**Project Overview**

In this section, we present the implementation of a machine learning approach to classify textual data based on the emotions expressed in the text. The goal of this project is to categorize text into predefined emotion labels (such as **sadness, joy, love, anger, fear,** and **surprise**) using the **XGBoost** classifier, a popular and efficient machine learning model known for its high performance in classification tasks. The model is trained on a dataset consisting of textual descriptions paired with corresponding emotion labels.

**Data Preparation and Preprocessing**

The dataset used in this project is derived from three different files, which were combined to form a single dataset with 436,806 rows and 2 columns. The dataset was sourced from Kaggle (links provided below):

* <https://www.kaggle.com/competitions/tweet-sentiment-extraction/data>
* <https://www.kaggle.com/datasets/nelgiriyewithana/emotions/data>
* <https://github.com/Mr-Appu/Text-Based-Emotion-Recognition/blob/main/Datasets/data.txt>

The dataset contains two key columns: 'Text' and 'Emotion.' The 'Text' column includes sentences or phrases, while the 'Emotion' column specifies the emotional label associated with each text sample.

To prepare the data for model training, the following steps were undertaken:

1. **Loading the Dataset**: The dataset was loaded from a CSV file.The text data (X) was extracted from the Text column, while the emotion labels (y) were extracted from the Emotion column.
2. **Label Encoding**: The emotion labels are categorical in nature, and machine learning models typically require numerical inputs. To address this, the Label Encoder class from Scikit-learn was employed to convert the emotion labels into numerical form. Each unique emotion (e.g., sadness, joy) was mapped to an integer value. For instance:
   * **Sadness** → 0
   * **Joy** → 1
   * **Love** → 2
   * **Anger** → 3
   * **Fear** → 4
   * **Surprise** → 5
3. **Train-Test Split**: To evaluate the performance of the model, the dataset was split into two sets: a training set (80% of the data) and a test set (20% of the data). This ensures that the model is trained on one portion of the data while being evaluated on a separate, unseen portion. The train\_test\_split function from Scikit-learn was used to perform this operation.

**Text Representation Using TF-IDF Vectorizer:**

To convert the textual data into a numerical form that can be interpreted by the machine learning model, the TF-IDF Vectorizer from Scikit-learn was used. This method transforms the text into a TF-IDF representation, where:

* Each unique word in the dataset becomes a feature.
* The value of each feature reflects the importance of the word in the corresponding text sample, calculated as the product of its term frequency (TF) and inverse document frequency (IDF).

This step results in a sparse matrix, where rows correspond to individual text samples, and columns represent the TF-IDF scores of the words. The fit\_transform method was applied to the training data, while the transform method was used for the test data.

**Model Selection: XGBoost Classifier**

For the classification task, we selected the **XGBoost** algorithm, a highly efficient and scalable implementation of gradient boosting that performs well on a variety of machine learning tasks. The reasons for selecting XGBoost include:

* **Performance**: XGBoost is known for its accuracy and speed, especially in tasks involving large datasets.
* **Handling Imbalanced Data**: XGBoost can be tuned to handle imbalanced datasets, which is important for emotion classification, where some emotions may be underrepresented.

In this project, the following settings were applied:

* The parameter use\_label\_encoder=False was set to prevent XGBoost’ s internal label encoder from conflicting with the Scikit-learn label encoding.
* The eval\_metric='mlogloss' was used to optimize for multi-class classification based on the logarithmic loss, which measures the difference between predicted probabilities and actual class labels.

The model was trained on the vectorized training data using the fit method, which optimized the classification model based on the labeled examples provided.

**Model Evaluation**

Once the model was trained, it was evaluated on the test set using several metrics. Predictions were generated on the test data using the predict method, and the following evaluation measures were applied:

* **Accuracy Score**: This metric was used to measure the overall performance of the model by calculating the proportion of correctly classified samples.
* **Classification Report**: A more detailed performance evaluation was conducted using the classification report, which provides the precision, recall, and F1-score for each emotion class. These metrics offer insights into how well the model performs on individual classes.

The XGBoost model achieved a notable accuracy of 0.92% on the test data, demonstrating its capability to classify emotions from textual data effectively. However, certain emotions—especially those with fewer samples—were less accurately predicted, which is consistent with the challenges posed by imbalanced datasets.

The confusion matrix provides a detailed breakdown of the model’s performance in classifying each emotion, revealing areas where the model struggled, such as misclassifications between emotions like "joy" and "love." The confusion matrix helps visualize the true positives, false positives, and false negatives for each emotion.

In addition to accuracy, the classification report further evaluates the model based on precision, recall, and F1-score. These metrics provide a comprehensive view of the classifier's performance:

* Precision: Measures the percentage of correct positive predictions for each emotion.
* Recall: Indicates the model’s ability to identify all actual instances of each emotion.
* F1-Score: The harmonic mean of precision and recall, offering a balance between the two.

Here is a summary of the model’s performance metrics:

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Accuracy | 0.92 |
| Precision (avg) | 93 |
| Recall (avg) | 92 |
| F1-Score (avg) | 92 |

|  |  |
| --- | --- |
|  |  |
|  |  |

The detailed classification report and confusion matrix further highlight the performance of the XGBoost model across various emotions, illustrating its strength in classifying more common emotions and pointing out the difficulties in distinguishing among rarer or similar emotions.

A screenshot of a graph

Description automatically generated

A diagram of a confused matrix

Description automatically generated

**Conclusion and Future Work**

In this project, we successfully implemented a text-based emotion classification system using the XGBoost algorithm. The system demonstrated strong performance in identifying the emotions expressed in text, with potential applications in sentiment analysis, mental health monitoring, and customer feedback analysis. The Following algorithms were also used.

|  |  |
| --- | --- |
| **Name Of Classifier** | **Accuracy** |
| XGboost | 0.92 |
| Naive Bayes | 0.75 |
| Random Forest | 0.91 |