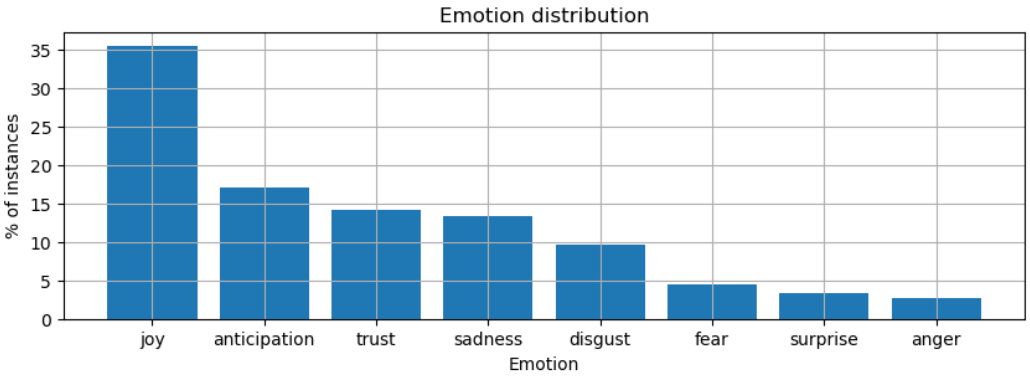
Part 3 Report

1. Exploring Data Analysis

Before analyzing the data to uncover potential relationships, I conducted an initial review of the dataset. The dataset comprises numerous tweets of varying lengths, and many tweets are not purely text–based, as they include emojis. Additionally, tweets often contain colloquial abbreviations or trendy expressions. Since the data was obtained through web scraping, many HTML tags were embedded within the text.

During this exploration, I examined the correlation between each tweet and its associated emotion label. One key observation was the significant class imbalance in the training data. Notably, tweets labeled as “joy” constituted a disproportionately larger portion compared to other labels. This imbalance could influence model performance, potentially biasing predictions toward the dominant class. The histogram below (Graph 1.) illustrates the distribution of emotion labels, clearly highlighting this imbalance.



Graph 1. Histogram of emotion labels

2. Pre–processing

Sentence standardizing:

- Emojis: All emojis were converted into their corresponding words using the demojize function from the emoji library.
- Spelling Correction: The spellchecker library was employed to correct misspelled words.
- Abbreviation Expansion: Common abbreviations often seen on Twitter were replaced with their original full forms.
- HTML Tags and Extra Punctuation: HTML tags and unnecessary punctuation were removed through substitution techniques.

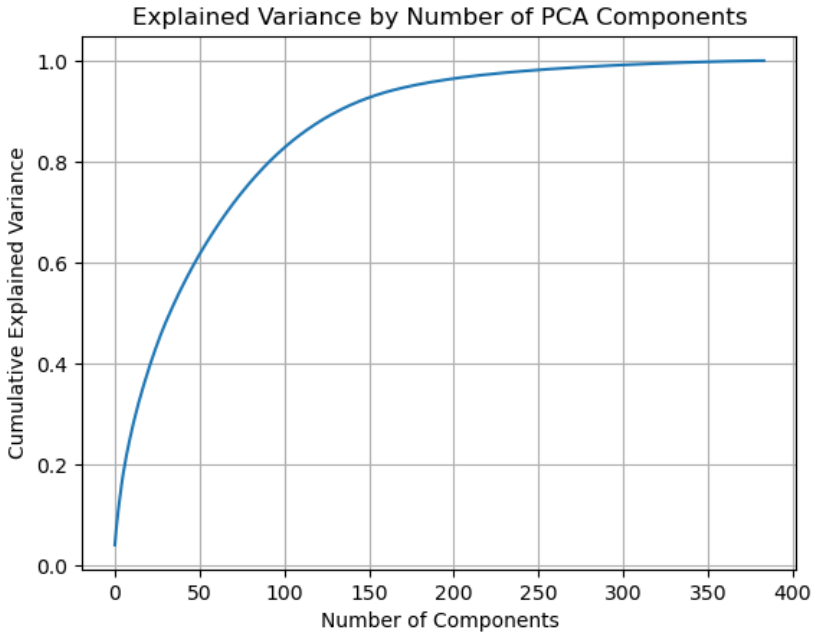
Extract emotion words:

Since the main goal of the competition is to recognize the emotion labels of each tweet, I observed that hashtag words could partially represent the sentiment of a text. By leveraging the hashtags in tweets that contained them, I trained a simple Decision Tree classifier. This approach achieved a validation accuracy of 0.53. However, since many tweets lacked hashtags, I supplemented the analysis with emotion–related features extracted using libraries like Afinn, VADER, and TextBlob. These tools helped identify words and phrases potentially linked to emotions, enriching the dataset for better model performance. Although

emotional word extraction helped reduce the data's sparsity, there were still many missing values. Therefore, in later analyses, I used the entire text content to ensure all information was retained.

3. Feature Engineering:

- TF–IDF: It was applied to convert the raw text into numerical features, capturing the importance of individual words relative to the entire corpus. This approach helped in identifying key terms that were significant for differentiating between emotion labels.
- Embedding: The paraphrase–mpnet–base–v2 model from Sentence Transformer was utilized to generate semantic embeddings for each tweet. These embeddings effectively captured contextual information and enhanced the model's ability to understand subtle nuances in the text. By transforming sentences into dense vector representations, this method contributed significantly to improving classification performance.
- PCA and UMAP: PCA and UMAP were applied to reduce the dimensionality of the embedding features. This step improved computational efficiency while retaining the most critical information for downstream classification tasks. Additionally, I used visualizations (Graph 2.) to determine the optimal dimensionality when applying PCA.

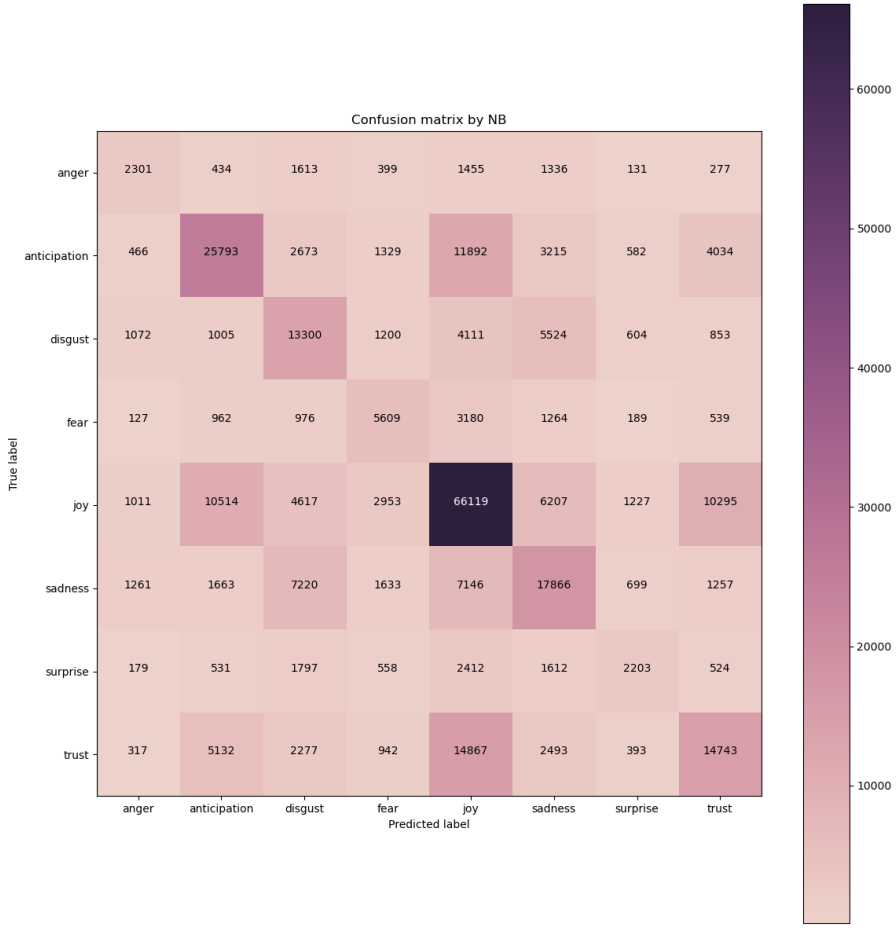


Graph 2. Cumulated Variance of different PCA degree

4. Model Experiments

1. NB classification: It is a probabilistic model based on Bayes' Theorem, which assumes that all features are independent given the class label.

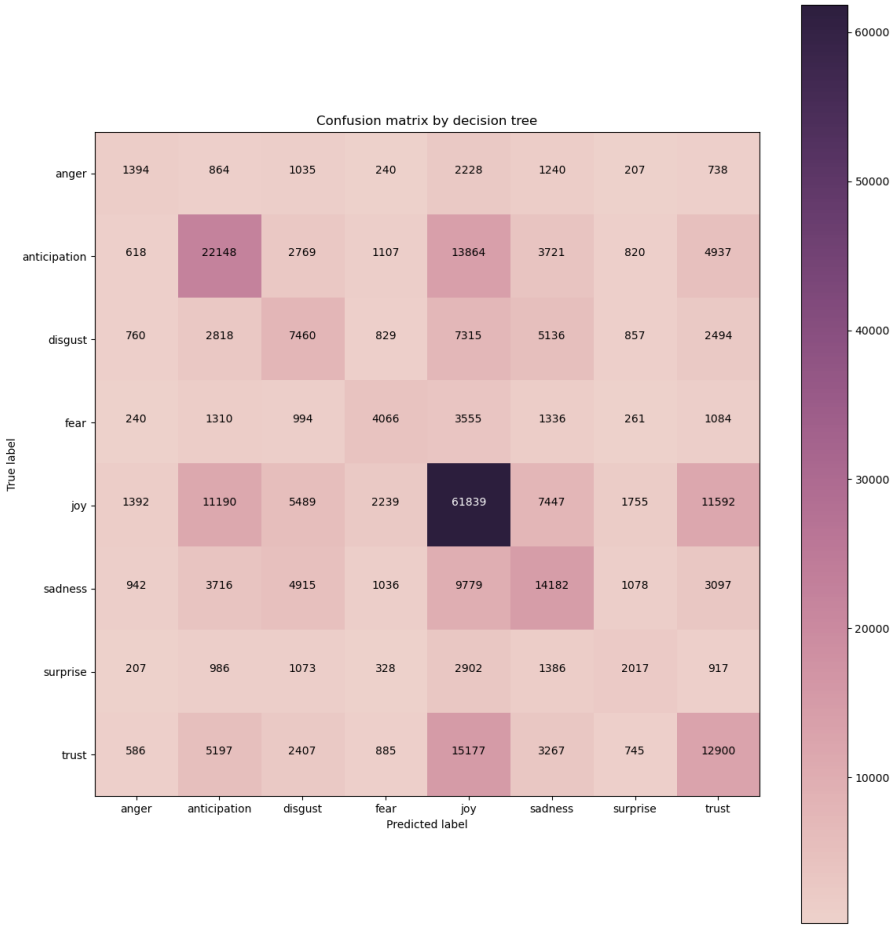
In this analysis, I used the TF–IDF matrix constructed from the entire text corpus as input to the NB classification model. The model achieved a training accuracy of 0.52 and a validation accuracy of 0.51. However, as shown in the confusion matrix below (Graph 3.), a significant proportion of predictions were misclassified as "joy." This result highlights the impact of class imbalance in the dataset, where the overrepresentation of the "joy" label influenced the model's predictions.



Graph 3. Confusion matrix of validation set using NB classifier

2. Decision Tree: It splits data based on certain conditions to make predictions. It is easy to interpret and works well when the relationship between features and the target is non–linear.

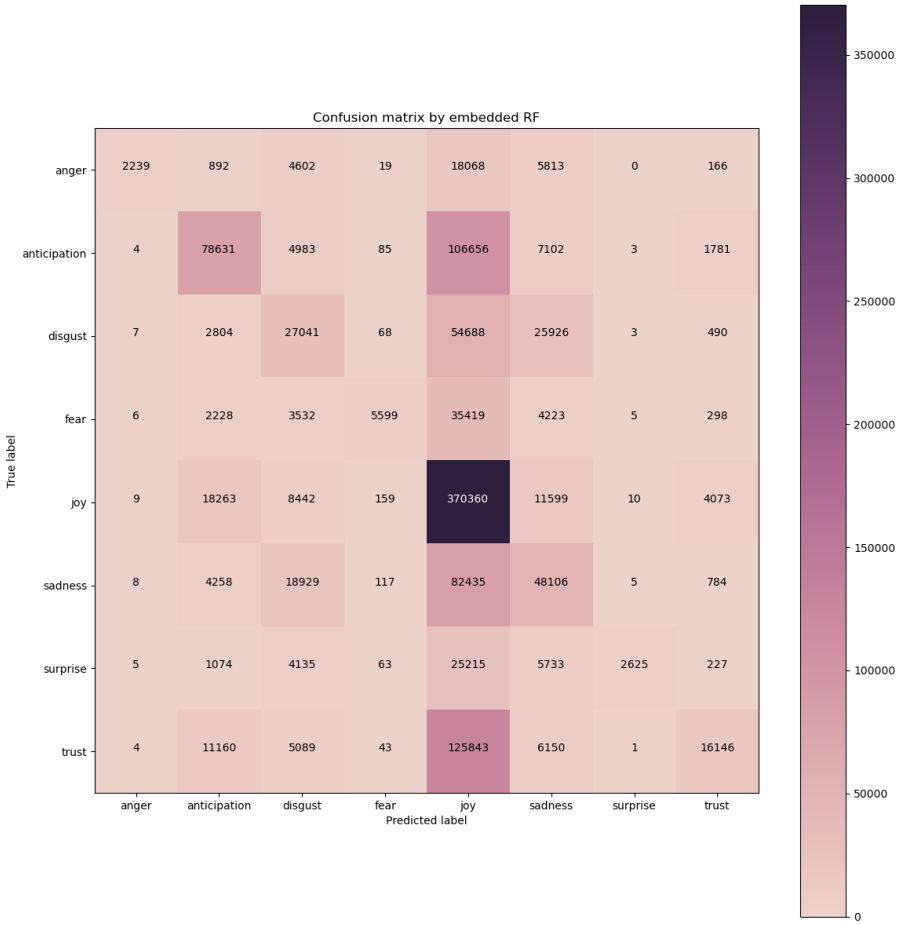
Using TF–IDF as input features, the Decision Tree model achieved a training accuracy of 0.99 and a validation accuracy of 0.43. The high training accuracy indicates that the model fits the training data very well, but the significantly lower validation accuracy suggests overfitting. The confusion matrix below(Graph 4.) highlights the issue of the imbalance training data with many validation data predicted as "joy."



Graph 4. Confusion matrix of validation set using decision tree

3. Random Forest: It iis an ensemble method that combines multiple Decision Trees to make more robust predictions. By averaging the outputs of multiple trees, it reduces overfitting and improves accuracy. Random Forest works particularly well with noisy data and when features are highly correlated.

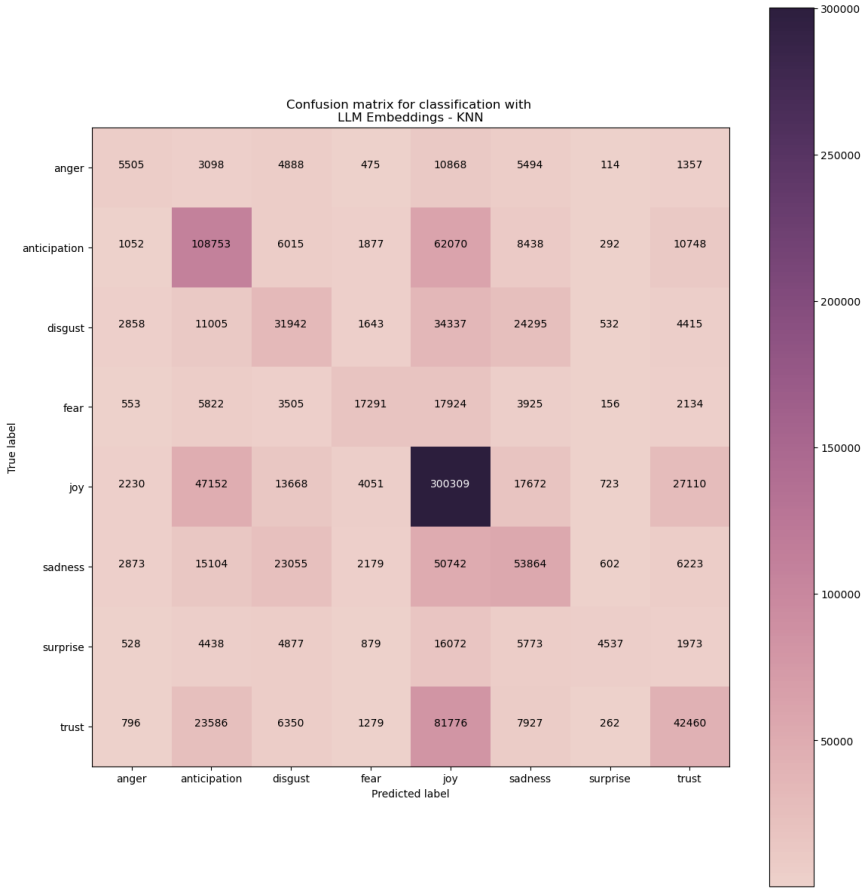
In this model I reduced the dimension into 100 degrees with PCA. By the cumulative variance graph of PCA we can know that 100 degree can capture about 80%. The confusion matrix below (Graph 5.) show that precision of anger is high, but the accuracy of “joy” low, lots of texts are predicted as joy. This may be due to the imbalanced training data, which caused the random forest trees to be biased towards 'joy,' leading to more predictions favoring ‘joy'.



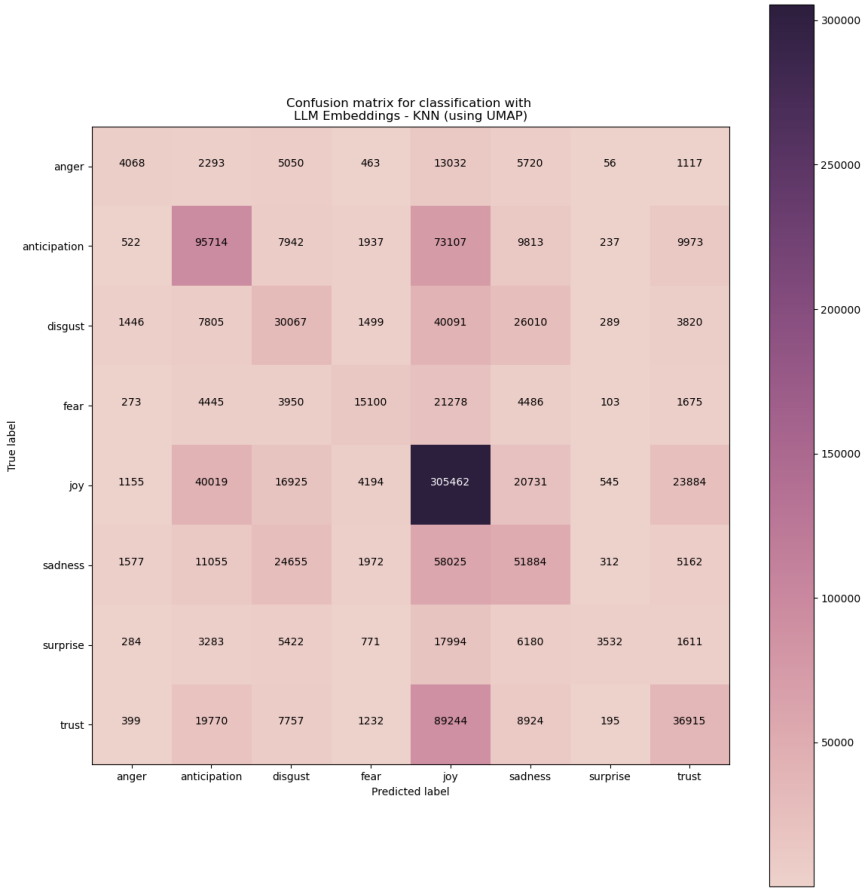
Graph 5. Confusion matrix of validation set using random tree

4. K–Nearest Neighbors: It predicts the label of a data point based on the majority label of its nearest neighbors. It performs well with small datasets and when the distance metric aligns well with the feature distribution. However, it is computationally expensive for large datasets and sensitive to the choice of k and feature scaling.

Using PCA, 150 dimensions were selected, capturing approximately 90% of the variance, and this achieved a validation accuracy of 0.487. Similarly, UMAP was employed to reduce the dimensionality to 130, achieving a slightly lower validation accuracy of 0.466. However, the confusion matrices (Graph 6. and Graph 7.) for both methods reveal a strong bias towards the “joy” class, which dominated the predictions due to the imbalanced nature of the dataset.



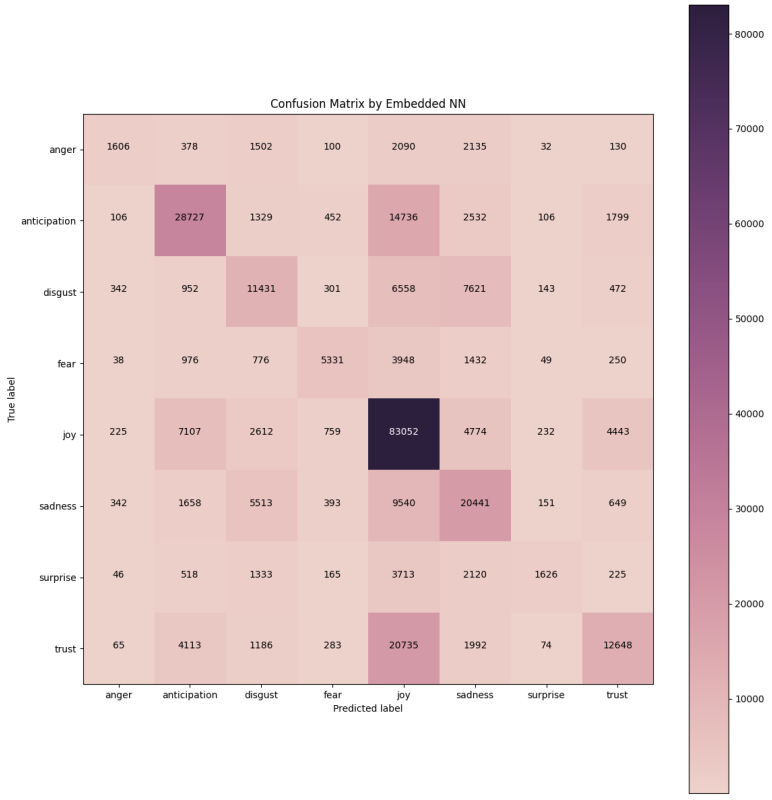
Graph 6. Confusion matrix of validation set using KNN with PCA (150 dimensions)



Graph 7. Confusion matrix of validation set using KNN with UMAP (130 dimensions)

5. Neural Network: It is a method that highly versatile and excel at capturing complex, non–linear relationships in data. However, Neural Networks require large amounts of data and computational power to perform optimally.

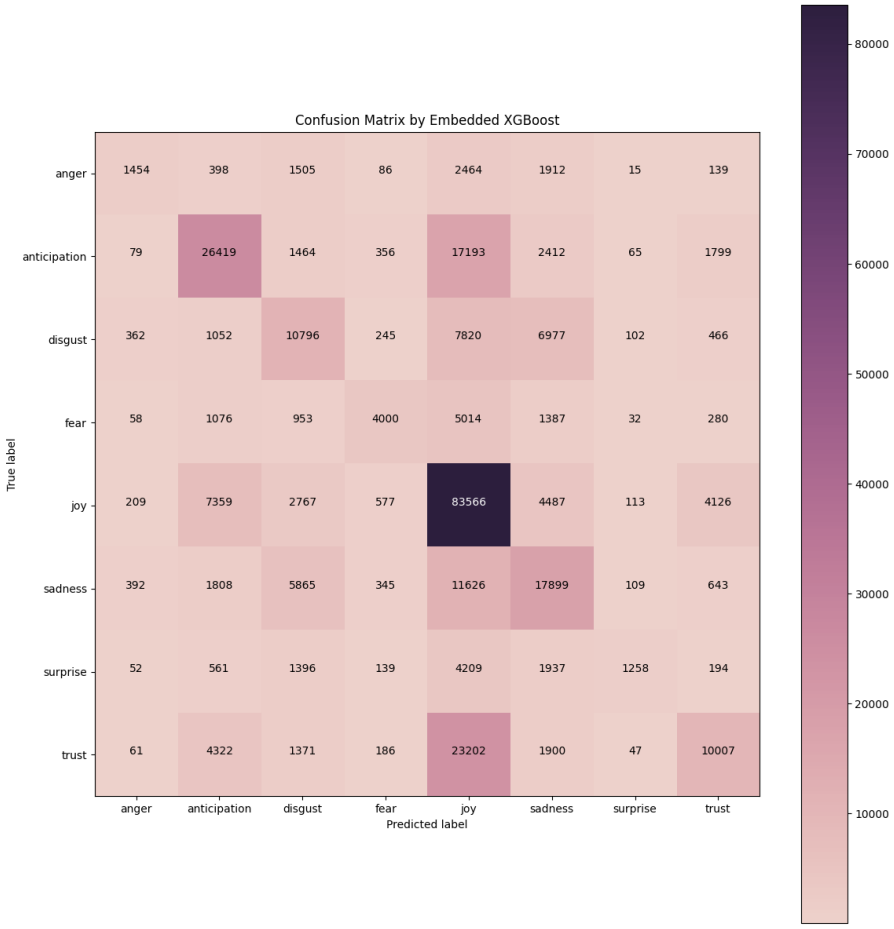
Using a Neural Network (NN) with embedded features, the model achieved a validation accuracy of 0.57. The confusion matrix (Graph 8.) reveals that the model performed well in classifying dominant labels like “joy” and “anticipation.” However, the results also indicate challenges in distinguishing minority classes such as “surprise” and “trust,” which had significantly lower recall and higher misclassification rates.



Graph 8. Confusion matrix of validation set using Neural Network

6. XGBoost: It is a gradient boosting algorithm known for its efficiency, scalability, and superior performance in structured data tasks. By iteratively improving upon errors made by previous models, it achieves high accuracy while reducing overfitting through techniques like regularization.

Using the embedded features generated from the dataset, XGBoost, a gradient boosting algorithm, achieved a validation accuracy of 0.5338, while the Kaggle test result was slightly lower at 0.46. The confusion matrix (Graph 9.) highlights that while the model performed well in predicting dominant classes such as “joy” and “anticipation,” with 83,566 and 26,419 correct predictions respectively, there was a noticeable bias toward these classes due to the dataset’s imbalanced nature. Minority classes like “trust” and “surprise” had lower recall, indicating challenges in capturing subtle distinctions within these categories.



Graph 9. Confusion matrix of validation set using XGBoost

7. DeBERTa: It is a transformer–based model designed to improve natural language understanding tasks. It uses two key methods: disentangled attention, which separates content and position information in the attention mechanism, and enhanced mask decoding, which improves training efficiency.

When applied to the cleaned text data, DeBERTa achieved a validation accuracy of 0.65, outperforming all other tested models. The confusion matrix (Graph 10.) shows significant improvements in classifying all emotion labels, with a notable reduction in the misclassification of the majority “joy” class. Minority classes such as “trust,” “surprise,” and “disgust” also saw substantial gains in prediction accuracy. DeBERTa’s advanced architecture enabled it to overcome the limitations faced by other models, offering both higher precision and recall across imbalanced classes, making it the most robust model for this dataset.

Graph 10. Confusion matrix of validation set using DeBERTa

5. Conclusion

In this analysis, I tried various models, from simpler ones like Naive Bayes and Decision Trees to more advanced ones like XGBoost and the transformer–based model DeBERTa. Each model had its own strengths and weaknesses, especially when dealing with the imbalanced dataset. Simpler models like Naive Bayes and Decision Trees often overfitted and predicted too many examples as the majority class, while models like Random Forest and XGBoost performed better overall but still struggled with the minority classes. Dimensionality reduction methods like PCA and UMAP helped improve computational efficiency and feature quality, which was especially useful for K–Nearest Neighbors.

Among all the models, DeBERTa performed the best, achieving a validation accuracy of 0.65. It was the only model that handled class imbalances well and accurately classified minority classes like “trust” and “surprise.” In my Kaggle submissions, I uploaded results from DeBERTa, Neural Network, and XGBoost. To optimize the final output, I plan to fine–tune the model parameters to achieve the best possible validation accuracy. After identifying the optimal parameters, I will retrain the model using all available data and then apply it to the test set to generate predictions for submission. While the validation scores were high for all these models, there was still a small difference in the Kaggle results. My best score on Kaggle was 0.54875, which came from DeBERTa (Graph 11.). This shows that advanced transformer models like DeBERTa are especially effective for emotion classification tasks, even with imbalanced data.

Graph 11. The result of DeBERTa on Kaggle