Chapter 3

A Preliminary investigation on Sentiment Analysis

By employing machine learning algorithms, sentiment analysis tools can effectively interpret and gauge public sentiments, providing valuable insights for businesses and researchers. It is important to investigate the appropriate machine learning based technique for sentiment analysis for Hindi. This chapter discusses the following:

- Implementation of machine learning classifiers for the Hindi-English language as a preliminary investigation utilizing the TU-HSA dataset.
- A comparative study of machine learning classifiers for sentiment analysis for Hindi-English Language.

The motivation is to identify effective machine learning techniques for accurate sentiment analysis in Hindi-English using the TU-HSA dataset.This study addresses the lack of sentiment analysis tools for Hindi-English code-mixed texts. By applying machine learning classifiers to the TU-HSA dataset, it identifies suitable techniques and benchmarks performance. The work supports inclusive NLP development, enabling better sentiment understanding for businesses and researchers in multilingual Indian contexts.

3.1 Methodology

The working framework is displayed in figure 3-1. In this framework the TU-HSA dataset is used to build a sentiment analysis model. The subsequent sections presents data preparation and the implementation of the sentiment analysis models. Only *positive* reviews and *negative* reviews from TU-HSA are used as data. It leads to binary classification problem [22].

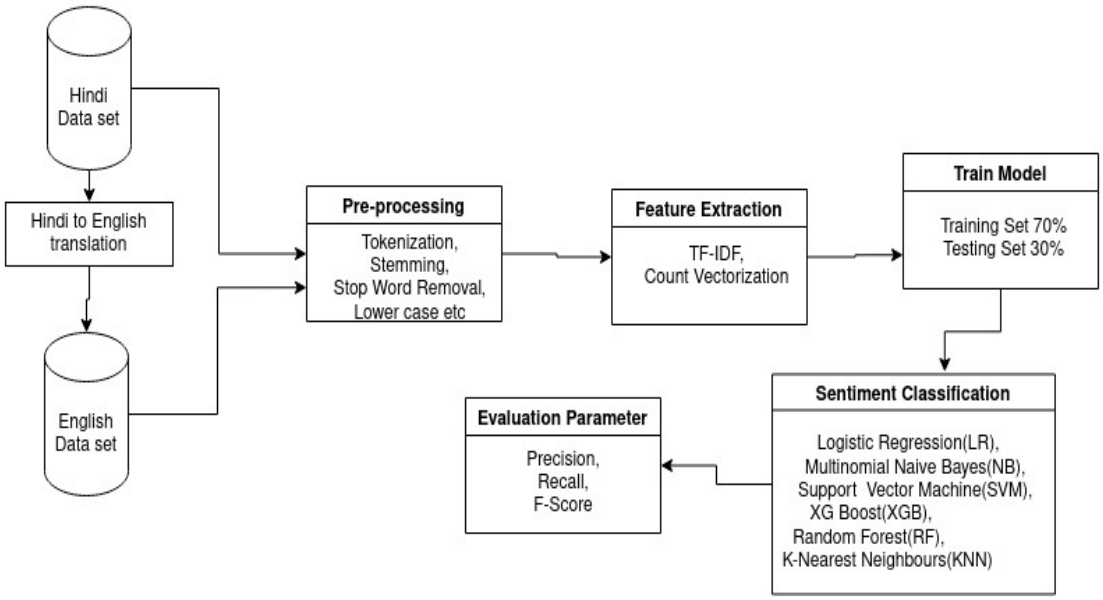

Figure 3-1: Proposed methodology

3.2 Setup for Experimentation

The experimental setup employed in the investigation is summarized in this section. It starts by providing an outline of data pre-processing [39] of both Hindi data and English data. The discussion then focuses on the experimental setups adopted by sentiment analysis systems.

3.2.1 Data pre-processing

In data pre-processing, the raw data is prepared for further processing and analysis. For machine learning algorithms to function as efficiently as possible, data must be clean and formatted correctly. Preparation, standardization and elimination of identical values from the data are few of key pre-processing techniques. Pre-processing techniques were applied to the Hindi dataset and the translated English dataset. At this stage, all unnecessary characters and words are eliminated, including punctuation like!, @, dot,?, and so on. Furthermore, as the attitude remains the same in any form, all English uppercase letters are converted to lowercase. In sentiment analysis, focus is often on overall sentiment rather than specific emphasis. The unprocessed text is divided into manageable chunks using tokens. To improve comprehension, tokenization entails segmenting each text into distinct word and phrase units. NLP uses stemming [55] as a normalization strategy to reduce computational load. Inflected words are returned to their original forms. A lot of English terms, such I, will, is, for, a, an, the, and so on, don't make sense grammatically and are therefore not required for sentiment analysis. These are stop words [103]. Stop words are removed from English sentences. Further processing is performed on remaining words of Hindi and English review sentences. An illustration of pre-processing one Hindi review and is provided below.

Raw sentence: फिल्म को दर्शकों से बेहिसाब प्यार मिला |

After pre-processing: फिल्म दर्शकों बेहिसाब प्यार मिला

Below is an example of the pre-processing of one English review.

Raw sentence: **The film received immense love from the audience.**

After pre-processing: **film received immense love audience**

3.2.2 Feature Extraction

The TF-IDF and count vectorization [83] models are frequently utilized for a wide range of NLP approaches because of their simplicity and effectiveness. The significance and frequency of a word to the corpus are determined using these mathematical-statistical methods [102]. TF-IDF enables a comprehensive analysis of term relevance, enhancing the performance of sentiment classification models. TF-IDF and Count Vectorization are discussed below:

- **TF-IDF**: The TF-IDF of each word in a text is found by multiplying two different metrics:

  1)Term frequency(TF) of a word is number of times a word occurs in a written piece.

  2) Inverse Document Frequency(IDF) of a word is calculated by taking the number of documents N, divided by the total number of documents that contains the given word, and then computing the logarithm. That is the TF-IDF score for the word t in the document d from the document collection D is

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{3.1}$$

$$\text{tf}(t, d) = \log(1 + \text{freq}(t, d)) \tag{3.2}$$

$$\text{idf}(t, D) = \log\left(\frac{N}{\text{count}(d \in D : t \in d)}\right) \tag{3.3}$$

- **Count Vectorization:** Count vectorization transforms text documents into a numerical matrix based on word frequency. Vocabulary V comprising all unique words found in the corpus and $|V|$ denote the size of this vocabulary. For each document d_i, represent it as a vector $\mathbf{x}_i$ of length $|V|$. Each component x_{ij} of this vector is the count of the j-th word from the vocabulary in the i-th document:

$$\mathbf{x}_i = (t_{i1}, t_{i2}, \ldots, t_{i|V|})$$

where t_{ij} is the count of the j-th word in the i-th document.

Combine these document vectors into a document-term matrix X. In this matrix, each row $\mathbf{x}_i$ represents a document, and each column corresponds to a word in V. Thus, X is an $m \times |V|$ matrix, where m is the number of documents:

$$X = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1|V|} \\ t_{21} & t_{22} & \cdots & t_{2|V|} \\ \vdots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{m|V|} \end{bmatrix}$$

Each entry t_{ij} represents the count of the j-th word in the i-th document, creating a matrix that reflects word frequencies across the entire corpus.

3.2.3 Model Training

70% samples are used as train set and 30% are used for test set, 70-30 rule. For the purpose of generalizing the model to new data, 10-fold stratified cross validation [97] is utilized. While stratified cross-validation guarantees that each fold has a representative distribution of the target variable. for example, if a dataset is 30% Class A and 70% Class B, each fold will maintain this ratio, ensuring consistent and representative performance evaluation across different subsets of the data. It is helpful for imbalanced datasets [63] to maintain the proportion of classes. Following ML based classifiers are used for model learning:-

- **Logistic Regression classifier(LR)** LR [67] is a foundational statistical method widely employed in binary classification tasks due to its robust mathematical framework. At its core, LR models the probability of a binary

outcome using the logistic function, which transforms a linear combination of predictor variables into a probability value bounded between 0 and 1.

In the context of sentiment analysis, LR serves as a pivotal tool for discerning sentiment polarity in textual data. The mathematical formulation of LR involves fitting a logistic regression model to the training data, where the probability p of the positive sentiment class is modeled as equation 3.4:

$$p(y = 1 \mid x) = \frac{1}{1 + e^{-(\mathbf{w}^\intercal \mathbf{x} + b)}} \tag{3.4}$$

Here, x is input feature vector derived from the text data, w denotes the weight vector, b is the bias term and the superscript $\top$ denotes the transpose operation. The logistic function $\frac{1}{1+e^{-z}}$ maps the linear combination of features $\mathbf{w}^\intercal \mathbf{x} + b$ to a probability value. Following example illustrates it:
Consider the following two features in SA dataset. Feature 1 is frequency of the word **"अच्छा"** (good) and feature 2 is frequency of the word **"बुरा"** (bad). Suppose weights $\mathbf{w}$ and bias b are as follows:

- Weight Vector: $\mathbf{w} = [0.6, -0.5]$
- Bias: $b = 0.1$

or a given Hindi sentence with the feature vector:

- Feature Vector: $\mathbf{x} = [4, 2]$ (which means the sentence contains 4 instances of **"अच्छा"** and 2 instances of **"बुरा"**)

We can plug these values into the logistic regression equation to compute the probability $p(y = 1 \mid \mathbf{x})$:

$$p(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\mathbf{w}^\top \mathbf{x} + b)}}$$

Here's the step-by-step calculation:

1. Compute the dot product $\mathbf{w}^\top \mathbf{x}$:

$$\mathbf{w}^\top \mathbf{x} = [0.6, -0.5] \cdot [4, 2] = (0.6 \times 4) + (-0.5 \times 2) = 2.4 - 1.0 = 1.4$$

2. Add the bias b:

$$\mathbf{w}^\top \mathbf{x} + b = 1.4 + 0.1 = 1.5$$

3. Compute the probability using the logistic function:

$$p(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-1.5}}$$

$$e^{-1.5} \approx 0.2231$$

$$p(y = 1 \mid \mathbf{x}) = \frac{1}{1 + 0.2231} \approx \frac{1}{1.2231} \approx 0.82$$

Thus, the probability of the sentiment being positive for this review is approximately 0.82, or 82%.

The LR model is trained by minimizing a loss function [113], like the binary cross-entropy loss, using an optimization approach like gradient descent. To optimize the likelihood of the observed labels in the training data, the optimization procedure iteratively modifies the weights and bias.

Once trained, the LR model can make predictions on new textual data by computing the probability of positive sentiment using the learned parameters. A threshold can be applied to convert these probabilities into binary predictions.

- **Naive-Bayes classifier(NB)** Bayes' theorem [92] is a fundamental concept in probability theory. It calculates the probability of a hypothesis (such as a sentiment label) given the evidence (the features or words in the

text). Mathematically, it is represented in equation 3.5:

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)} \tag{3.5}$$

Where:

$P(y|x)$ is the probability of class y given the evidence x. $P(x|y)$ is the probability of observing evidence x given class y. $P(y)$ is the prior probability of class y. $P(x)$ is the probability of observing evidence x. It is normalization constant.

The "naive" part of NB comes from the assumption that features (words in this case) are conditionally independent given the class label. In other words, the presence of one word in the text does not affect the presence of another word. Mathematically, this is expressed in equation 3.6:

$$P(x_1, x_2, ..., x_n|y) = P(x_1|y) \cdot P(x_2|y) \cdot ... \cdot P(x_n|y) \tag{3.6}$$

Once we calculate the probability of each sentiment class given the document, we select the class with the highest probability as the predicted sentiment $\hat{y}$ for the document as equation 3.7:

$$\hat{y} = \arg\max_y P(y|x) \tag{3.7}$$

During training, we estimate the probabilities $P(w_i|y)$ and $P(y)$ from the labeled training data. We then evaluate the model's performance on the test dataset.

- **Support-Vector Machine(SVM)** SVM is a powerful supervised machine learning algorithm used for sentiment analysis task. It is effective for sentiment analysis because it is capable of capturing complex relationships in high-dimensional data, like text, and can handle both linear and non-linear decision boundaries.

  SVM works by optimizing the margin—also referred to as support vec-

tors—between the nearest data points and the hyperplane. The decision boundary (hyperplane) is defined by the equation 3.8:

$$w \cdot x + b = 0 \tag{3.8}$$

Where:
The weight vector, normal to the hyperplane, is denoted by w. The input feature vector is x. The bias term is b. The distance between a data point x_i and the hyperplane is given by equation 3.9:

$$\frac{\mid w \cdot x_i + b \mid}{\|w\|} \tag{3.9}$$

SVM locates the hyperplane with the least classification error and the highest margin. The formulation of this is as an optimization problem expressed in equation 3.10 :

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{3.10}$$

Subject to the constraints:

$$y_i(w \cdot x_i + b) \geq 1 \text{ for } i = 1, 2, \ldots, N$$

where N is the number of data points. y_i is the class label of the i^{th} review. In binary sentiment classification y_i represents the sentiment such as $+1$ for positive sentiment and -1 for negative sentiment. b is the bias term in the SVM. During training, SVM learns the optimal hyperplane [29] and kernel parameters [114] from the labeled training data. Then, during prediction, it assigns new data points to classes based on which side of the hyperplane they fall.

- **XG Boost classifier** A well-liked and potent machine learning algorithm for classification tasks, including sentiment analysis, is called XGBoost [11].

It's an ensemble learning technique that creates a powerful learner by combining the predictions of several weak learners, usually decision trees.

In gradient boosting, the objective is to minimize a loss function L by iteratively adding weak learners h_i to the ensemble. The final model $F(x)$ is the sum of all weak learners:

$$F(x) = \sum_{i=1}^{N} h_i(x) \tag{3.11}$$

$F(x)$ is expressed in equation 3.11. The goal of training each weak learner is to minimize the loss function, which is usually a measure of the discrepancy between the actual and predicted values (e.g., cross-entropy loss for classification, mean squared error for regression).

XGBoost uses a more efficient tree learning algorithm compared to traditional gradient boosting. It applies a technique called "weighted quantile sketch" to efficiently find the best splits in each tree node. This reduces the computational cost of building trees and improves the overall training speed. The objective function in XGBoost combines the loss function L with the regularization terms to be optimized during training. For classification tasks, the objective function often includes the softmax function to compute class probabilities.

During training, XGBoost iteratively adds trees to the ensemble, optimizing the objective function using gradient descent. It stops adding trees when adding more trees no longer improves performance or when a specified number of trees is reached.

- **Random Forest classifier (RF)** Random feature selection and bagging, or bootstrap aggregating, are the foundations of random forest [17]. In order to create a more reliable and accurate model, it independently constructs several decision trees and then combines their predictions. A decision tree is a model that resembles a tree, in which each leaf node indicates the class

label (in classification) or the predicted value (in regression), and each inside node reflects a decision based on the value of a feature. Recursively building decision trees involves dividing the data at each node according to the feature that yields the greatest increase in information or reduction in impurity.

By training each tree on a random subset of the training data (with replacement) and a random subset of features at each node split, random forest creates an ensemble of decision trees. This unpredictability lessens over-fitting and de-correlates the trees. Every decision tree in the random forest receives training using a boot-strapped sample of the initial training set. Bootstrapping is the process of generating numerous equal-sized subsets by randomly choosing the training data with replacement. Only a random subset of features—rather than all features—are taken into account at each node split of a decision tree in order to determine the optimal split. By doing this, the ensemble's variety is increased and the correlation between the trees is decreased.

Random forest uses a majority voting mechanism to aggregate the predictions of individual trees in classification problems. The ultimate forecast is given to the class that the trees vote on the most. Recursively splitting the data according to the features that are chosen, random forest constructs several decision trees on its own during the training process. In order to arrive at the final forecast, RF uses the voting mechanism (for classification) or averaging (for regression) to integrate the predictions of all the trees in the ensemble.

- **k-Nearest Neighbour classifier(KNN)** KNN [104] is based on the principle that data points with similar features tend to belong to the same class. It classifies a new data point by examining the classes of its k-nearest neighbors in the feature space.

  Given a dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\}$ where x_i represents the feature vector of the i^{th} data point and y_i represents its class label. KNN

uses a distance metric (e.g., Euclidean distance, Manhattan distance [75]) to measure the similarity between data points in the feature space. Let $d(x_i, x_j)$ denote the distance between data points x_i and x_j. For a given test data point

x_{test}, KNN identifies its k-nearest neighbors from the training data based on the chosen distance metric. KNN assigns the class label to the test data point by performing a majority vote among the classes of its k nearest neighbors. If $k = 1$, the class label of the closest neighbor is directly assigned to the test point. KNN assigns the class label to the test data point by performing a majority vote among the classes of its k nearest neighbors. If $k = 1$, the class label of the closest neighbor is directly assigned to the test point. During training, KNN simply memorizes the training data. During prediction, KNN calculates the distance between the test data point and all training data points, identifies the k nearest neighbors, and assigns the majority class label to the test point. KNN is a straightforward and interpretable algorithm for sentiment analysis.

These classifiers are chosen due to their proven effectiveness in text classification [43] tasks and their ability to represent diverse learning.

3.3 Results and Analysis

F1-score, precision, and recall were preferred over accuracy as they provide more reliable evaluation under class imbalance conditions. The performance metrics of machine learning-based classifiers for TU-HSA described in Section 2.6 using the TF-IDF feature extraction technique are shown in table 3.1 and figure 3-2. With the greatest precision (79.51%), recall (96.11%), and F1-score (87.03%) among all classifiers, the support vector machine(SVM) performed better than any other.

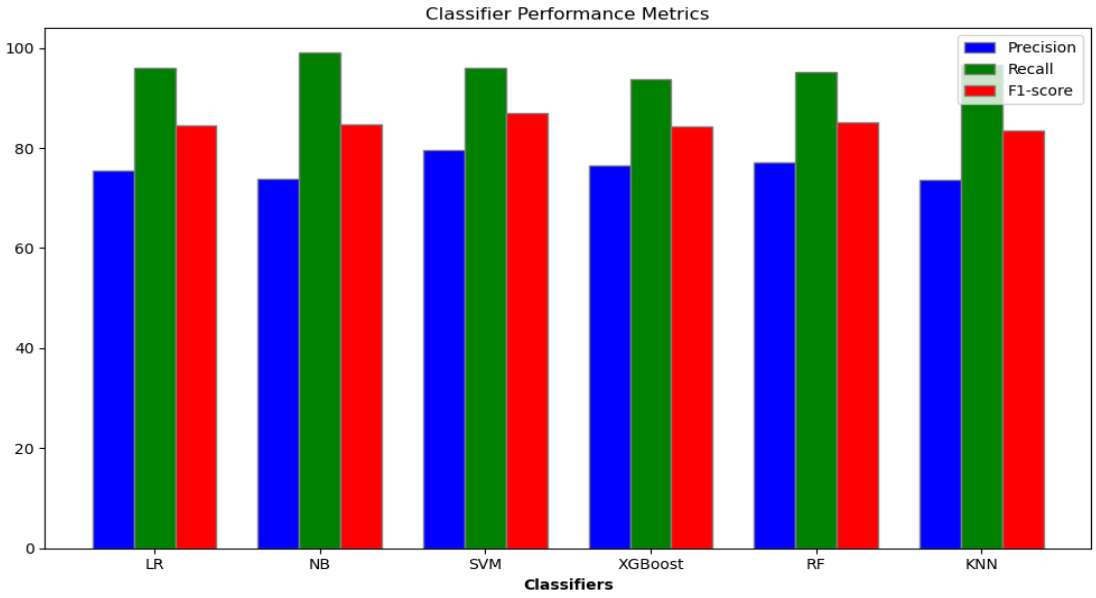

Figure 3-2: Evaluation for TU-HSA dataset(TF-IDF)

For TU-HSA data set

The performance metrics of machine learning based classifiers for TU-HSA dataset with Count Vectorizer feature extraction technique is depicted in table 3.2 and figure 3-3. The SVM outperformed all other classifiers with the highest

Table 3.1: Evaluation Metrics (TF-IDF)

Classifier	**Precision(%)**	**Recall(%)**	**F1-score**(%)
LR	75.57	96.11	84.61
NB	73.91	99.02	84.64
SVM	**79.51**	**96.11**	**87.03**
XGBoost	76.58	93.68	84.27
RF	77.16	95.14	85.21
***k*-NN**	73.70	96.60	83.61

Table 3.2: Evaluation Metrics (Count Vectorizer)

Classifier	**Precision(%)**	**Recall(%)**	**F1-score(%)**
LR	76.34	89.32	82.32
NB	79.56	88.83	83.94
SVM	**79.91**	**90.77**	**84.85**
XGBoost	76.82	91.74	83.62
RF	79.14	90.29	84.35
***k*-NN**	74.04	94.17	82.90

precision 79.91%, recall 90.77% and F1-score 84.85%.

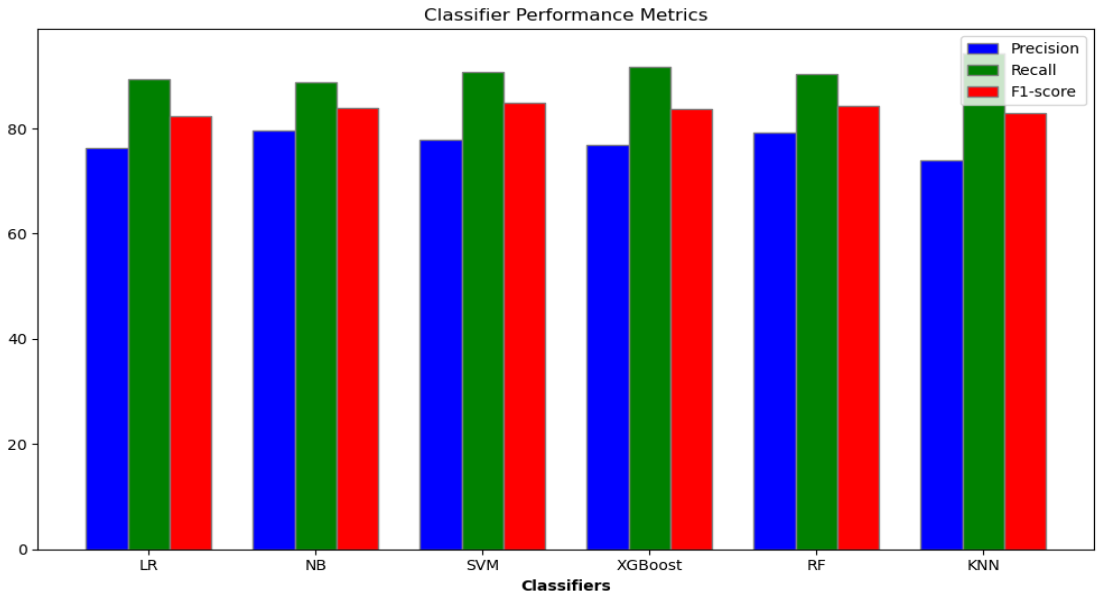

Figure 3-3: Evaluation for TU-HSA(Count Vectorizer)

The performance metrics of machine learning based classifiers for TU-HSA(English) dataset with TF-IDF feature extraction technique is depicted in figure 3-4 and table 3.3. The SVM outperformed all other classifiers with the highest precision 83.40%, recall 90.95% and F1-score 87.01%.

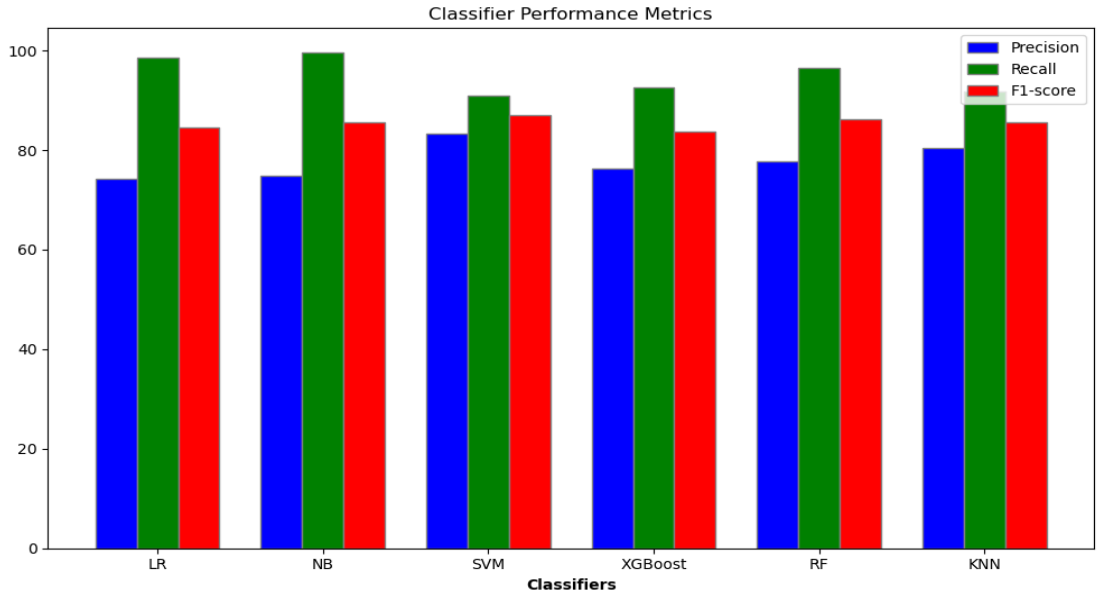

Figure 3-4: Evaluation for TU-HSA(English)(TF-IDF)

The performance metrics of machine learning based classifiers for TU-HSA(English) with Count Vectorizer feature extraction technique is depicted in figure 3-5 and table 3.4. The SVM outperformed all other classifiers with the highest precision 86.38%, recall 85.98% and F1-score 88.98%.

For TU-HSA(English)

Table 3.3: Evaluation Metrics (TF-IDF)

Classifier	Precision(%)	Recall(%)	F1-score(%)
LR	74.16	98.52	84.63
NB	74.90	99.51	85.47
SVM	**83.40**	**90.95**	**87.01**
XGBoost	76.20	92.64	83.62
RF	77.68	96.53	86.09
***k*-NN**	80.34	91.70	85.64

Table 3.4: Evaluation Metrics (Count Vectorizer)

Classifier	Precision(%)	Recall(%)	F1-score(%)
LR	79.07	91.30	84.75
NB	81.93	89.86	85.71
SVM	**86.38**	**85.98**	**86.98**
XGBoost	77.30	98.04	86.45
RF	78.68	95.04	86.09
***k*-NN**	79.90	81.77	80.83

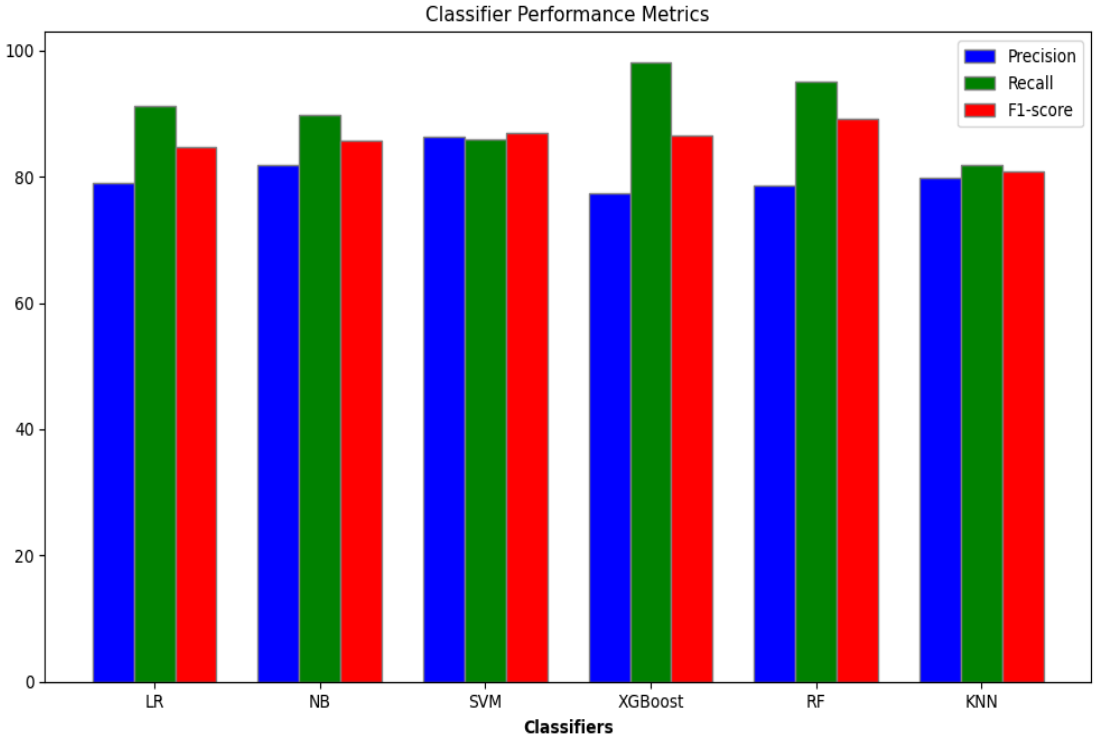

Figure 3-5: Evaluation for TU-HSA(English)(Count Vectorizer)

Every classifier has a high level of precision. The table shows that SVM functions with greater precision. If the recall is greater than 0.5, that is ideal. The F1-score for the SVM classifier is also the highest in each of the testing cases.

3.4 Summary

This work investigated machine learning-based methods to analyze post-web reviews written in the Hindi language. The proposed approach primarily involves the

extraction of features and their subsequent categorization. The feature extraction process involved the utilization of the Count Vectorizer and TF-IDF algorithms to choose relevant features. Various classification algorithms have been successfully tested as classifiers. The SVM demonstrated superior performance compared to other models when applied to Hindi and English data for machine learning classification. The achievement can be ascribed to the good handling of TU-HSA data by SVM, which creates a strong decision boundary that performs exceptionally well in both languages. The high efficacy of SVM indicates that it is especially suitable for text classification jobs that need precise differentiation between various classes. The model's performance demonstrates robust generalization abilities and efficient feature extraction, rendering it a dependable option for multilingual sentiment analysis [27].