Sentiment Analysis of IMDb Movie Reviews : A comparative study on Performance of Hyperparameter-tuned Classification Algorithms

Ayanabha Ghosh
Department of Computer Science and Engineering
Indian Institute of Technology Jodhpur
Jodhpur, India
ghosh.8@iitj.ac.in ghoshayanabha@gmail.com

2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS) | 978-1-6654-0816-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICACCS54159.2022.9784961

***Abstract*—Sentiment Analysis (SA) is a sub-domain of Natural Language Processing where useful insights on sentiment and opinion of people can be obtained and analyzed from various textual data in structured, unstructured and semi-structured format. In this work, I have tried to analyze the sentiment of viewers from the IMDb Movie Reviews dataset. For this, I have taken three different Supervised learning methods, namely, Linear Support Vector Machine, Logistic Regression and Multinomial Naive Bayes Classifier, each with different settings of hyperparameters. Moreover, to capture the notion of informal jargon, approach of N-grams has been followed. Furthermore, a comparative study has been performed to find the one best model from each of the different types of above mentioned Supervised learning techniques based on their Accuracy Score, F1-Score and AUC Score. In this approach, I have obtained the best accuracy score of around 0.910 and mean F1-score after 10-fold Cross validation of around 0.894.**

***Index Terms*—Text mining, Information retrieval, Sentiment analysis, Binary Classification, Supervised learning, IMDb dataset, AUC score.**

I. Introduction

In the present era of social media and online content distribution, huge amount of data is generated at a daily basis by the users of such platforms and thus, the amount of data is exponentially growing day by day. One of the major part of such data is the reviews given by the users on various things such as their purchases on e-commerce platforms, the shows they watch, the news they read and so on. Such opinions expressed by the people help the stakeholders to understand their sentiment. From a business point of view, the reviews given by a group of consumers on a particular product help to gain insights regarding customers' needs or requirements, which help a business to grow and improve.

As the reviews or opinions come from various sources in unstructured or semi-structured format, it becomes quite difficult for a human being to bring those in structured format, analyse them and find valuable insights from those. Here is the need of machines which have the capability to do aforementioned objectives in a short span of time.

The ability of a machine to learn and process things at a higher rate gives rise to a sub-domain of text mining called Sentiment Analysis or Opinion Mining. Text processing and Sentiment analysis is a challenging task as it includes processing of streams of textual data in unstructured or semi-structured format depending upon the source. Pre-processing of texts includes several computational steps such as Removing punctuation, stemming and/or lemmatization, removal of stopwords and building vocabulary with filtered tokens. It is an important step as it eventually filters out various important tag words. One can easily understand the implication of a review by looking at the tokens. Thus, a wrong or bad pre-processing may sometimes lead to wrong conclusions, which is not at all intended. After this, the analysis can be done by the machines by capturing the tag words that may actually describe the meaning of a sentence.

The next step of pre-processing involves building the classifier which can use the knowledge base and extract meaningful insights. There are several algorithms out there such as Linear Support Vector Machines, Logistic Regression, K-Nearest Neighbours, Random Forest and so on. There are two types of learners namely Eager Learners and Lazy Learners. Eager learners build model from the existing knowledge base i.e. training data and use the model to predict the unknown data which is given to it. On the other hand, Lazy learners are those who store the training data and perform classification at the inference stage when unknown data is available.

In this work, I have used the IMDb Movie Reviews dataset to perform the **task of sentiment analysis** using three different supervised learning algorithms namely **Linear Support Vector Machine, Logistic Regression and Multinomial Naive Bayes Classifier. Various combinations of the hyperparameters have been taken and fine-tuned** so as to find the best performing model on this dataset. Furthermore, the performance of the models have been evaluated using three different evaluation metrics namely, Accuracy Score, F1-Score and AUC Score.

In Section II, I have given a brief survey of previous works done in this area. Section III describes the methodology of experiments carried out in this work. In the second last Section IV, experimental results have been shown and explained with proper diagrams and plots.

II. Previous Works

The authors of [1] have applied four different classification algorithms on IMDb movie reviews dataset and evaluated their performance using Accuracy Score, F1 Score and AUC Score. The authors have not mentioned

978-1-6654-0816-5/22/$31.00 ©2022 IEEE

any particular settings of the hyperparameters for the classification algorithms.

The authors of [6] have used 3 different classification algorithms for performing sentiment analysis. The drawback is they have not taken different hyperparameters and language models. Moreover, the authors have only used Term Frequency for quantifying the reviews. They have obtained accuracies 85.69%, 86.23% and 83.54% for Linear SVM, Logistic Regressor and Multinomial Naive Bayes classifiers respectively.

The authors of [3] have examined the sentiment from IMDb Movie Reviews dataset to find the polarity of the movie reviews on a scale of 0 (highly disliked) to 4 (highly liked). Then they have performed feature extraction, followed by training a multilabel classifier to classify the reviews into its correct label. They have obtained an accuracy of around 88.95%.

The authors have [4] have applied sentiment analysis on IMDb Movie Reviews Dataset in which, they have applied various steps of text processing and feature selection, followed by classifying them into positive or negative reviews. Furthermore, they have evaluated the model with eight different classifiers using five different evaluation metrics.

III. Methodology

The steps which have been followed are described in this section.

A. The Dataset

I have chosen the IMDb Movie Reviews Dataset from Kaggle. This dataset is originally derived from the **Large Movie Review Dataset**, which consists of around 50,000 highly polar movie reviews among which, 25,000 are given as Training data and rest 25,000 are for Test. The dataset in CSV format I have used here is made from the original dataset. It has two columns namely *review* and *sentiment*. The *review* column consists of all 50,000 reviews from Training and Test set. The *sentiment* column consists of the corresponding sentiment as either *positive* or *negative*.

The entire dataset has been divided into 3 subsets by randomly picking up each instances, Train, Validation and Test set in a ratio of 70:15:15.

B. Preprocessing

After loading the dataset, it has to be pre-processed in a proper step-by-step manner. The pre-processing steps that we have followed are described below,

1) Removal of HTML and CSS tags: The reviews contain certain HTML tags such as `<br>...</br>` and CSS stylings, which need to be removed during pre-processing. For this purpose I have used the built-in HTML parser of `BeautifulSoup` library.

2) Casefolding: Casefolding is done in order to normalize tokens so that same words written with different cases or one with first letter capitalized can be treated as same tokens.

3) Removal of Punctuations: The punctuations have been removed from each of the reviews.

4) Removal of Stopwords: Each review is split into tokens and from the generated token streams, stopwords are removed. For this step, the English language stopwords' set from `nltk.corpora` has been used.

5) Stemming: Stemming is a heuristic process of chopping off the tail of each word in order to achieve a base form of the token. In this step, stemming has been performed on the remaining tokens using Porter's Stemming Algorithm. In this work, I have used the `PorterStemmer()` method of `nltk`.

Another alternative to stemming is the process called Lemmatization, which involves converting the token into their actual dictionary form. It is a computationally expensive process as compared to Stemming and using this to pre-process large dataset will take much more time.

C. Bag-of-Words & TF-IDF Vectorization

In text dataset, tokens are the features of a particular document. Unique token are extracted from the entire dataset or corpus and a dictionary is created. Relevant information such as Count and/or TF_IDF weightage etc. is stored corresponding to each of the features and a **Bag-of-Words** representation is created. As machines cannot understand the tokens in text format, these numerical valued features are fed to the learner so as to make it analyzable by the machine.

Here, I have used the **TF-IDF** scoring of the tokens to map them into numerical values. TF-IDF stands for **Term Frequency - Inverted Document Frequency**. Term Frequency is defined as the total number of times a token is appearing in a document. As the number of words in a document may greatly vary, Document Length Normalization has been performed while finding TF. The expression for finding TF is as follows,

$$tf(t,d) = \frac{f_{t,d}}{d.length}$$

where $f_{t,d}$ is the number of times term t occurs in document d; $d.length$ denotes the total number of words in document d.

Inverted Document Frequency of a term is defined as the logarithmically scaled fraction of the corpus size and the number of documents which contain that term. It captures how much information a particular term may contain. Moreover, the logarithm rewards the terms appearing less number of times as well as penalizes the influence of terms occurring higher number of times. The expression for finding IDF is as follows,

$$idf(t) = \log \frac{N}{df(t)}$$

where $df(t) = |d \in D : t \in d|$ is the Document Frequency of t; D is the corpus; $N = |D|$ is the size of corpus. It may also happen that a term may not be present in the corpus, therefore making the denominator $df(t) = 0$. To avoid such problem, we add 1, hence smoothing the IDF score.

$$idf(t) = \log \frac{N}{1+df(t)}$$

Therefore, the TF-IDF score is calculated by multiplying the TF and IDF of a term obtained by following the above steps.

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

But in this approach also, *tf(t, d)* of term *t* gets rewarded when it appears significantly higher number of times. So, a common modification can be done as taking the logarithm of the term frequency as shown,

$$tfidf(t, d, D) = (1 + log(tf(t, d))) \times idf(t, D)$$

All the reviews of the dataset are pre-processed through the steps explained above. Text pre-processing is a lengthy and time-taking process. For this reason, to ease the implementation, I have saved a copy of the pre-processed dataset in a seperate CSV file and used in further steps.

D. Model Selection and Hyperparameter tuning

As the problem, being addressed, is a Binary Classification Problem, we have chosen the Linear Support Vector Machine, which uses Linear hyperplane to separate the two classes. Other classification methods I have chosen are Logistic Regression and Multinomial Naive Bayes. Linear SVM has been widely regarded to be one of the best classification algorithm for classifying text data [5].

1) Setting up different hyperparameters: At first, different hyperparameter values for each of the algorithms have been taken. In case of Linear SVM and Logistic Regression, the following hyperparameter settings have been taken,

- **C** : It is a regularization parameter which is inversely proportional to the regularization strength. Lower values of C allow the classifier to misclassify more number of points. The following values of C have been taken while finding the best model : 0.001, 0.01, 0.1, 1, 10, 100.
- **Class Weight** : In case of imbalanced training dataset, we can provide more weightage to the class having less number of corresponding training samples. This is set to 'balanced' which adjusts weights inversely proportional to class frequencies in the input data.
- **Maximum iteration** : It denotes the maximum iterations to be taken by the solver to converge. In this case, the value has been kept as 100000.

In case of Multinomial Naive Bayes Classifier, different values of smoothing hyperparameter α have been taken. I have taken the same values as of the hyperparameter C in case of Linear SVM and Logistic Regression. The values of hyperparameters fit_prior and class_prior have been left with default values which are True and None respectively, as per Scikit-learn's implementation.

Furthermore, I have also considered different n-gram ranges starting from (1, 1) to (1, 4) i.e. considering a single token as a single feature and considering upto 4 consecutive tokens together as a single feature. In case of n-gram value of 2, it means taking single token as single feature, then considering two consecutive tokens as features, therefore increasing the size of feature space.

2) Training: For each different hyperparameters and for each different n-gram range, separate classifiers of the selected supervised learning techniques have been trained using Train set. In each iterations, a `best_model` flag keeps track of the model's accuracy score and F1 score on Validation set. If a model yields better F1 score than in the previous iteration, the `best_model` flag gets updated. At the end, the model is being saved into the secondary memory. For each of the candidate supervised techniques, best model is found and saved into the secondary storage.

3) Loading and Prediction: At the inference stage, the saved models are listed and parsing the filenames, the particular type of classifier is loaded into main memory. The model predicts the unseen Test data and gives its prediction. Then, based on the prediction of Test data points, the Accuracy Score, F1 Score and AUC Score are computed for Test dataset.

The steps have been depicted in Fig. 1

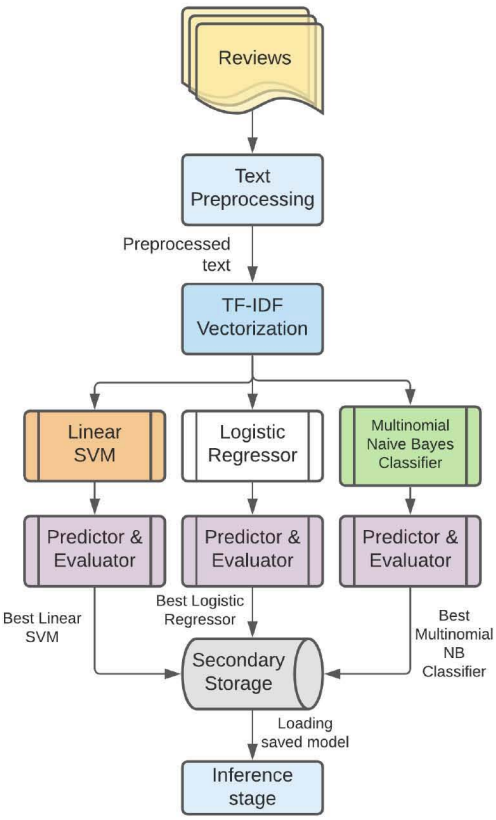

Fig. 1. Flowchart of Methodology

IV. Experimental Results

For the evaluation of the models, I have taken 3 metrics namely Accuracy Score, F1 Score and Area Under the Curve (AUC) Score.

- **Accuracy Score** : It is defined by the total number of correct classifications predicted by the classifier. It is computed as,

$$accuracy(y, \hat{y}) = \frac{1}{N} \times \sum_{i=0}^{N-1} 1(y_i = \hat{y_i})$$

where N is the total number of samples, y is the ground truth and $\hat{y}$ is the predicted values given by the classifier.

- **F1 Score** : The formula for computing F1-score is,

$$f1score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

where **Precision** is defined as, for a particular label, the fraction of total data points correctly classified into that label (True Positives) by the classifier among all the data points that the classifier has predicted to be of that label (i.e. fraction of Relevant data among the Retrieved data).

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Recall is the fraction of total data points correctly classified into a label (True Positves) by the classifier among all the data points with that label present in the dataset (i.e. fraction of Retrieved among the Relevant Data).

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

The F1-score has been calculated by taking the mean of F1-scores after 10-fold Cross validation using the Validation dataset.

- **AUC Score** : Area Under Curve (AUC) Score is the measure or the ability of a classifier to distinguish between the classes. It measures the area under the Receiver Operating Characteristic (ROC) curve. ROC Curve plots two parameters, namely, True Positive Rate (TPR) along the vertical axis and False Positive Rate (FPR) along the horizontal axis.

$$TPR = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$FPR = \frac{FalsePositive}{FalsePositive + TrueNegative}$$

The higher the AUC score, the better the classifier can distinguish between positive and negative classes, in case of a binary classification problem.

During the training, at each iteration a record of accuracy score and F1 score along with the C and n-gram range values have been kept and plotted for each model in a 2D space. The obtained results are as follows,

A. Linear SVM

The C vs Accuracy Score plot is shown in Fig.-2 and the C vs F1 Score for the same is shown in Fig.-3. Clearly, it can be said that for 1-gram, the accuracy and F1 score increase at first but it drastically reduce for higher value of C. Moreover, it never performed better than other n-gram range. Among all, the best performing n-gram range is (1, 2) when the values of C are higher.

The Test Accuracy score and F1-score of the best model are **0.910** and **0.894** respectively. The AUC score for the best performing Linear SVM model is **0.97** [Fig. 4].

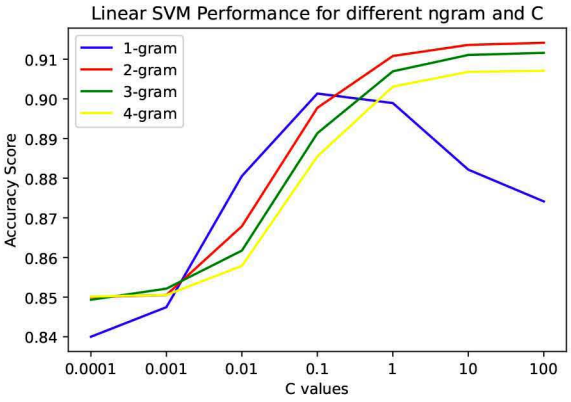

Fig. 2. C-Accuracy Score plot of Linear SVM with different n-gram range

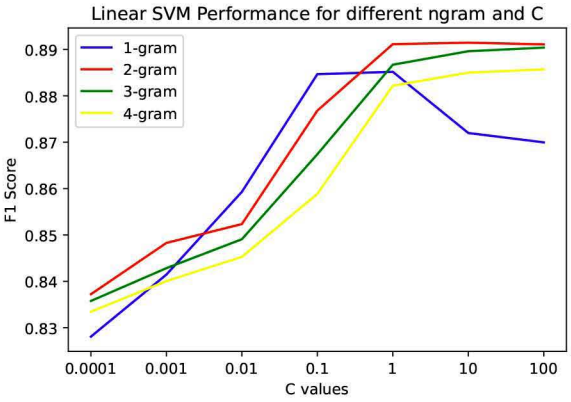

Fig. 3. C-F1 Score plot of Linear SVM with different n-gram range

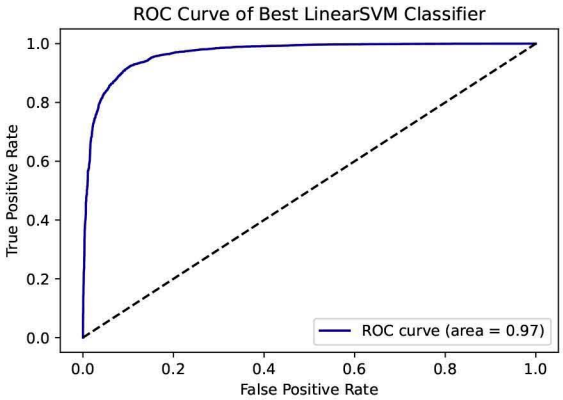

Fig. 4. ROC plot of Linear SVM with AUC Score

B. Logistic Regression

The C vs Accuracy Score plot is shown in Fig.5 and the C vs F1 Score for the same is shown in Fig.-6. Clearly, it can be said that for 1-gram, the F1 score was higher than other n-gram ranges at first but it reduced for higher value of C. On the other hand, the Accuracy can be seen varying somewhat similar to what has been observed in case of Linear SVM. Among all, the best performing n-gram range is (1, 2) when the value of C is higher.

The Test Accuracy score and F1-score of the best model

are **0.907** and **0.893** respectively. The AUC score for the

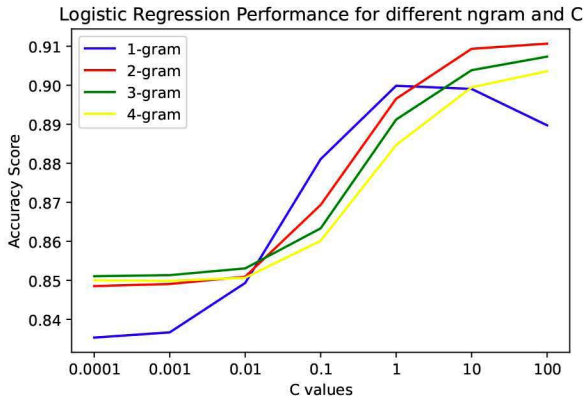

Fig. 5. C-Accuracy Score plot of Logistic Regression with different n-gram range

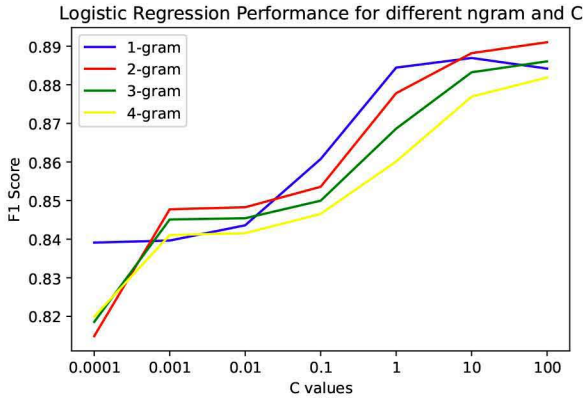

Fig. 6. C-F1 Score plot of Logistic Regression with different n-gram range

best performing Logistic Regression model is **0.97** [Fig. 7].

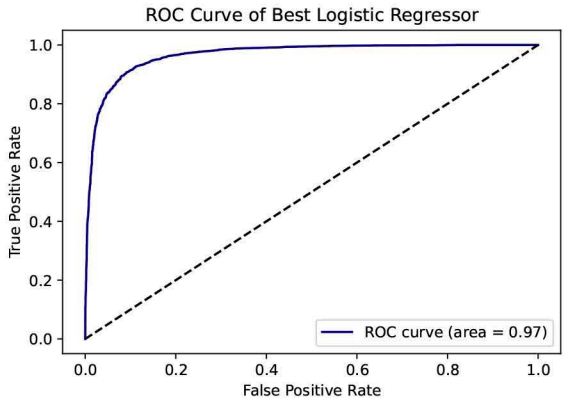

Fig. 7. ROC plot of Logistic Regression with AUC Score

C. Multinomial Naive Bayes

The α values vs Accuracy Score plot is shown in Fig.-8 and the α values vs F1 Score for the same is shown in Fig.-9. Clearly, it can be said that for unigram models, Multinomoal Naive Bayes is giving very poor performance as compared to other n-gram language models considered here. The Accuracy score as well as F1 Scores (for different α values) of Trigram and 4-gram language models are close to each other and decrease with higher values of α. However, in this experiment, 4-gram model has yielded highest Accuracy score for α=0.1 and highest F1 Score for α=1.

The Test Accuracy score and F1-score of the best model are **0.896** and **0.872** respectively. The AUC score of the

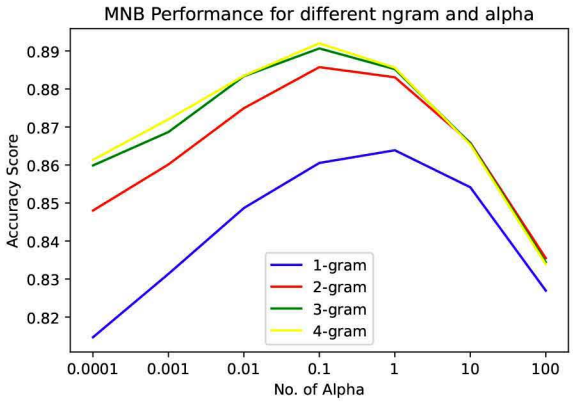

Fig. 8. α value - Accuracy Score plot of Multinomial NB with different n-gram range

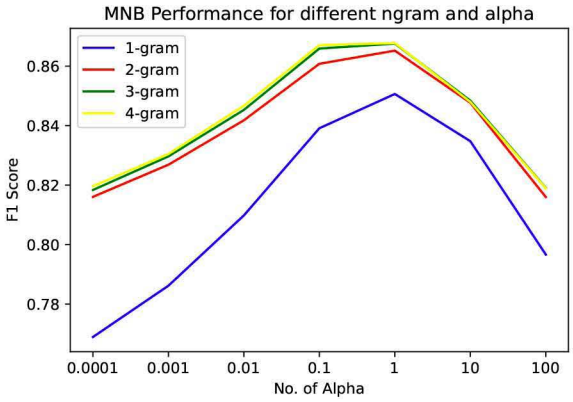

Fig. 9. α value - F1 Score plot of Multinomial NB with different n-gram range

best performing Multinomial Naive Bayes is **0.96** [Fig. 10].

Classifier	Accuracy Score	F1 Score	AUC Score
Linear SVM	**0.910**	**0.894**	**0.97**
Logistic Regression	0.903	0.893	0.97
Multinomial Naive Bayes	0.896	0.872	0.96

TABLE I
EXPERIMENTAL RESULTS

V. CONCLUSION

In this work, I have tried to present a comparative study of different traditional classification algorithms with different hyperparameter settings on the task of sentiment

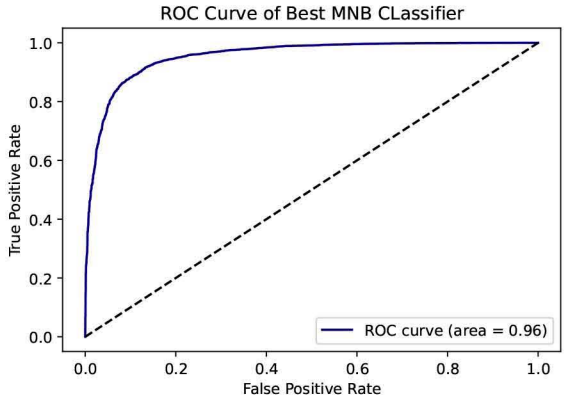

Fig. 10. ROC plot of Multinomial NB with AUC Score

analysis from IMDb Movie Reviews Dataset. The reviews are properly pre-processed so that it can be fed to these models. The final results were evaluated using 3 different evaluation metrics. From that, it can be concluded that in this problem, the combination of **TF-IDF with n-gram range (1, 2) + Linear SVM with C=10** has given highest accuracy score and F1-score.

In future, the experiment can be carried out using more advanced word embeddings such as Word2Vec which captures the semantic similarity between the words. Experiment can also be carried out to find if one classification algorithm can correctly classify the misclassified examples of another classifier. In this way, committee can be created which will be able to correctly predict all the examples, yielding higher performance.

REFERENCES

[1] S. Tripathi, R. Mehrotra, V. Bansal and S. Upadhyay, "Analyzing Sentiment using IMDb Dataset," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), 2020, pp. 30-33, doi: 10.1109/CICN49253.2020.9242570.

[2] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[3] T. P. Sahu and S. Ahuja, "Sentiment analysis of movie reviews: A study on feature selection & classification algorithms," 2016 International Conference on Microelectronics, Computing and Communications (MicroCom), 2016, pp. 1-6, doi: 10.1109/MicroCom.2016.7522583.

[4] M. Yasen and S. Tedmori, "Movies Reviews Sentiment Analysis and Classification," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), 2019, pp. 860-865, doi: 10.1109/JEEIT.2019.8717422.

[5] D. O. Ratmana, G. Fajar Shidik, A. Z. Fanani, Muljono and R. A. Pramunendar, "Evaluation of Feature Selections on Movie Reviews Sentiment," 2020 International Seminar on Application for Technology of Information and Communication (iSemantic), 2020, pp. 567-571, doi: 10.1109/iSemantic50169.2020.9234287.

[6] A. Poornima and K. S. Priya, "A Comparative Sentiment Analysis Of Sentence Embedding Using Machine Learning Techniques," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), 2020, pp. 493-496, doi: 10.1109/ICACCS48705.2020.9074312.