# Multilevel Classification of Pakistani News

**L1F23MSDS0005**[1]

[1]Department of Computer Science or Data Science, Faculty of Information Technology & Computer Science, University of Central Punjab, Lahore, Pakistan

## ABSTRACT

The public has benefited from the abundance of online news sources since it allows them to obtain information from a wide range of sources. It has never been easy to classify the massive amounts of data that are produced on a regular basis, though. Only when the text is handled to optimize its utility—a feat made feasible by automatic text classification—can this textual data be considered insignificant. In this specific field, a lot of work has been done using machine learning and natural language processing (NLP) techniques to address this difficulty. There are various situations where text classification comes in handy, including sentiment analysis, emotions, and product mining. One of its uses is news categorization, whereby news content is evaluated and processed to assign predetermined label(s).The classification of Pakistani news derived from a dataset made available on Open Data Pakistan is the main topic of this study. I use 3 ML algorithm on this pakistani news classification task on data obtain from Open Data Pakistan website.I use Navive Bayes, Random Forest Classifier and Adaboost algorithm. Comparative analysis of these algorithms is also presented. We achieved a maximum of 98.9Navive Bayes in single-level classification and 83classification.

## Introduction

There is a growing need for effective automatic classification due to the abundance of news and information sources. It is incredibly difficult, time-consuming, and practically impossible to manually classify or cluster documents based on their content. Many text processing domains, including sentiment analysis, text summarization, document clustering, and classification, heavily rely on NLP and machine learning approaches. Research and studies that are pecific can benefit from text classification. Real-time public opinion analysis and interpretation can be highly beneficial, particularly in the context of social media and internet platforms. The classification process differs depending on the sources and kind of text being used.By way of example, there will be variations in the text classification method used for data scraped from any website and in the classification of plain data from any site that collects datasets. The approach will change while the aim will stay the same.

While news is often categorized by categories or belonging to a certain location, texts can also be categorized based on the typical places they are associated with, which makes them helpful.Text retrieval, information abstraction and summarization, and question answering all heavily rely on text classification. The data that are typically found on the internet and utilized for classification come from a variety of sources, including printed or broadcast news, message boards, newsgroups, ads, and movie reviews. Automatic text classification is crucial due to their heterogeneous character,which includes being multi-sourced, possessing distinct vocabularies, formats, and writing styles for texts[1]. Unstructured text rising mostly the information is available in digital form for example world wide web, e-mail,release, etc. A word and a sentence ambiguity may include in an actual form or sequence occurred in that text Any kind of data is referred to as unstructured text. As this text in a row for information extraction that is helpful is being used Techniques for processing and pre-processing are necessary. As the In a computer, the processing method is not applicable. As the the text is being used by the computer as a string order without any kind of data. for improved categorization in each domain There are several text resources accessible. These include the fields of medicine, finance, image processing, and many more, where the main goal of text mining is to use techniques like supervised or unsupervised classification or Natural Language Processing to extract useful information from semi-structured or unstructured text. It is well known that all conventional natural language processing algorithms primarily use words to determine predetermined classifications for specific texts or text documents[2].

Articles are categorized by topics, including technology, sports, politics, entertainment, and business. Users may save time and cut down on news overhead by using the news classification system to locate news pieces that pique their attention[3].For generalists in the news industry, this would be an excellent automatic system to classify news. Commonly used algorithms for news classification are logistic regression, random forest, support vector machine, k-Nearest Neighbor, Naïve Bayesian s, and decision tree. included while I am using CNN and execute Algorithm in my experiment.First, let's discuss word embeddings. In order to look at the review as a whole rather than concentrating on the influence of each individual word, we need to take into account word similarity in various reviews when using Naive Bayes and KNN. Previously, we represented our text as a vector and ran the algorithm on that vector. We make advantage of a pre-made word embedding that the library has to provide. In

general, there are several other open-source embeddings accessible, such as Glove and Word2Vec, if the data is not embedded. Even though they may be in the same class, the results of a dot product of text vectors may be zero; nevertheless, if you do a dot product of the embedded word vectors to determine their similarity, you will be able to determine the relationships between the words for a given class. further, we apply the filter or kernel to these embeddings to obtain convolutions, which are further dimensionally reduced to minimize complexity and computation via the Max Pooling layer. Activation functions on the outputs that will provide values for each class and completely linked layers are the final components.
A portion of the 27000-record dataset that is accessible on Open Data Pakistan was used by us. The dataset was selected using news information that was collected from Pakistan's leading English newspapers that were published between January 2020 and March 2021.[4]. Urdu news headlines, the majority voting classification of Urdu text, the classification of Sindhi headlines, and the recognition of named entities in Pashto. Handwritten text identification and categorization for Pashto is another aspect of this classification process. Along with text classification, query processing, news item categorization, and parts of speech tagging, work is being done in areas such as sentiment analysis[4].

## Background and Related Work

Scholars have recognized the challenge of categorizing with uneven data and have developed a variety of potential answers and applications. When working with uneven data, classifying using machine learning techniques might be difficult. In order to conduct their experiments, the study's authors used an unstructured dataset that they obtained from the human resources department. This dataset contained job description material that was significantly skewed toward distinct classes. All of the papers are divided into segments in order to extract relevant information for conversion into structured data. In addition, a number of popular techniques are employed to convert text documents into numerical representations. Experiments were carried out on many text demonstrations utilizing different classification algorithms inside a well-balanced scheme to address the problem of imbalanced text data to varying degrees. .. They suggested a cost-sensitive strategy in which a Differential Evolution algorithm is used to calculate the expenses. The strategy was put into practice in two stages: cost-sensitive learning and resampling techniques. They are first produced at the data instance level following class-level cost adjustments[5].
Deep learning (DL) has emerged as a prominent global research trend in recent years, attracting the attention of academic groups worldwide. Many sectors, including text processing, healthcare, education, and agriculture, have used deep learning (DL) in their operations. DL technology has been utilized by certain academics to extract text feature information. For instance, when extracting text characteristics using conventional ML methods, the issues of data dimension expansion and excessive sparsity are difficult to manifest. The word2vec model, which offers technical support for word vector conversion, was initially suggested by Longfeng. Simultaneously, other researchers advocated using neural networks as classification models. For instance, Alswaidan and Menai generated word vectors using the word2vec model and subsequently learned feature information from CNN. In the process of researching the pre-Qin classics, long short-term memory (LSTM) is utilized to examine public opinion emotion of emergency network based on the word vector. They started by building the classification system. Next, they used TF-IDF to represent the text features, which they then fed into the Bi-LSTM (bi-directional LSTM) model. The outcomes of the experiment demonstrate that the DL approach had a greater impact than the ML approaches. The accuracy of the DL methods is much higher than that of the conventional ML approaches[6].
Yoon (2004)'s CNN improved upon the original CNN proposal for image processing by making it more useful for phrase categorization. In the basic Text CNN mode (Yoon, 2014), the word vector acquired by an unsupervised neural language model is designed with a layer of convolution on top of it. The initial word vector is kept static, and only the model's other parameters are learned. Nevertheless, the Word2vec model does not take into account the relationship between the feature word and the document as a whole; rather, it only takes into account the semantic relationship between the feature word and its fixed-length context. The research (Song et al., 2019) generated the document vector by combining the word2vec word vector with the TF-IDF method, which determined the weight of each word in the text. The significant connection between the feature word and the document is developed based on word frequency. However, based just on word frequency as the weight, the produced document vector lacks sufficient accuracy. When utilizing word2vec to train word vectors, the study (Tian  Wu, 2018) took into account the semantic significance of the distance between words. However, because additional factors like word frequency and location throughout the full article are ignored, the generated word vector is still insufficiently precise.
Word2vec generates word vectors with a more acceptable weighting, which can increase text categorization efficiency. By turning location data into weights, the pooling layer might be improved, potentially increasing the accuracy of text categorization, as the titles and opening paragraphs of most news stories frequently contain significant elements that can more correctly reflect the topic of the text[7]. Classification issues are widely used in text filtering, indexing, document organization, sentiment classification, and other fields. They have been thoroughly examined in fields such as data mining and information retrieval by utilizing machine learning techniques. Simple tasks like feature engineering, feature selection, and document representation should be completed before beginning any text categorization work. Traditional text classification techniques employ machine learning algorithms such as logistic regression, support vector machines, and naive Bayes classifiers to develop models

using a bag of words as feature representation. These approaches, however, suffer from problems with data sparsity. Recent developments in neural network models and deep learning have demonstrated notable efficacy in addressing sparsity concerns. Many context-sensitive neural network models for feature representation from unprocessed text have been presented in the literature.

## Data and Materials

I have followed this paper in which this data set is used secondarily and I am also not using primary data set but using the same data set as secondary data set and this data set I have obtain from open data pakistan.I have taken it from the Pakistan website and it has four columns and there are 27,000 records in addition to the four columns, but this data is all internal not structure data and I have cleaned it myself and cleaned this data. I have again created a new data set and in this data set there were instruction objects which made the data completely useless but in this data there are 2100 records out of 27 thousand which are correct records. I have downloaded my data set from the website, this data set has total 27 thousand records and has four columns out of which the following column is Story Heading Story , Story Excerpt, Time Stamp, and Section.This total data consists of 27 thousand records out of which 2100 are almost record labels which have four categories in which Pakistan category total is 963, Newspaper 617, world 393 and Sports 43. Then when I separated the records of 2100, in it I found this listed category that Pakistan, World, Support, Business, SP Supplements News, Prism, Tech, Multimedia, and Corona Virus this category was present out of which 596 of Pakistan's 920 Newspapers. World has 381 support, 42 business, 24 , 16 supplements, 15 prism, four multimedia, and one category of Corona virus. This entire data set, which consists of thousands of records, I pre-processed and I trained the model on it.

## Proposed Methodology

Importing Libraries: The necessary libraries are imported to support various functionalities throughout the code. Here's a brief explanation of each library: pandas (imported as pd): A powerful data manipulation library used for data analysis and manipulation. numpy (imported as np): A library for numerical computing that provides support for large, multi-dimensional arrays and matrices. matplotlib.pyplot (imported as plt): A plotting library used for creating visualizations, such as charts and graphs. seaborn: A data visualization library that provides a high-level interface for drawing attractive and informative statistical graphics. itertools: A module that provides various functions for efficient iteration and combination of data. sklearn.model selection: A module that provides functions for splitting data into training and testing sets. sklearn.feature extraction.text: A module that provides utilities for extracting features from text data. sklearn.metrics: A module that provides functions for evaluating the performance of machine learning models.
Data Loading and Preprocessing: The code loads the dataset from a CSV file called "data.csv" using the pd.read csv function. It then checks for missing values in the dataset using the isna().sum() method. Any rows with missing values are dropped using dropna(inplace=True). The "Timestamp" column is dropped from the dataframe using drop('Timestamp', axis=1, inplace=True).
Data Filtering: The code filters the dataset by removing rows with specific values in the "Section" column. It uses the drop function to remove rows with sections such as 'Coronavirus', 'Business', 'Sp Supplements', 'Tech', 'Multimedia', 'Prism', and 'Sport'.
Label Encoding: The code uses scikit-learn's LabelEncoder to encode the labels in the "Section" column. The LabelEncoder is fitted on the "Section" column using label encoder.fittransform(df['Section']). This step assigns a unique numerical value to each unique section.
Text Preprocessing: The code performs several text preprocessing steps to clean and normalize the text data. Here's a breakdown of the preprocessing steps:Word Count: The code calculates the word count for each "Story Heading" and adds the values as a new column called "wordcount". Preprocessing Function: The code defines a preprocessing function called preprocess that converts text to lowercase, removes HTML tags, removes punctuation, removes special characters, removes digits, and reduces multiple spaces to a single space. Stopword Removal: The code defines a function called stopword that removes stopwords (commonly used words like "the", "and", "is", etc.) from the text. Lemmatization: The code defines a function called lemmatizer that uses part-of-speech tagging to lemmatize the words (reduce them to their base or dictionary form). Final Preprocessing: The code defines a function called finalpreprocess that applies the preprocessing steps in a specific order: lemmatization, stopword removal, and preprocessing. The cleaned text is stored in a new column called "cleantext".
TF-IDF Vectorization: The code uses scikit-learn's TfidfVectorizer to convert the cleaned text data into a numerical representation suitable for machine learning algorithms. The TfidfVectorizer converts a collection of raw documents to a matrix of TF-IDF features. It considers the frequency of each word in a document and the inverse document frequency across the entire corpus. The resulting TF-IDF matrix is obtained by calling fit.transform on the "cleantext" column. Splitting the Data: The code splits the dataset into training and testing subsets using the train-test-split function from scikit-learn. It assigns 70 percent

of the data to the training set (X-train and y-train) and 30 percent to the testing set (X-test and y-test). Model Training and Evaluation: The code trains three different classifiers on the training data and evaluates their performance on the testing data. Here's a summary of each classifier:

Naive Bayes: The code trains a Naive Bayes classifier (MultinomialNB) on the training data and predicts the labels for the testing data. It calculates classification metrics such as accuracy and classification report for evaluating the model's performance. Random Forest: The code trains a Random Forest classifier (RandomForestClassifier) on the training data and predicts the labels for the testing data. It also calculates classification metrics, including accuracy and classification report. AdaBoost: The code trains an AdaBoost classifier (AdaBoostClassifier) on the training data and predicts the labels for the testing data. Similarly, it calculates classification metrics like accuracy and classification report. Results: The code prints the accuracy and classification report for each classifier, providing insights into the performance of each model for the text classification task. i got maximum 98.9 percent from Navive Bayes algorithm.

Term Frequency (TF): Term Frequency measures the frequency of a term (word) in a document. It represents how often a term occurs in a document relative to the total number of terms in that document. It is calculated using the formula:

TF(t, d) = (Number of occurrences of term t in document d) / (Total number of terms in document d)

The TF value is higher if a term appears more frequently in a document.

Inverse Document Frequency (IDF): Inverse Document Frequency measures the importance of a term across a collection of documents. It represents how rare or common a term is in the entire document collection. IDF is calculated using the formula:

IDF(t) = log((Total number of documents) / (Number of documents containing term t))

The IDF value is higher if a term is less common across the document collection.

TF-IDF Calculation: The TF-IDF value is obtained by multiplying the Term Frequency (TF) of a term in a document by its Inverse Document Frequency (IDF) across the entire document collection. The TF-IDF value highlights terms that are both frequent in a specific document and rare in the overall document collection.

TF-IDF(t, d) = TF(t, d) * IDF(t)

The TF-IDF value is higher for terms that are important within a document but relatively rare in the entire document collection.

Vectorization: To convert text into a numerical representation, TF-IDF vectorization assigns each term in the corpus a unique index position. It then represents each document as a vector, where each element corresponds to the TF-IDF value of a specific term in that document.

The TF-IDF vectorization process typically involves the following steps:

Tokenization: Breaking down the text into individual terms (words). Counting: Counting the frequency of each term in a document (Term Frequency). Calculating IDF: Calculating the Inverse Document Frequency for each term across the entire document collection. Calculating TF-IDF: Multiplying the Term Frequency by the Inverse Document Frequency for each term in each document. Normalizing: Scaling the TF-IDF values to ensure that documents of different lengths are comparable.

## Experimental Setups

To conduct experiments on the text classification project using the provided code, you can follow the following experimental setup:

Dataset: Start by selecting or preparing a suitable dataset for text classification. Ensure that the dataset contains a sufficient number of documents with labeled sections or categories. You can use the provided "data.csv" file or replace it with your own dataset.

Data Preprocessing: Apply necessary data preprocessing steps to clean and transform the text data. This may include removing HTML tags, punctuation, special characters, and digits. Lowercase the text, perform lemmatization, and remove stopwords. Ensure that the dataset is in the required format for further processing.

Train-Test Split: Split the preprocessed dataset into training and testing subsets. The typical split ratio is 70

Feature Extraction: Apply TF-IDF vectorization to convert the preprocessed text data into a numerical representation. Use scikit-learn's TfidfVectorizer for this step. Consider adjusting the vectorization parameters such as n-grams, maximum features, and token patterns to optimize the representation of your text data.

Model Selection: Choose the machine learning models you want to evaluate for text classification. The provided code demonstrates the usage of Naive Bayes, Random Forest, and AdaBoost classifiers. However, you can experiment with other models such as Support Vector Machines (SVM), Logistic Regression, or Deep Learning models (e.g., Recurrent Neural Networks, Transformers) to compare their performance.

Model Training and Evaluation: Train the selected models on the training data and evaluate their performance on the testing data. Calculate metrics such as accuracy, precision, recall, and F1 score to assess the models' classification performance. Use scikit-learn's fit and predict functions to train the models and make predictions.

Hyperparameter Tuning: Optionally, perform hyperparameter tuning for the selected models to optimize their performance. Use techniques like grid search or random search to find the best combination of hyperparameters for each model. This step can help improve the models' performance by fine-tuning their configurations.

Results and Analysis: Analyze the experimental results to compare the performance of different models. Consider factors such as accuracy, precision, recall, and F1 score to assess the models' effectiveness in classifying the text data. Visualize the results using plots, such as confusion matrices or ROC curves, to gain deeper insights into the models' behavior.

Iteration and Refinement: Iterate on the experimental setup by trying different variations, such as using different preprocessing techniques, adjusting the vectorization parameters, or exploring alternative models. This iterative process helps in refining the text classification pipeline and improving the overall performance.

## Findings and Discussions

Your manuscript's **Findings and Discussions** section is where you present and discuss the findings of your investigation. This part is essential for explaining the relevance of your research and setting it within the body of current knowledge. The following information ought to be presented in the **Findings and Discussions** section:

- Begin by presenting your results in a clear and organized manner. Use tables, graphs, or other visual aids to enhance the presentation of data.
- Provide descriptive **statistics** or relevant **metrics** to summarize key findings.
- Interpret the meaning of your findings in the context of your research questions or hypotheses. Discuss any **unexpected** or **surprising results** and consider potential explanations.
- Compare your findings with existing literature and research. Highlight **similarities**, **differences**, or **novel** contributions to the field.
- Explicitly connect your findings to the research objectives or questions stated in the introduction. Discuss whether your results support or contradict your initial hypotheses.
- Acknowledge any limitations or constraints in your study that may affect the interpretation of findings. Discuss the implications of these limitations on the validity and generalizability of your results.

The **Findings and Discussions** should be no more than **3000 words** nor fewer than **2500 words**.

## Ablation Studies

In an ablation study, a system's components are systematically removed or altered to evaluate the effect on the system's overall performance. Ablation studies are frequently employed in scientific research to examine the relative contributions of various variables or factors on the study's findings. The following information ought to be presented in a manuscript's **Ablation Studies** section:

- Clearly state the objective or purpose of conducting ablation studies.
- Provide a detailed description of the system, model, or methodology under investigation.
- Clearly define the components or variables that will be subject to ablation.
- Describe the specific methodology used for conducting the ablation studies.
- Outline the criteria for removing or modifying components. Explain the order or sequence in which ablations were performed.
- Specify the metrics or criteria used to evaluate the performance or outcomes of the system. Define how the impact of each ablation will be measured or assessed.
- Clearly outline each ablation, indicating which components were removed or modified. Interpret the findings of the ablation studies.
- Discuss the implications of each ablation on the overall system or model.

The **Ablation Studies** should be no more than **1500 words** nor fewer than **1000 words**.

## Limitations and Future Works

A manuscript's **Limitations and Future Work** section is essential for giving a clear, thoughtful explanation of the limitations of the study and for suggesting possible directions for further investigation. What ought to be in this section is as follows:

- Acknowledge any limitations related to the participants or sample size. Discuss whether the findings can be generalized to a broader population.
- Address any challenges or constraints related to the research methods employed. Discuss potential sources of bias, errors, or shortcomings.
- Reflect on the appropriateness of your study design. Discuss whether alternative designs could have provided different insights or improved the robustness of the study.
- Discuss the generalizability of your findings. Address any factors that may limit the applicability of the results to other contexts or populations.
- Propose potential extensions or follow-up studies that could build upon the current research. Discuss how these extensions could address the limitations identified.
- Explore how the findings might be applied in different contexts or industries. Discuss potential collaborations or interdisciplinary approaches.

The **Limitations and Future Works** should be no more than **1500 words** nor fewer than **1000 words**.

## References


**1.** Ahmed, J. & Ahmed, M. Online news classification using machine learning techniques. *IIUM Eng. J.* **22**, 210–225 (2021).

**2.** Mahajan, S. & Ingle, D. News classification using machine learning. *Int. J. Recent Innov. Trends Comput. Commun* **9**, 23–27 (2021).

**3.** Singh, Y. V., Naithani, P., Ansari, P. & Agnihotri, P. News classification system using machine learning approach. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 186–188 (IEEE, 2021).

**4.** Ilyas, A., Obaid, S. & Bawany, N. Z. Multilevel classification of pakistani news using machine learning. In *2021 22nd International Arab Conference on Information Technology (ACIT)*, 1–5 (IEEE, 2021).

**5.** Hasib, K. M. *et al.* Mcnn-lstm: Combining cnn and lstm to classify multi-class text in imbalanced news data. *IEEE Access* (2023).

**6.** Zhang, M. Applications of deep learning in news text classification. *Sci. Program.* **2021**, 1–9 (2021).

**7.** Zhao, W., Zhu, L., Wang, M., Zhang, X. & Zhang, J. Wtl-cnn: A news text classification method of convolutional neural network based on weighted word embedding. *Connect. Sci.* **34**, 2291–2312 (2022).

**8.** Hao, Z., AghaKouchak, A., Nakhjiri, N. & Farahmand, A. Global integrated drought monitoring and prediction system (GIDMaPS) data sets. *figshare* http://dx.doi.org/10.6084/m9.figshare.853801 (2014).


LaTeX formats citations and references automatically using the bibliography records in your .bib file, which you can edit via the project menu. Use the cite command for an inline citation, e.g.[8].

For data citations of datasets uploaded to e.g. *figshare*, please use the `howpublished` option in the bib entry to specify the platform and the link, as in the `Hao:gidmaps:2014` example in the sample bibliography file.

## Additional information

### Coding Details

**Please post your code to your public GitHub repository for the Proposed Methodology, and include a link to it in your article. Your article could not be considered if you don't comply with this requirement.**

- Organize your research code in a clear and well-documented manner.
- Create a new repository (public) for your research project on GitHub.

- Upload your code files and any associated documentation to the repository.

To include, in this order: **Accession codes** (where applicable); **Competing interests** (mandatory statement).

The corresponding author is responsible for submitting a competing interests statement on behalf of all authors of the paper. This statement must be included in the submitted article file.

**Figure 1.** Legend (350 words max). Example legend text.

| Condition | n | p |
|---|---|---|
| A | 5 | 0.1 |
| B | 10 | 0.01 |

**Table 1.** Legend (350 words max). Example legend text.

Figures and tables can be referenced in LaTeX using the ref command, e.g. Figure 1 and Table 1.