM	Importance of sentiment analysis	Limited focus on approaches.
CNN for situations understanding based on sentiment analysis of twitter data	Liao, S (2017)	Twitter database	CNN	CNNs are more effective than the traditional based approaches	Focuses on single model
Sentiment Analysis Using Deep Learning Approaches	Medhat et al. (2020)	Amazon Reviews	LSTM, GRU	Demonstrated the effectiveness of GRU and LSTM for large-scale sentiment classification.	Limited analysis of contextual understanding.
Improving Sentiment Analysis for Social Media Applications Using an Ensembled Deep Learning	Ahmed Alsayat	Multiple social media datasets	Machine learning And deep learning models	Deep learning models are more efficient than machine learning models	Lack of specific dataset used and clarity.
Contextual Sentiment Detection Using BERT	Devlin et al. (2019)	SST-2, Amazon Reviews	BERT	BERT's contextual embeddings outperformed traditional ML models in sentiment detection tasks.	Requires high memory and computational power.

Emotion detection

Research Paper Title	Authors	Dataset Used	Models Explored	Conclusion	Limitations
A technical review on emotion detection on informal text	Patil et al. (2019)	Twitter data	Naive Bayes, SVM	SVM outperforms Naive Bayes for emotion classification.	Dataset size and ambiguity in social media context.
Emotion Classification Using Machine Learning Approaches	Yasseri et al. (2014)	Twitter data	Naive Bayes, SVM, Logistic Regression	SVM achieves higher accuracy for emotion classification compared to Naive Bayes.	Small dataset, social media context challenges.
A Survey on Emotion Detection from Text using Deep Learning	Zhou et al. (2020)	Textual data from multiple domains (social media, movie reviews)	CNN, LSTM, Transformer-based models	LSTM and Transformers perform better than machine learning methods.	Challenges with sarcasm, irony, and long-context understanding.

References

Feldman, R. (2013). Techniques and analysis. Communications of the ACM, 56(4), 82-89.applications for sentiment

Liao, S., Wang, J., Yu, R., Sato, K., & Cheng, Z. (2017). CNN for situations understanding based on sentiment analysis of twitter data. Procedia computer science, 111, 376-381.

Alsayat, A. (2022). Improving sentiment analysis for social media applications using an ensemble deep learning language model. Arabian Journal for Science and Engineering, 47(2), 2499-2511.

Habimana, O., Li, Y., Li, R., Gu, X., & Yu, G. (2020). Sentiment analysis using deep learning approaches: an overview. Science China Information Sciences, 63, 1-36.

Chatterjee, A., Narahari, K. N., Joshi, M., & Agrawal, P. (2019, June). SemEval-2019 task 3: EmoContext contextual emotion detection in text. In Proceedings of the 13th international workshop on semantic evaluation (pp. 39-48).

Zhou, X., Wan, X., & Xiao, J. (2016, November). Attention-based LSTM network for cross-lingual sentiment classification. In Proceedings of the 2016 conference on empirical methods in natural language processing (pp. 247-256).

Patil, K., Nasriwala, J., & Savant, R. A Technical Review on Emotion Detection from Informal Text: Approaches and Challenges.

Cao, L., Peng, S., Yin, P., Zhou, Y., Yang, A., & Li, X. (2020, December). A survey of emotion analysis in text based on deep learning. In 2020 IEEE 8th International Conference on Smart City and Informatization (iSCI) (pp. 81-88). IEEE.

Bhagat, A., Sharma, A., & Chettri, S. (2020). Machine learning based sentiment analysis for text messages. International Journal of Computing and Technology.

Sanjana, S., Vibhakar, A., Singh, O., Raina, P., Singh, R., & Kumar Mall, P. (2023). A Study on the Use of Machine Learning Techniques for Text Sentiment Analysis. Kilby, 100, 7th.

Yohanes, D., Putra, J. S., Filbert, K., Suryaningrum, K. M., & Saputri, H. A. (2023). Emotion Detection in Textual Data using Deep Learning. Procedia Computer Science, 227, 464-473.

Deng, J., & Ren, F. (2021). A survey of textual emotion recognition and its challenges. IEEE Transactions on Affective Computing, 14(1), 49-67.

Acheampong, F. A., Wenyu, C., & Nunoo-Mensah, H. (2020). Text-based emotion detection: Advances, challenges, and opportunities. Engineering Reports, 2(7), e12189.

Salam, S. A., & Gupta, R. (2018). Emotion detection and recognition from text using machine learning. Int. J. Comput. Sci. Eng, 6(6), 341-345.

Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. Social network analysis and mining, 11(1), 81.

Banothu, S., Akula, S., Akarapu, V., & Rao, T. R. L. (2021, July). Emotion extraction and classification from twitter text. In Proceedings of the International Conference on IoT Based Control Networks & Intelligent Systems-ICICNIS.

Rana, R., & Kolhe, V. (2015). Analysis of students' emotion for twitter data using Naïve Bayes and non-linear support vector machine Approachs. Int. J. Recent Innov. Trends Comput. Commun, 3(5), 3211-3217.

Saxena, A., Khanna, A., & Gupta, D. (2020). Emotion recognition and detection methods: A comprehensive survey. Journal of Artificial Intelligence and Systems, 2(1), 53-79.