IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 37, No. 3, June, 2023, pp. 745-751
Journal homepage: http://iieta.org/journals/ria

A Comprehensive Machine Learning Framework for Automated Book Genre Classifier

Abhisek Sethy[1*], Ajit Kumar Rout[2], Archana Uriti[2], Surya Prakash Yalla[3]

[1] Department of Computer Science & Engineering, Silicon Institute of Technology, Bhubaneswar 751024, Odisha, India
[2] Department of Information Technology, GMR Institute of Technology, Rajam 532127, Andhra Pradesh, India
[3] Department of Computer Science & Engineering, MVGR College of Engineering, Chintalavalasa, Vizianagaram 535005, Andhra Pradesh, India

Corresponding Author Email: abhisek.sethy@silicon.ac.in

https://doi.org/10.18280/ria.370323

Received: 21 February 2023
Accepted: 1 April 2023

Keywords:
machine learning, natural language Toolkit, Naive Bayes, TF-IDF vectorizer, Random Forest, Gradient Boosting

ABSTRACT

Machine learning has been leveraged in the digital era, resulting in an increasing desire for computers to perform human-like tasks. Text classification is rapidly becoming one of the most significant applications of machine learning. However, the manual reading and classification of books based on genre requires substantial time and effort. As a result, machine learning methods are critical for enabling automated classification. In this study, a book description-based text classification framework was proposed, utilizing a wealth of information about book contents. The automated classification of books was achieved through the implementation of supervised machine learning. A variety of classifiers were employed, including Multinomial Naive Bayes, Gradient Boosting, and Random Forest, to categorize book genres. According to the results, the Naive Bayes classifier outperformed the other two techniques in classification accuracy, while comparable performance was achieved with Gradient Boosting and Random Forest. The comprehensive machine learning framework efficiently and accurately categorized books by extracting information from book descriptions. The proposed methodology has the potential to facilitate large-scale book classification for both academic and industrial purposes. Overall, this study provided an automated solution to relieve the burden of manual classification while achieving high accuracy.

1. INTRODUCTION

Machine learning enables computers to learn from previous experiences and improve future predictions without being explicitly programmed. Machine learning can address many real-time complex situations. Book genre classification is one of the most prevalent issues facing the world today. It is quite important in daily life. Humans can quickly identify a particular book's genre by reading its content. However, new readers may have difficulty determining the book's category, and there are many volumes in libraries that are not organized by genre. It is not always feasible for people to read the complete text to discern its genre because doing so requires substantial time and effort.

As a result, machine learning and natural language processing to address this prevalent problem could be quite beneficial. Natural language processing is commonly used in text classification and summarization challenges for application-oriented tasks. Gathering data, applying data pre-processing techniques, selecting features, and eventually implementing classifiers on the dataset are common steps in text classification using natural language processing [1]. The proposed method employed a labeled book dataset for testing purposes, as well as several classifiers such as Naive Bayes, Gradient Boosting, and Random Forest. The data pre-processing procedures and implementation of multiple machine learning classifiers were explained in this study. All classifiers were trained to categorize books into up to six distinct genres. Finally, the findings were reviewed and depicted using graphical representations, followed by a conclusion.

2. PRIOR WORK

In the study, Panchal et al. [2] used a variety of machine learning approaches on the books dataset. Text cleaning is done as part of pre-processing to eliminate tokens that are not important to categorization. Several classifiers were used, including KNN, SVM, and LR. To predict the genre of test samples, the above classifiers were trained using the feature matrix or values created using the TF-IDF vectorizer. However, one issue discovered is that all of the above-mentioned classifiers provide erroneous results and are unable to function well on bigger datasets. In the study, Gupta et al. [3] presented an Adaboost classifier approach to determine books based on genre. The tagged data from the gathered books is pre-processed through several phases. However, the feature matrix turned out to be sparse, which might prevent the model from correctly classifying. In order to solve this problem, the Principle Component Analysis approach is then used to minimise the dimensionality of the feature matrix. The Decision tree classifier was initially employed as a model, but because it was inaccurate, the Adaboost classifier was used to the training set to study it and it improved the accuracy of the Decision tree classifier model. Agarwal and Vijay [4] used a

Character Network-based Genre Classification in their study. The pre-processed text is run through a parts of speech (POS) tagger, which assigns all of the words to their appropriate parts of speech. All the same characters but referred to by various names were connected together once the text was POS tagged [5], and all the interactions between those characters were identified. The character interaction graphs for the texts in the dataset are then created utilising the information provided above. The edges were weighed using the interaction scores. The eigenvalues are then used as a comparison to determine how similar the graphs are. Thus, for each book text, the p closest graphs were picked, yielding a (nsamples × p) matrix, which was then fed into a variety of classifiers, including SVM, Multi-Layer Perceptron, Random Forest, Adaboost, Gradient Boosting, and Gaussian Naive Bayes, Gaussian Process Classifier. However, pronouns are sometimes difficult to determine and resolve in the context of a book, which is a huge disadvantage.

Li et al. [6] developed a text categorization model with the use of the Latent Dirichlet Allocation (LDA) approach. The information is initially obtained via twenty different news group data packets, which are a component that is widely utilised in classification algorithms. The authors decided to use three distinct varieties of 20 news group data packets for their research. After the data has been pre-processed, the LDA model is applied in order to determine the topic distribution of the news data used for the training collection. The distribution of topics is carried out with the assistance of a topic model in order to reduce the text dimension, which is too high, and to acquire features. After the training and modelling have been finished, the results of the anticipated categories are obtained with the use of a classifier called Softmax Regression. Yao et al. [7] demonstrated a fastText-based text classification model. This model included modules for data pre-processing, feature extraction, training, and assessment. Because the sample that was used in this research came from a Chinese source, the training samples that were used in the training phase of the method were all tokenized in advance. Once the word tokenization process is complete, the useless garbled letters, phrases, and stop-words are removed. This helps to improve the overall quality of the corpus. The input layer of the fast Text approach, in contrast to the typical bag-of-words model, takes into consideration the n-gram feature of the sentence in addition to the word forms for each word in the sentence as an additional feature to input. In addition to having elements that enable it to acquire word order information to a level, fastText also has features that enable it to generate a more accurate representation of a phrase. FastText combines word form and n-gram features from the input layer in the hidden layer and uses hierarchical softmax to discover the tag of the input data in the output layer. H-Softmax speeds training. Learning rate, epoch, window size, bucket, loss, and dim are hyper-parameters.

Zhang et al. [8] proposed the Mahalanobis-based KNN classifier. In this paper undergo pre-processing. Training simples initially filtered out stop words. Popular classifiers and learning algorithms cannot directly process text documents. During pre-processing, feature vectors represent documents. TF-IDF uses term and document frequency to compute a word's weight in a document and create a weight table. Document r's phrase frequency shows e's appearances. Document frequency is the number of e-containing documents. The TF-Gini feature weight approach by Shang [9] greatly increases classification performance. A promising text feature weighting technique. This algorithm computes no logarithms. Text feature weight methods require more calculation. Gini index non-purity split. SVM, kNN, and fkNN classifiers assessed the unique feature weight technique Gini Index. kNN finds the most similar (cosine similar) documents in training sets. It analyses the test document's candidate classes by their neighbours' classes. The test document's similarity determines the neighbor document's class weight. Fuzzy theory improved kNN. In the text pre-processing phase, each classifier was compared to the revised Gini index using Information Gain, Odds Ratio, Mutual Information, Expected Cross Entropy, Weight of Evidence of Text, and CHI.

Liu et al. [10] presented multi-hierarchy text categorization. It's Vector Space Model-based. This method encodes document content as a dot in multi-dimensional space and represents it as a vector. Calculating and comparing vector distances determines the vector's related classes. Most VSM documents are represented in the TF-IDF vector format, which calculates term weight using term frequency and inverse document frequency. This study explores the TF.IDF.IG approach and rationalises the term weight calculation algorithm to improve TF-IDF. VSM-based multi-hierarchy text classification is also demonstrated. All class training records are combined into a single class document as a tree based on hierarchical relationships. Building class models is as simple as comparing class documents related to the same layer node. Checking texts hierarchically until a relevant subclass is found. Roger Alan Stein et al. proposed hierarchical text categorization using word embeddings in the study [11]. With publicly available data, it employed classification models with fastText, SVM, XGBoost, Keras' CNN, GloVe, word2vec, and fastText. Text can be expressed numerically to become a vector. The Bag-of-words (BoW) model describes this representation. Data cleaning and homogenization usually precede BoW conversion from plain text. Weight is calculated using the compound value from TF and IDF.

3. PROPOSED METHODOLOGY

The proposed system has been established to perform a Book genre classification using various machine learning approaches [12] and all the essential stages has been depicted in below Figure 1.

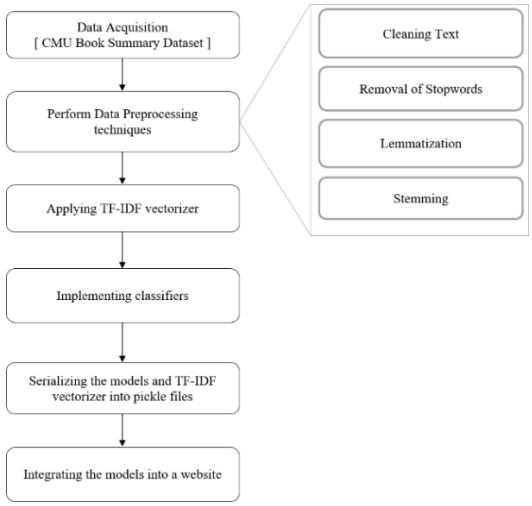

Figure 1. Steps implemented in proposed system

3.1 Data acquisition

Initially, a book dataset with a.csv file and around 3000 rows was obtained from the website [1]. In this paper the data acquisition has been collected CMU book summary dataset. About books that were converted from Gujarati or Hindi to English are in the dataset. The columns in the dataset are Book Id, Book Name, Summary, Authors, details, language, Abstarct, Summary and Genre. Among all these attributes, the proposed work has focused on four attributes like Book_id, Book_name, Genre and summary. All the rows included of each above book genre type. The collection included about books. Every 500 rows of the 3000 rows are the same genre, completing six different genres. Python is used for all of the programming in the proposed work. In Figure 2 depicts the process of reading a CSV file into the project environment, which is done with the help of the read_csv() function of the pandas library.

```
books = pd.read_csv('BooksDataSet.csv')
books
```

	Unnamed: 0	book_id	book_name	genre	summary
0	0	3248537	Drowned Wednesday	Fantasy	Drowned Wednesday is the first Trustee among ...
1	1	27796919	The Lost Hero	Fantasy	As the book opens, Jason awakens on a school ...
2	2	3910776	The Eyes of the Overworld	Fantasy	Cugel is easily persuaded by the merchant Fia...
3	3	5969644	Magic's Promise	Fantasy	The book opens with Herald-Mage Vanyel return...
4	4	3173445	Taran Wanderer	Fantasy	Taran and Gurgi have returned to Caer Dallben...
...	...	...	...	...	...
2995	2995	10372180	White Death	Thriller	A Novel from the NUMA files, A Kurt Austin Ad...
2996	2996	14504372	Venus with Pistol	Thriller	Gilbert Kemp is dealer specializing in antiqu...
2997	2997	3617412	Blackwater	Thriller	"How do you know when you're in too deep? Dav...
2998	2998	11320975	The Rainbow and the Rose	Thriller	The story concerns the life of Johnnie Pascoe...
2999	2999	17227674	Chiefs	Thriller	The First Chief: Will Henry Lee: The novel op...

3000 rows × 5 columns

Figure 2. Books dataset

3.2 Data preprocessing

Pre-processing [13] of data is done using libraries like pandas and numpy. The dataset being collected comprises columns that are redundant for genre classification, such as Book Id and Book Name, which have been removed from the dataset and staged for various preprocessing approaches as follows. In the preprocessing steps involves removal of punctuation and symbols in text followed up by removal of stop words that described in the below section.

3.2.1 Cleaning test

In order to classify the book appropriately according to its category, the proposed need to read an overview of it. However, the synopsis of each book in our collection contains unwanted characters such as punctuation and other special symbols that aren't required for classification [14]. As a consequence of this, the proposed system made use of a significant number of the built-in Python functions in order to clean up the text. All the steps involved in the data cleaning are listed below.

3.2.2 Removal of stopwords

The cleansed data is next pre-processed by removing stopwords, which are useless for data classification. The elimination of stop words is one of the preprocessing stages that is utilized the vast majority of the time in a variety of applications that utilize NLP. The objective is to simply get rid of the words that are used regularly throughout all of the different documents that make up the database. Stop words are typically comprised of grammatical constructs such as articles and pronouns [15] like "the", "and" and "to" in English, for example, which are inappropriate for categorization and cannot be employed. So the nltk.corpus package is imported to get all the English stopwords and delete them from all summaries by scanning through entire text and depicted in Figure 3.

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
def removestopwords(text):
    removedstopword = [word for word in text.split() if word not in stop_words]
    return ' '.join(removedstopword)
books['after_stopwords_removal'] = books['summary'].apply(lambda x:removestopwords(x))
books.head(3)
```

	book_id	book_name	genre	summary	after_stopwords_removal
0	3248537	Drowned Wednesday	Fantasy	drowned wednesday is the first trustee among t...	drowned wednesday first trustee among morrow d...
1	27796919	The Lost Hero	Fantasy	as the book opens jason awakens on a school bu...	book opens jason awakens school bus unable rem...
2	3910776	The Eyes of the Overworld	Fantasy	cugel is easily persuaded by the merchant fian...	cugel easily persuaded merchant fianosther att...

Figure 3. Removal of stopwords

3.2.3 Lemmatization

Lemmatization is a process that attempts to reduce a word to its root form, which is also known as a lemma. As an illustration, the verb "running" would be rendered simply as "run." The study of lemmatization focuses on the morphological, or structural, analysis of words in addition to their context.A word is analysed by the process of lemmatization, which reduces it to its lemma. Following the elimination of stopwords, the text is lemmatized [16], which reduces all versions of a given word to their simplest form. The nltk.stem library, which includes the WordNetLemmatizer class, is imported here. All of the terms are lemmatized to their basic forms using the lemmatize() function and have shown in Figure 4.

```
from nltk.stem import WordNetLemmatizer
lemma=WordNetLemmatizer()
def lematizing(sentence):
    stemSentence = ""
    for word in sentence.split():
        stem = lemma.lemmatize(word)
        stemSentence += stem
        stemSentence += " "
    stemSentence = stemSentence.strip()
    return stemSentence
books["lemmatized_text"]=books['summary'].apply(lambda x: lematizing(x))
books[["summary","lemmatized_text"]].iloc[1]
```

```
summary           book opens jason awakens school bus unable rem...
lemmatized_text   book open jason awakens school bus unable reme...
Name: 1, dtype: object
```

Figure 4. Lemmatization

3.2.4 Stemming

Stemming is a method of eliminating attachments from the word in order to ensure we are left with the word's primary

component, which is called the stem. After having stemming performed on them, the words 'run,' 'running,' and 'runs' can all be reconstructed from the root word 'run,' as seen in the following example. One of the most important aspects of stem words is that it is not necessary for them to have any meaning. After that, the text is subjected to a stemming process in order to acquire the base form of all words by eliminating affixes. The library nltk.stem, which contains the Stemmer class, is imported to accomplish this. The stem() method is used to execute a text stemming procedure and shown in Figure 5.

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
def stemming(sentence):
    stemmed_sentence = ""
    for word in sentence.split():
        stem = stemmer.stem(word)
        stemmed_sentence+=stem
        stemmed_sentence+=" "
    stemmed_sentence = stemmed_sentence.strip()
    return stemmed_sentence
books['stemmed_text'] = books['summary'].apply(lambda text:stemming(text))
books[['summary','stemmed_text']].iloc[1]
```

```
summary         book open jason awakens school bus unable reme...
stemmed_text    book open jason awaken school bu unabl rememb ...
Name: 1, dtype: object
```

Figure 5. Lemmatization

3.3 Applying TF-IDF vectorizer

In order to understand how term frequency (TF) works, you must first examine the frequency of a certain phrase that you are interested in in relation to the document. There are many different standards or approaches to defining frequency, including the unrounded count of the number of times a word appears in a document. The raw count of occurrences is divided by the total number of words in the document, and the result is the term frequency adjusted for the length of the document. The summary of any book in the dataset is of the type string, and it should not be put into the model directly. The input to ML models is solely in the form of numerical representations which is doable with the TF-IDF vectorizer [17]. It's a method for converting text into finite-length vectors that's mostly used to normalize data. The two terms that it is constructed on are Term Frequency (TF) and Inverse Document Frequency (IDF). The following formula is used to find TF-IDF values shown in Eq. (1), Eq. (2) and Eq. (3).

$$tf_{idf} = Term\ Frequency * Inverse\ Document\ Frequency \quad (1)$$

Term frequency (TF)=It is a statistic that counts the number of times a word (x) appears in a document (y).

$$\mathrm{TF(x.y)} = \frac{\text{number of instances of x in document y}}{\text{whole number of words in document y}} \quad (2)$$

Inverse Document Frequency (IDF) It's a statistic for assessing a word's importance. The Term frequency (TF) method overlooks the importance of terms. The word's IDF is measured as follows

$$\mathrm{IDF(x,C)} = \ln\left(\frac{\text{whole number of documents contained in corpus C}}{\text{whole number of documents that contain x}}\right) \quad (3)$$

After substituting Eq. (2) and Eq. (3) in Eq. (1), we get

$$\mathrm{tf-idf} = \mathrm{TF(x,y)} \times \mathrm{IDF(x,C)} \quad (4)$$

Instead of manually computing the TF-IDF values, the library sklearn.feature_extraction.text, which contains the TFidfVectorizer class, is used to automate the process. As a result, in Eq. (4), the feature matrix is created to represent the original summary text in numerical form, and this matrix together with the target variable "genre" is supplied into the model as an input.

3.4 Implementing classifiers

3.4.1 Naïve Bayes (NB)

The Naive Bayes classifier [18] is widely used in classification tasks, such as text categorization, and is a prominent supervised machine learning technique. It's a machine learning algorithm; therefore, it simulates how a class's inputs are typically distributed. The speed and accuracy of the algorithm's recommendations are predicated on a belief that the features of the input data are conditionally independent given the class. Given a set of facts and some background knowledge, this theorem calculates the probability of a hypothesis. The inexperienced Bayes classifier makes an inference that the input data's features are unrelated to one another, which is rarely the case in reality. It is mainly used for classification problems. It's a probabilistic learning approach, which means it makes predictions based on the likelihood of the object. The Bayes theorem formula is

$$P(X/Y) = \frac{P(Y/X) \times P(X)}{P(Y)} \quad (5)$$

where, P(X) = prior probability
P(X/Y) = posterior probability
P(Y/X) = likelihood probability
P(Y) = Marginal probability

It computes the event A's probability given that event B already occurred. Using the MultinomialNB in Eq. (5) class from the sklearn.naive_bayes package, the classifier instance is first generated. For multi-class text classification, the Multinomial Naive Bayes classifier is appropriate. The feature matrix, together with the encoded target variable "genre" is then input into the Naive Bayes Classifier model using the fit() method for training and depicted in Figure 6.

```
from sklearn.naive_bayes import MultinomialNB
mb = MultinomialNB()
mb.fit(xtrain_tfidf,y_train)
```

Figure 6. Implementing Naive Bayes classifier

3.4.2 Random Forest (RF)

Random Forest [19] is a popular supervised Machine Learning algorithm which can be used for both regression and classification problems. When there is a labeled target variable,

Random Forests can be used for supervised machine learning. Both regression (with a numeric target variable) and classification (with a categorical target variable) issues are amenable to Random Forests. As an ensemble method, Random Forests pool the results of various models to get an overall forecast. Individually, the component models that make up the Random Forest ensemble are decision trees. First, "bagging" generates a new training subset from existing training data samples using a majority voting system. Take Random Forest as an illustration. The second method, called "boosting," mixes models in a sequential fashion to increase the accuracy of the final model. Such brands include ADA BOOST and XG BOOST. It uses ensemble learning where it uses a combination of classifiers (decision trees) to predict the final result taken based on max voting. It is a special case of bagging in which the Bootstrap Sampling is the core idea in the bagging technique. This algorithm is very helpful for solving complicated problems [17] and in turn, the model's performance would be greatly improved.

Random Forest Classifier model object is imported from the sklearn.ensemble package to construct Random Forest. The fit() method is then used to train the model using the feature matrix and encoded target variable, show in Figure 7.

```
from sklearn.ensemble import RandomForestClassifier
clf=RandomForestClassifier(n_estimators=100)
clf.fit(xtrain_tfidf,y_train)
```

Figure 7. Implementing Random Forest classifier

3.4.3 Gradient Boosting classifier

It's a Boosting technique that uses an ensemble learning method. It combines numerous weak classifiers into a powerful model (a good accuracy model). Each model tries to fix the flaws of the previous one. At each iteration, it does not fit a predictor to the data, instead the new predictor [20] is fitted upon the residual bias given by the previous predictor. In especially when dealing with huge and complicated datasets, the method known as "Gradient Boosting" stands out because to the speed and precision with which it makes predictions. Errors play a significant part in every machine learning system, which is something we already know. Error can be broken down into two primary categories: bias error and variance error. The gradient boost approach assists us in reducing the amount of bias error produced by the model. Gradient Boosting Classifier class imported from the sklearn.ensemble package is used to implement Gradient Boosting in Python. The feature matrix and encoded target variable, which are supplied as input to the model, are then utilised to train the model using the fit() function and shown in Figure 8.

```
from sklearn.ensemble import GradientBoostingClassifier
gradient_booster = GradientBoostingClassifier(learning_rate=0.1)
gradient_booster.fit(xtrain_tfidf, y_train)
```

Figure 8. Implementing Gradient Boosting classifier

3.5 Serializing the models and TF-IDF vectorizer into pickle files

The models established in the preceding phase are in the form of Python objects. In order to use the models created in the real websites, they must be stored as pickle files. Pickle is Python's standard serialization and de-serialization mechanism. Pickling is the process of transforming any Python object into byte streams. Unpickling is the process of converting byte streams back into Python objects. It's the pickling process in reverse. By importing the pickle library, Python objects are serialized to pickle files.

3.6 Integrating the models into a website

In the project built, the pickle files generated following the pickling phase are used. Unpickling is accomplished by loading the pickle files into a website and re-importing the same models and shown in Figure 9.

```
#importing the model
file = open('bookgenremodel.pkl', 'rb')
model = pickle.load(file)
file.close()

#importing the TF-IDF vectorizer
file1 = open('tfdifvector.pkl', 'rb')
tfidf_vectorizer = pickle.load(file1)
file1.close()
```

Figure 9. Integrating the models into a website

4. RESULTS AND DISCUSSIONS

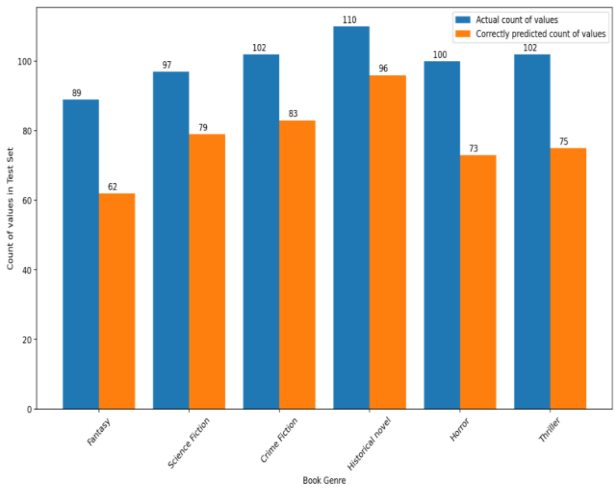

Figure 10. Naive Bayes classifier performance

The 3000 row books dataset is divided into 80:20 training and testing sets, with the test set containing 600 rows. The graphs below depict the working of several classifiers on the Test dataset. The entire feature has been evaluated with respect to various classifiers. All the results have been evaluated based on six class type of Book genre. Among the datasets certain amount of data or rows has been taken for the classification of Book category. In below Figure 10 depicts the analysis with

respect to NaiveBayes, Figure 11 depicts the analysis with respect to Random Forest [21], and Figure 12 depicts the analysis of Gradient Boosting approach.

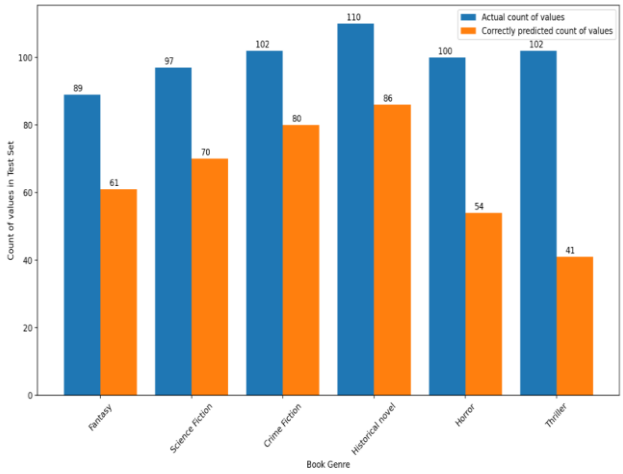

Figure 11. Random Forest classifier performance

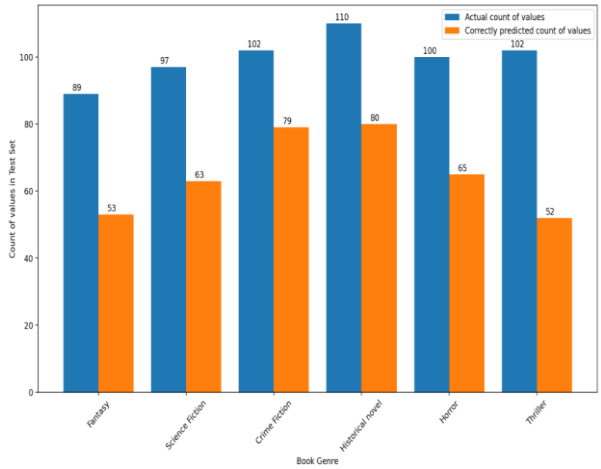

Figure 12. Gradient Boosting classifier performance

The accuracy of several classifiers is tabulated below. By looking the results, the Naive Bayes classifier clearly seems to be better than that of the other classifiers [19] and comparison over all techniques has been listed in the Table 1 below.

Table 1. Accuracies of various classifiers

Model	Training Set (%)	Test Set (%)	CMU Book Accuracy Classification
Naive Bayes	80	20	78%
Random Forest	80	20	65.66%
Gradient Boosting	80	20	65.33%

5. CONCLUSIONS

Text classification is becoming an increasingly widespread challenge in modern times, and it is challenging for people to discover solutions. As a result, the use of machine learning in this domain is a viable option for simplifying the situation. Therefore, this proposed classification scheme makes use of a variety of distinct classifiers in order to categorize the types of books. The text is pre-processed first, and then it is turned into feature vectors. Finally, the feature vectors are input into the multiple classifiers, where they are trained. According to the findings, Naive Bayes performs significantly better than the other two classifiers, which are Random Forest and Gradient Boosting.

REFERENCE

[1] Bejan, A. (2015). Constructal thermodynamics. Constructal Law & Second Law Conference, Parma, pp. S1-S8.

[2] Panchal, B.Y., Shiroya, P., Vaghasiya, D., Soni, M. (2021). Book Genre Categorization Using Machine Learning Algorithms (K-Nearest Neighbor, Support Vector Machine and Logistic Regression) Using Customized Dataset. http://dx.doi.org/10.47760/ijcsmc.2021.v10i03.002

[3] Gupta, S., Agarwal, M., Jain, S. (2019). Automated genre classification of books using machine learning and natural language processing. In 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, pp. 269-272. https://doi.org/10.1109/CONFLUENCE.2019.8776935

[4] Agarwal, D., Vijay, D. (2021). Genre classification using character networks. In 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, pp. 216-222. https://doi.org/10.1109/ICICCS51141.2021.9432303

[5] Ozsarfati, E., Sahin, E., Saul, C.J., Yilmaz, A. (2019). Book genre classification based on titles with comparative machine learning algorithms. In 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), Singapore, pp. 14-20. https://doi.org/10.1109/CCOMS.2019.8821643

[6] Li, Z., Shang, W., Yan, M. (2016). News text classification model based on topic model. In 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), Okayama, Japan, pp. 1-5. https://doi.org/10.1109/ICIS.2016.7550929

[7] Yao, T., Zhai, Z., Gao, B. (2020). Text classification model based on fasttext. In 2020 IEEE International Conference on Artificial Intelligence and Information Systems (ICAIIS), Dalian, China, pp. 154-157. https://doi.org/10.1109/ICAIIS49377.2020.9194939

[8] Zhang, S., Pan, X. (2011). A novel text classification based on Mahalanobis distance. In 2011 3rd International Conference on Computer Research and Development, 3: 156-158. https://doi.org/10.1109/ICCRD.2011.5764268

[9] Shang, W., Dong, H., Zhu, H., Wang, Y. (2008). A novel feature weight algorithm for text categorization. In 2008 International Conference on Natural Language Processing and Knowledge Engineering, Beijing, China, pp. 1-7.

[10] Liu, S., Dong, M., Zhang, H., Li, R., Shi, Z. (2001). An

approach of multi-hierarchy text classification. In 2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No. 01EX479), 3: 95-100.

[11] Stein, R.A., Jaques, P.A., Valiati, J.F. (2019). An analysis of hierarchical text classification using word embeddings. Information Sciences, 471: 216-232.

[12] Sethy, A., Patra, P.K., Nayak, S.R. (2022). A hybrid system for handwritten character recognition with high robustness. Traitement du Signal, 39(2): 567-576. http://dx.doi.org/10.18280/ts.390218

[13] Sethy, A., Patra, P.K., Nayak, S.R., Poonia, R.C. (2022). Offline handwritten character and numeral recognition: A kernel-based approach. International Journal of Social Ecology and Sustainable Development (IJSESD), 13(1): 1-21. http://dx.doi.org/10.4018/IJSESD.295087

[14] Sethy, A., Patra, P.K., Nayak, S.R. (2022). A deep convolutional neural network-based approach for handwritten recognition system. In Computational Intelligence in Pattern Recognition, Singapore, pp. 607-617. http://dx.doi.org/10.1007/978-981-16-2543-5_52

[15] Uriti, A., Yalla, S.P., Chintada, K.R. (2021). An approach of understanding customer behavior with an emphasis on rides. In 2021 Innovations in Power and Advanced Computing Technologies (i-PACT), Malaysia, pp. 1-5. https://doi.org/10.1109/i-PACT52855.2021.9696837

[16] Yalla, S.P., Uriti, A., Sethy, A. (2022). Wheel chair movement through eyeball recognition using raspberry Pi. Specialusis Ugdymas, 1(43): 8583-8591.

[17] Rout, A.K., Sethy, A., Kumar, M.R., Ahamed, M.F., Mohan, S. (2022). Text summarization adaptive models for semantic relevance information: A survey. Specialusis Ugdymas, 1(43): 10836-10844.

[18] Archana, U., Sridhar, U. (2017). A novel quantization approach for approximate nearest neighbor search to minimize the quantization error. JIRSET, pp. 11976-11982. https://doi.org/10.15680/IJIRSET.2017.0606287

[19] Yalla, S.P., Uriti, A., Sethy, A. (2022). GUI implementation of modified and secure image steganography using least significant bit substitution. International Journal of Safety and Security Engineering, 12(5): 639-643. https://doi.org/10.18280/ijsse.120513

[20] Rout, A.K., Sethy, A., Nayak, S.R. (2022). Adaptive MLELM-AE model for efficient prediction of stock market data. Journal of Statistics and Management Systems, 25(7): 1541-1552. http://dx.doi.org/10.1080/09720510.2022.2130567

[21] Sethy, A., Patra, P.K. (2018). Optical character recognition of Odia handwritten scripts and numerals: A survey on web based utility application. Journal of Web Engineering, 17(6): 3629-3653.