**Nlp**

the Natural Language Processing (NLP) tasks focus on text preprocessing and feature extraction for training a machine learning model. The specific NLP library used for text preprocessing is Neattext (neattext.functions)

The **remove\_userhandles** function from Neattext is used to eliminate user handles (e.g., @username) from the text. This is a common preprocessing step in NLP to focus on the content rather than specific users.

The **remove\_stopwords** function from Neattext is applied to remove common stopwords (e.g., "the," "and," "is") from the text. Removing stopwords is a standard text preprocessing step to focus on significant words and improve the efficiency of machine learning models.

The machine learning model itself is a pipeline that uses scikit-learn's **CountVectorizer** for feature extraction and a logistic regression classifier for predicting emotions

In summary, Neattext is employed for basic text preprocessing tasks, and scikit-learn is used for traditional machine learning-based NLP tasks in this project.

**Dataset**

Exploratory data analysis (EDA) is performed to understand the distribution of emotions in the dataset. Data preprocessing includes removing user handles and stopwords to clean the text data before training the model.

**Model(s)**

The project employs a machine learning model based on a logistic regression classifier. The model is trained to predict emotions in text using the labeled dataset. The serialized model is saved as "text\_emotion.pkl" for easy deployment and reuse in the Streamlit app.

This model is a traditional machine learning model, and it does not involve neural networks. It leverages classic NLP techniques, such as Bag-of-Words, for feature representation and logistic regression for classification.

**Pandas** (import pandas as pd):

Function: Pandas is a powerful data manipulation library. In this project:

pd.read\_csv("emotion\_dataset\_raw.csv"): Reads a CSV file into a Pandas DataFrame.

df.head(): Displays the first few rows of the DataFrame.

df['Emotion'].value\_counts(): Counts the occurrences of each unique value in the 'Emotion' column.

df['Clean\_Text'] = ...: Creates a new column 'Clean\_Text' in the DataFrame based on text preprocessing.

NumPy (import numpy as np):

Function: NumPy is a numerical computing library.

np.max(probability): Calculates the maximum value in an array.

Seaborn (import seaborn as sns):

Function: Seaborn is a data visualization library.

sns.countplot(x='Emotion', data=df): Creates a bar plot to visualize the distribution of emotions.

Neattext (import neattext.functions as nfx):

Function: Neattext provides text preprocessing functions.

nfx.remove\_userhandles(text): Removes user handles (e.g., @username) from text.

nfx.remove\_stopwords(text): Removes common stopwords from text.

Scikit-learn (from sklearn...):

Function: Scikit-learn is a machine learning library.

CountVectorizer: Converts a collection of text documents to a matrix of token counts.

Pipeline: Chains together a sequence of data processing and modeling steps.

train\_test\_split: Splits data into training and testing sets.

RandomForestClassifier, LogisticRegression: Machine learning models used for classification tasks.

Joblib (import joblib):

Function: Joblib is used for serialization (saving and loading) of machine learning models.

joblib.dump(pipe\_lr, pipeline\_file): Saves the trained model to a file.

joblib.load(open("model/text\_emotion.pkl", "rb")): Loads the trained model from a file.

Streamlit (import streamlit as st):

Function: Streamlit is a web app framework for turning data scripts into shareable web apps.

st.title, st.subheader: Displays titles and subheaders in the app.

st.form, st.text\_area, st.form\_submit\_button: Creates a form and text area in the app.

st.success, st.write: Displays success messages and outputs in the app.

st.altair\_chart: Displays Altair visualizations in the app.

Altair (import altair as alt):

Function: Altair is a declarative statistical visualization library.

alt.Chart: Creates a chart object.

mark\_bar, encode: Specifies the type of chart and its encoding.

* + - **LogisticRegression** is a linear model for binary and multiclass classification.
    - Despite its name, it's commonly used for classification tasks. It estimates the probability that a given instance belongs to a particular class and assigns the class with the highest probability.

Both algorithms are part of a machine learning pipeline (**Pipeline**) that includes a text vectorization step (**CountVectorizer**) to convert text data into a format suitable for machine learning models 0.6200421536692853

**Description:**

* The **RandomForestClassifier** is an ensemble learning method that constructs a multitude of decision trees at training time.
* It uses bagging (Bootstrap Aggregating) to build multiple decision trees and combines their predictions to improve accuracy and control overfitting.
* 0.5200421536692853

from sklearn.model\_selection import train\_test\_split 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

ChatGPT

**ChatGPT**

The code you provided includes several components from the scikit-learn library, which are commonly used for building and evaluating machine learning models. Here's a brief explanation of each component:

1. **train\_test\_split from sklearn.model\_selection:**
   * **Function:**
     + Splits the dataset into training and testing sets.
   * **Usage:**

pythonCopy code

from sklearn.model\_selection import train\_test\_split x\_train, x\_test, y\_train, y\_test = train\_test\_split(features, labels, test\_size=0.3, random\_state=42)

1. **Pipeline from sklearn.pipeline:**
   * **Function:**
     + Chains multiple steps into a single unit, allowing for a streamlined workflow.
   * **Usage:**

pythonCopy code

from sklearn.pipeline import Pipeline pipeline = Pipeline(steps=[('step1', component1), ('step2', component2), ...])

1. **CountVectorizer from sklearn.feature\_extraction.text:**
   * **Function:**
     + Converts a collection of text documents to a matrix of token counts (Bag-of-Words representation).
   * **Usage:**

pythonCopy code

from sklearn.feature\_extraction.text import CountVectorizer vectorizer = CountVectorizer()

1. **SVC (Support Vector Classifier) from sklearn.svm:**
   * **Function:**
     + Implements the Support Vector Machine (SVM) algorithm for classification tasks.
   * **Usage:**

pythonCopy code

from sklearn.svm import SVC svc\_classifier = SVC()

1. **RandomForestClassifier from sklearn.ensemble:**
   * **Function:**
     + Implements the Random Forest algorithm, an ensemble method for classification and regression tasks.
   * **Usage:**

pythonCopy code

from sklearn.ensemble import RandomForestClassifier rf\_classifier = RandomForestClassifier(n\_estimators=100)

1. **LogisticRegression from sklearn.linear\_model:**
   * **Function:**
     + Implements logistic regression, a linear model for binary and multiclass classification.
   * **Usage:**

pythonCopy code

from sklearn.linear\_model import LogisticRegression lr\_classifier = LogisticRegression()

These components are often used together in a

In the provided code snippets, the **transformers** library is not explicitly used. The project appears to rely on traditional machine learning techniques and the scikit-learn library for the text classification task. Specifically, it uses a pipeline that includes a **CountVectorizer** for text vectorization and a classifier (either **LogisticRegression**, **SVC**, or **RandomForestClassifier**).

Research paper

https://arxiv.org/abs/1906.09774

<https://www.researchgate.net/publication/333994024_Emotionally-Aware_Chatbots_A_Survey>

**conclusion**

This paper addresses an important and less examined area of sentiment research, that is, emotion identification from text. The major contribution of this work is to show that it is practical to apply computational methods to identify and distinguish various types of emotions in text. The goal of the emotion identification experiment is to manually add emotion information to each sentence in a dataset of blogs collected from the web or standard dataset. This manually annotated data can be used to train computer based systems to automatically identify emotion information on a large-scale. This paper is mainly focused on automatic emotion detection and also suggested that to achieve good performance, it is important to include a wide variety of words that go away from the conventional emotion words. This study also introduced a novel approach of automatically building emotion lexicon utilizing the classification system. In the approach introduced here, a variety of emotion-related words were learned, and their usability demonstrated by their effectiveness in VSM classification methods for emotion detection. In Future, This system should also detect not only the existence of keywords, but also their linguistic information to detect emotions more accuratel