To begin with, the dataset used for this sentiment classification task contained 99 labeled samples, with a notable imbalance across the three sentiment categories: 45 neutral, 42 negative, and only 12 positive examples. The average word count per sample was approximately 43.44 words, with the text consisting of a total of 3,806 Bangla words and 434 transliterated Banglish words, giving a Bangla-to-Banglish word ratio of 8.77.

Moving on, as the dataset was too small I opted for note removing any banglish words from the dataset. In a super small dataset, every piece of information is valuable, and removing Banglish words may risk losing potentially important context or features that could aid the model in making accurate predictions. Additionally, several emojis were present in the dataset, which were mapped to their Bangla equivalents to improve sentiment analysis.

Furthermore, in the preprocessing phase, I performed a thorough cleaning of the text data by removing unnecessary elements like white spaces, special characters, numerical values, and punctuation in both Bangla and English. Stopwords were also eliminated to focus on the most meaningful parts of the text. A key part of the preprocessing involved mapping emojis to their Bangla translations, such as '🙏' to "অনেক ধন্যবাদ" and '❤' to "ভালবাসা" , which helped in preserving the sentiment conveyed by these symbols. Furthermore, I applied Bangla stemming using the Bangla-stemmer library to reduce words to their root forms, ensuring consistency in the feature space. Stemming is a text preprocessing technique used in natural language processing (NLP) to reduce words to their root or base form by stripping off suffixes, prefixes, or other affixes. The goal of stemming is to group together different forms of the same word so that they can be treated as a single item in text analysis.

Then I generated word cloud to see what types of words are most frequent in my dataset  


After seeing the word cloud I can visualize that See translation is present in the dataset then I removed the english word "See translation" using regular expression. Because there is likely some noise from scraping. After that I saved the remainder of data named "**Preprocess\_dataset.csv"**.

Moreover, for feature extraction, I opted for FastText, a model well-suited to handling the grammatical complexity of mixed-language text like Bangla and Banglish. FastText captures subword information and more focused grammatical syntax rather than semantic meaning of a sentence, making it more effective than other embedding methods such as Word2Vec or TF-IDF for this particular dataset.

Then the next challenge I faced was that the dataset is highly imbalanced. To address this, I employed SMOTE (Synthetic Minority Over-sampling Technique), which helped balance the class distribution by generating synthetic samples for the minority classes. SMOTE helps by generating synthetic examples for the minority class rather than simply duplicating existing examples. It works by selecting a data point from the minority class, identifying its nearest neighbors, and then creating new data points that are combinations of the selected data point and its neighbors. This process helps the model learn more about the minority class, leading to better overall performance and reducing bias toward the majority class. SMOTE is highly effective for traditional machine learning techniques, like SVMs, Decision Trees, and Logistic Regression, because these models rely heavily on balanced datasets to learn patterns from the features. I tried using CNN, LSTM, RNN and even BiLSTM but they rely heavily on semantics, especially when dealing with sequential data such as text or speech. Thus struggling to fully exploit semantic dependencies due to limited examples and low quality dataset with Bangla and Banglish words are present, resulting in underperformance.

However, SMOTE is not particularly useful for large language models (LLMs) like LLaMA or GPT. LLMs are trained on massive amounts of data and can capture intricate patterns, relationships, and contextual meanings across vast linguistic spaces. These models excel at learning from contextual clues, even in imbalanced datasets, due to their deep architecture and extensive pre-training. Instead of relying on synthetic data points like SMOTE generates, LLMs benefit more from augmentation techniques that preserve the linguistic and contextual integrity of the data, such as back-translation.

So that's why I used back-translation, a technique that expands the dataset by translating text to another language and back again to generate new contextual examples.

By utilizing googletrans library I was able to translate bangla texts to English then translate it back to the original language. When using it I noticed googletrans library was able to convert Banglish words to English then translating back to bangla again.

After back translation I got about:

* positive 45
* neutral 45
* negative 45

classes.

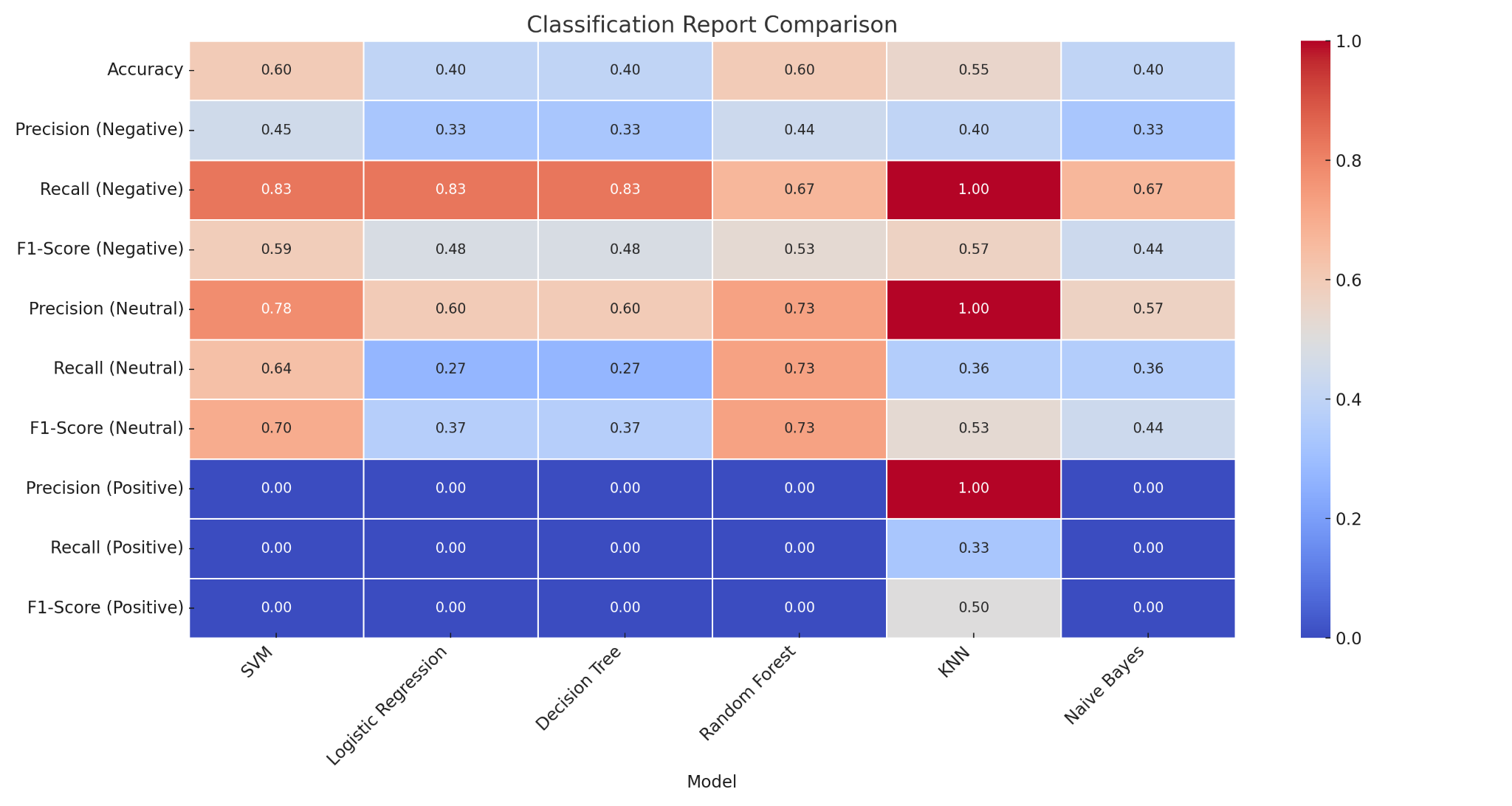
I saved this new preprocessed dataset as a CSV format and named it **balanced\_dataset\_v1.csv**.

After that, I started by loading this dataset. To ensure a good mix of samples, I shuffled the data and split it into training (80%), evaluation (10%), and test (10%) sets. This split helps in training the model, validating its performance during training, and finally testing it on unseen data. I created custom prompts for the model. For training and evaluation, I included both the text and the sentiment label. For testing, I only included the text, leaving the label blank for the model to predict. This approach frames the task as a text completion problem which is well-suited for a transformer based model. I used 4-bit quantization using the BitsAndBytes library. I used the PEFT (Parameter-Efficient Fine-Tuning) library with LoRA (Low-Rank Adaptation). I targeted specific modules in the model for fine-tuning, focusing on key components like attention projections and feed-forward layers. I set up the training arguments with careful consideration Small batch size (1) with gradient accumulation (8 steps) to cover up my small dataset.

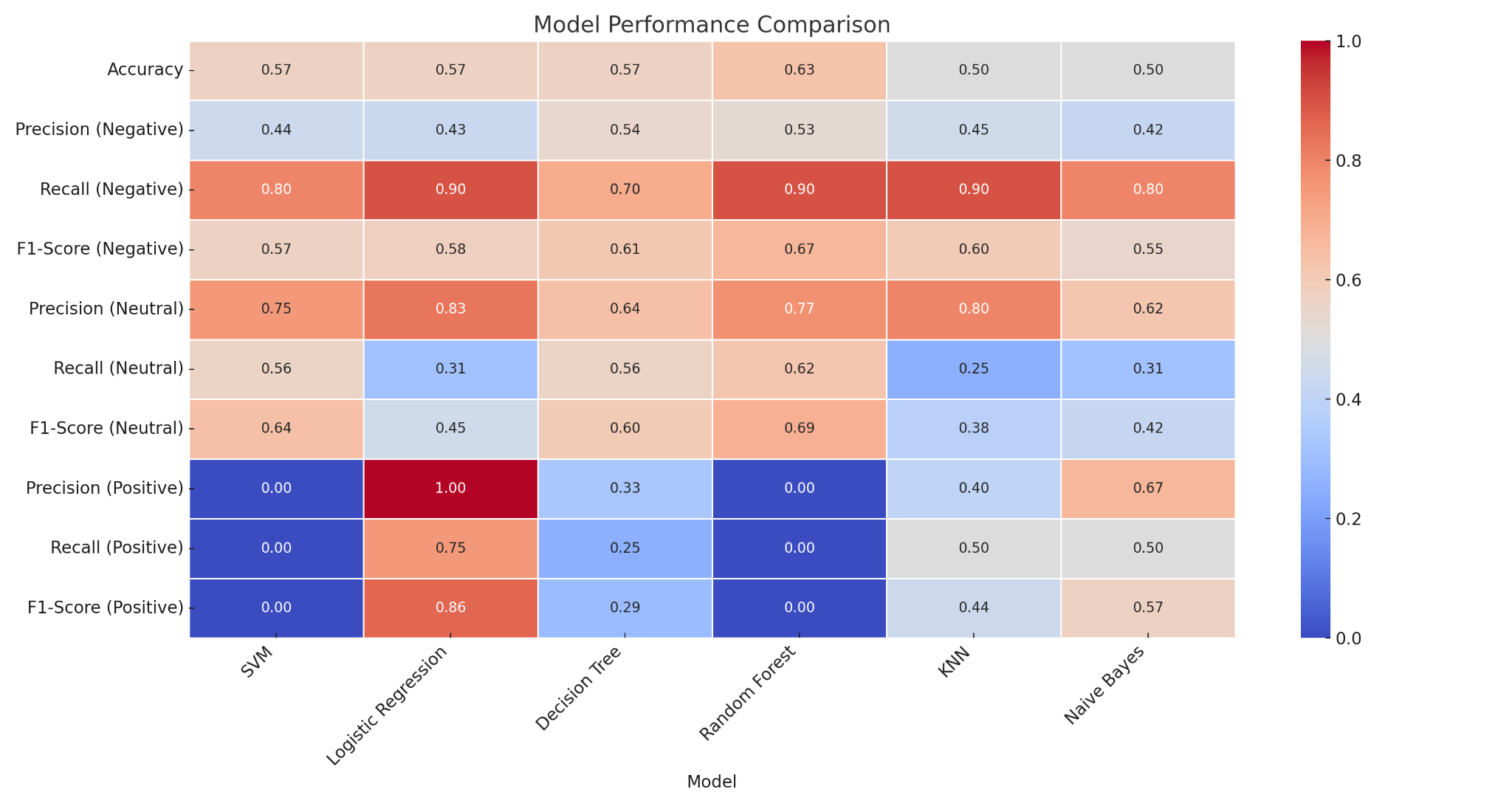
Finally, I used the SFTTrainer from the TRL (Transformer Reinforcement Learning) library for supervised fine-tuning. After training, I evaluated the model on the test set, using custom functions to generate predictions and compute metrics like accuracy, classification report, and confusion matrix. Which can be found on Task Assignments\_ML Engineer Role\_Part3(Augmentation-backtranslation).ipynb file.

Result Analysis:

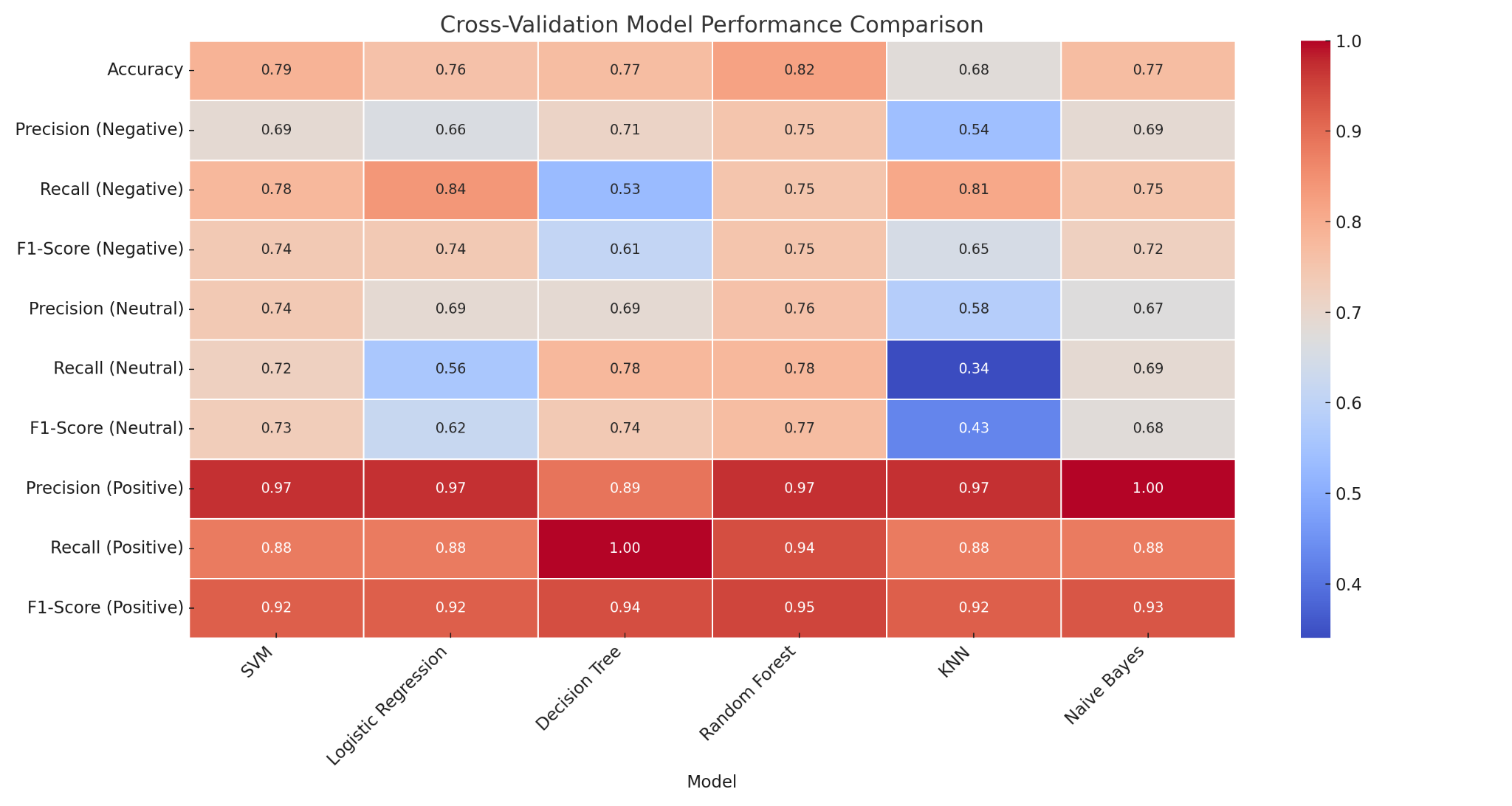
Before preprocessing:



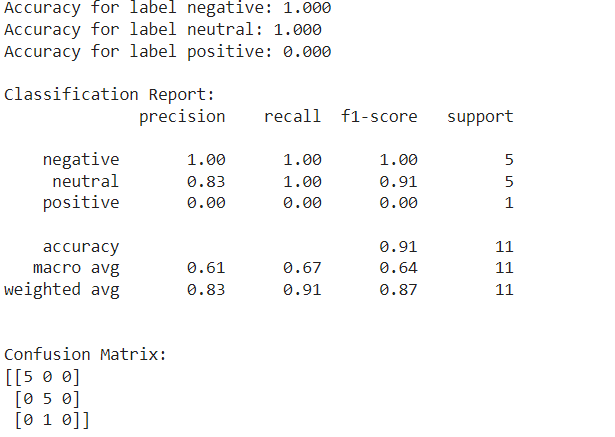
After Preprocessing and applying SMOTE:



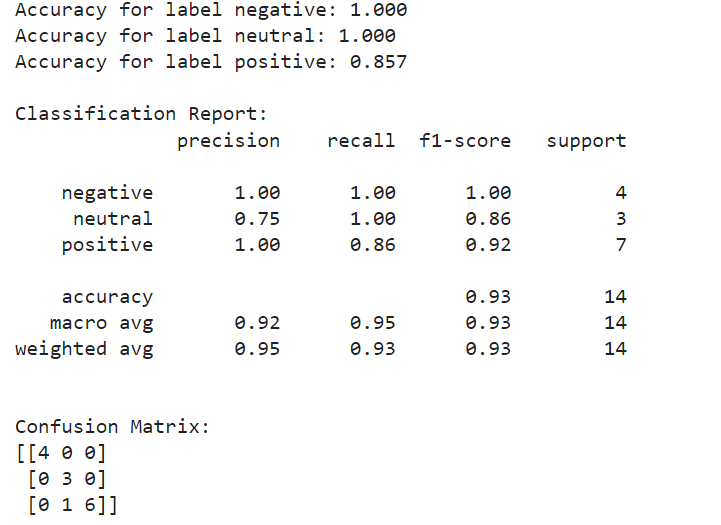
After Preprocessing and SMOTE and k-fold cross-validation:



Below is the results for Bangla sentiment analysis using LLaMA 3.1 (8B) trained on **Preprocess\_dataset.csv**



Below is the results for Bangla sentiment analysis using LLaMA 3.1 (8B) trained on **balanced\_dataset\_v1.csv**



The model perfectly classifies negative and neutral samples, with 100% accuracy for both. It's slightly less accurate for positive samples at 85.7%. The classification report provides detailed metrics for each label, showing high precision and recall overall. The model achieves an overall accuracy of 93%, with macro and weighted averages of 93% for precision, recall, and F1-score. The confusion matrix at the bottom indicates that there was one misclassification where a positive sample was incorrectly labeled as neutral.

LLaMA 3.1 has been shown to exhibit few-shot learning capabilities. This means that they can rapidly adapt to new tasks with minimal examples. The apparent near-perfect results in sentiment analysis might be an instance of such few-shot learning in action.

Challenges:

**Small Dataset Size and Imbalance**: The dataset contained only 99 samples, which is quite small for training any machine learning model. Additionally, the distribution of sentiment labels was highly imbalanced, with 45 neutral, 42 negative, and only 12 positive examples. This imbalance made it difficult for models to learn about the minority class (positive sentiment), as the majority class dominated the training process. Moreover, the small size of the dataset increased the risk of overfitting, where the model might memorize the data rather than generalize well to new samples.

**Handling Mixed-Language Text**: The dataset contained both Bangla and transliterated Banglish words, adding complexity to the preprocessing and feature extraction stages. In a larger dataset, Banglish words could potentially be removed, but since every piece of information is valuable in a small dataset, removing Banglish words risked losing important features. Additionally, preprocessing these mixed-language texts to maintain context while cleaning noise was challenging. The approach to retain Banglish words and apply the FastText model, which is well-suited for mixed-language datasets, was a strategic decision.

**Choosing the Right Model and Approach**: Initially, traditional machine learning models and neural network architectures like CNN, LSTM, RNN, and BiLSTM were explored. However, these models struggled to fully exploit semantic dependencies due to the small size of the dataset and the mixed-language nature of the text. Transitioning to a transformer-based model like LLaMA was a smart move, as these models excel at learning contextual relationships, even in small and imbalanced datasets.

**Low VRAM:** LLMs like LLama required higher VRAM capacity although using a Rtx 4090 was alright but only after quantizing the model within 4 bit quantization.