Project Report

On

News Text Classification using Various Machine Learning Models

Submitted by

Sagar Rathore, Aman Gaur, Satyam Sahu

Under the guidance of

Dr. Ankit Rajpal
Department of Computer Science
University of Delhi

April 15, 2024

Contents

Abstract

In recent years, machine learning models have demonstrated remarkable capabilities in various domains, including healthcare. This project aims to conduct a thorough comparative study of five prominent machine learning models on Multi Class Text Classification Problems. —Random Forest, Logistic Regression, K-Nearest Neighbour, Support Vector Machine —in predicting the category of news articles like technology, sports.

1 Introduction

In today's digital age, the vast amount of textual data generated daily presents both opportunities and challenges. The ability to effectively categorize and analyze this data is crucial for numerous applications, ranging from information retrieval to sentiment analysis and beyond. Text classification, a fundamental task in natural language processing (NLP), plays a pivotal role in organizing and understanding textual data.

Classification is quite a challenging field in text mining as it requires prepossessing steps to convert unstructured data to structured information. With the increase in the number of news it has got difficult for users to access news of his interest which makes it a necessity to categories news so that it could be easily accessed.

Categorization refers to grouping that allows easier navigation among articles. Internet news needs to be divided into categories. This will help users to access the news of their interest in real-time without wasting any time. Furthermore, this report explores real-world applications of text classification across various sectors, including but not limited to customer service, finance, healthcare, and social media analysis.

In this report a review of news classification based on its contents is presented. A variety of classification has been performed in past, their performance and various flaws have also been discussed.

2 Related Works

2.1 Topic Classification of Online News Articles Using Optimized Machine Learning Models.

Background

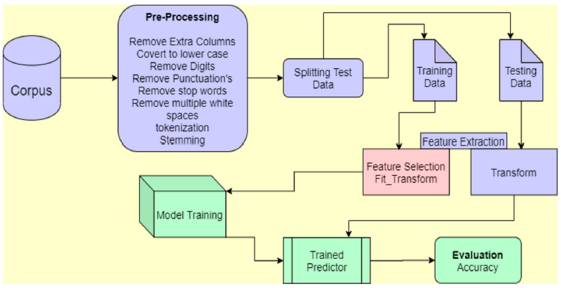
There are many online sources for news and information, and with so many sources, the need to categorize these news is crucial. For this purpose, text mining is the most common approach that has served greatly in the text-classification domain. Various Machine learning algorithms can be used to classify these articles involving various steps from information extraction, information retrieval, natural language processing, to deep learning. This paper focuses on the classifiers that are RF, SVM, SGD, KNN, LR, and NB.

Materials and Methods:

Datasets: In this research, the Reuters news dataset was used. This dataset contains 40,063 English news articles of four different categories of news articles from Reuters (www.reuters.com, accessed on 1 December 2022) covering from 2016 to 2018 years. The dataset is a real-world dataset.

The class labels of these news articles are specified in the German, namely Inlandsnachrichten, Politik, Weltnachrichten, and TopNachrichten. These categories translate to Domestic, Politics, International, and Top news in English.

Methodology: The research methodology includes several steps in a structured flow. The methodology uses a benchmark dataset as a corpus. It consists of data preprocessing, feature selection, cross-validation (dataset splitting), model training, prediction, and accuracy evaluation.



Preprocessing: First, the headline and body of each news article were merged into a single column named "news." A column named "kat" was then renamed to the more descriptive "category." Finally, unnecessary columns like "headline," "body," and "date" were removed. As some categories had more entries than others, articles were removed to create a balanced dataset with an equal number of articles per category.

The class labels' categories were encoded into integers as some models do not accept class labels in text form. The categories of the articles were encoded into numeric values.

Article preprocessing was done, including removing punctuations, lowercase conversion, stopword and extra space removals etc. Then the news articles were tokenized and stemmed.

Feature Extraction: In this step, document features were converted into vector form. As the ML algorithms work in mathematical form, these models cannot understand the text form of the feature. To convert features into vectors, the TFIDF technique was used as it is an efficient and reliable technique in categorizing news articles. In TFIDF, two different terms are used: TF and IDF. These two terms define two different types of frequencies for each feature. After combining these two frequencies, each feature is formed. TF defines the Term Frequency, which is the frequency of a word in a document, which calculates how many times a single term has occurred in a single document from all the terms present in a document.

Model Training: Both the articles after being converted into feature vectors and categories are the class labels and are fed into five different ML models: LR, NB, SGD, KNN, RF, and hyperparameter-optimized SVM; then, the models were trained using this training data. This step was performed repetitively to optimize these models by their tuning parameters (i.e., hyperparameter tuning). **Results**:

Model	Hyper-Parameters
RF	n_estimators = 3000, max_depth = 100
NB	Alfa is reduced to 0.01
SGD	max_depth = 10, average = True
KNN	Default setting K = 9
SVM	Kernel = rbf, C = 1.0
LR	Solver=saga, C = 2.8

By taking a close look, it can be easily understood that SVM outperformed all the other classifiers in the classification of real-world news articles. However, SGD, KNN, and L.R. also competed for the best-performing classifier, and their results were not much lower than SVM. RF is a great model in text classification, but in this scenario, R.F. failed in competing with other classifiers and provided the worst performance. The evaluation of models without parameter tuning was also carried out to see which algorithm performs better in classifying news articles. By

Classifier	Accuracy (w/o Optimization)	Accuracy (With Optimization)
SVM	0.6435	0.8516
SGD	0.8480	0.8476
LR	0.8437	0.8470
RF	0.7587	0.8110
NB	0.8106	0.8183
KNN	0.8104	0.8135

Figure 1: Result

comparing the results among tuned models and models that are not tuned, SGD, KNN, and LR had similar results, which means the performance of these models was not much improved even with hyperparameter tuning for the news classification. Whereas SVM and RF showed significant improvement in performance after tuning.

Conclusion : This paper investigated SVM, a machine learning technique, for classifying news articles. The researchers optimized SVM's performance through hyperparameter tuning and compared it to other popular methods. By optimizing SVM, they achieved the best classification accuracy and highlighted the importance of hyperparameter tuning (which improved accuracy by 20.81). The resulting model, trained on real-world data, can be used for practical news article classification.

2.2 Text classification of BBC news articles and text summarization using text rank

Automatic text classification and summarization are crucial tasks for managing and processing vast amounts of textual data in the digital era. These tasks are typically addressed using machine learning techniques and algorithms.

Materials & Methods: This study utilized a dataset from BBC World News consisting of documents from various categories like Business, Politics, Sports, Tech, and Entertainment. Machine learning algorithms such as Logistic Regression, Naive Bayes, Support Vector Machines (SVM), and Random Forest Classifier were employed for classification. Text summarization was performed using the TextRank algorithm, an unsupervised method that selects important sentences based on their significance and similarity. Additionally, the study explored document clustering to group similar documents based on their content. It used the Vector Space Model to represent documents as vectors and applied K-means clustering to partition the samples.

Results: In text classification, Logistic Regression achieved an accuracy of 98.29, outperforming other methods like Naive Bayes, SVM, and Random Forest Classifier. Text summarization using TextRank demonstrated effectiveness across various categories, with some variations in performance observed, especially with short stories. The clustering analysis revealed interesting patterns in document similarities, particularly between politics and business articles.

Conclusion: The study highlights the effectiveness of machine learning techniques in text classification and the utility of the TextRank algorithm for text summarization. Despite some challenges, such as summarizing short stories, the overall results demonstrate the potential of these methods in managing and processing textual data. Future work may focus on addressing polysemy in text classification, improving summarization techniques, and automating evaluation processes for better consistency.

2.3 Multi-Category News Classification using Support Vector Machine-based Classifiers Pooja Saigal1 · Vaibhav Khanna2

The paper delves into text classification, particularly addressing the challenge posed by the abundance of text-based data on the internet. It introduces machine learning algorithms, specifically Support Vector Machines (SVM) and its variants, as effective tools for automating text classification tasks. The focus lies on comparing three established SVM-based classifiers – Least-squares Support Vector Machines (LS-SVM), Twin Support Vector Machines (TWSVM), and Least-squares Twin Support Vector Machines (LS-TWSVM) – specifically in the context of multi-category text classification using news data.

Materials & Methods: The study begins with the preprocessing of the dataset, involving tokenization, stemming, elimination of stop words, and setting a minimum document frequency to remove less significant terms. Feature set generation is then accomplished using Term Frequency-Inverse Document Frequency (TF-IDF) matrix construction, which assigns weights to each term based on its relevance. The paper utilizes the One-Against-All approach to extend binary classifiers to handle multi-category data. Experimental comparisons are conducted on benchmark UCI News datasets – Reuters and 20 Newsgroups – through simulations.

Results: The comparative analysis reveals LS-TWSVM as the most effective classifier among the three, demonstrating superior performance in terms of both accuracy and time complexity (for both training and testing). LS-TWSVM outperforms LS-SVM and TWSVM, showcasing better generalization ability and computational efficiency for the multi-category news classification problem.

Conclusion: In conclusion, the study highlights the efficacy of LS-TWSVM for text categorization, particularly in handling multi-category datasets like news data. The paper suggests a significant enhancement in performance and computational speed with the least squares version of classifiers due to their ability to solve linear equations rather than quadratic programming problems. The findings open avenues for further exploration, encouraging the development and experimentation of new SVM-based classifiers for similar applications, which could further bolster

the effectiveness of SVM approaches in text categorization tasks.

3 Problem Statement

Develop a **Classification Model** that accurately classify a news article based across different categories.

4 Methodology

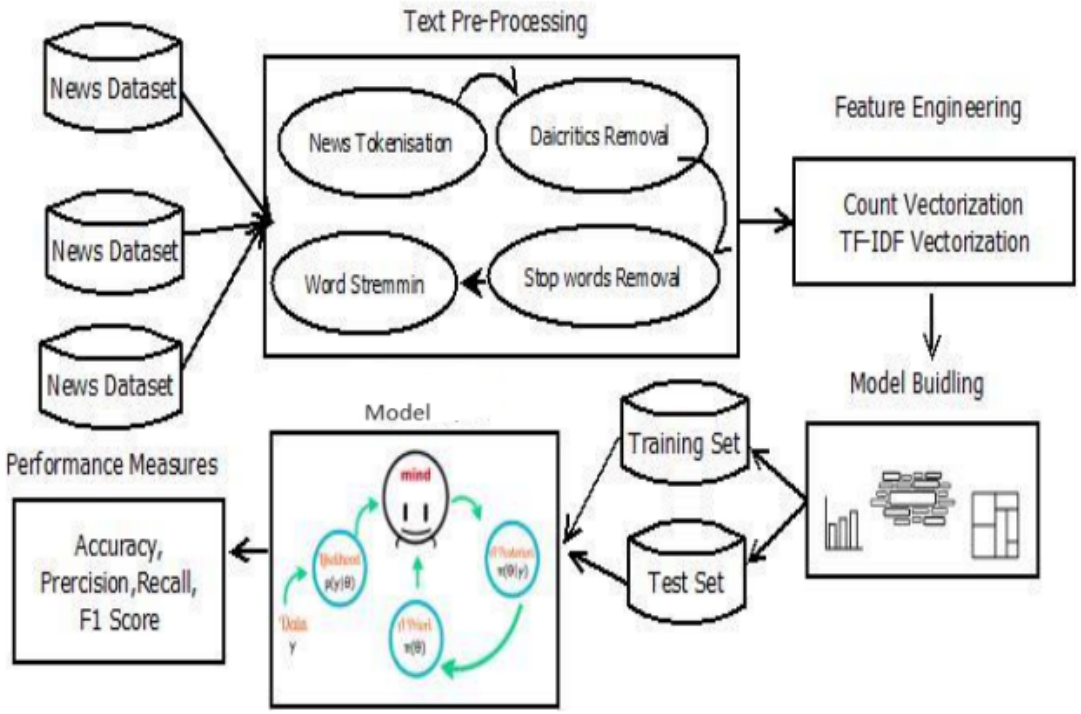


Figure 2: Flow Diagram

The methodology for depends on following phases for prediction of survival of horse.

1. **Text Preprocessing**: The process of Text Categorization involves categorizing the text-based documents into predefned classes. But this task involves a level of complexity in the sense that the dataset contains a large number of documents which are required to be classifed into predefned categories. For the purpose of completion of categorization process, two advanced techniques are implemented on the dataset, called extraction and preprocessing.

2. **Feature Engineering**: The process of selecting and transforming features. This includes features scaling normalization and encoding categorical attributes.

3. **Model Selection and Training**: Evaluate the performance of various machine learning models.

4. **Performance Measures** :Evaluate the final model using metrics such as accuracy, precision, recall, and **F1-score** on an unseen test dataset. Analyze the errors made by the model to identify potential biases or limitations.

4.1 Dataset

For this research, we used the BBC world news dataset. The data consists of a total of 2225 text documents from the BBC world news website corresponding to the stories covered in 2004-2005. The total size of the data is around 5 MB.

The data is pre-classified into 5 different class labels. These class labels are Business, Sport, Tech, Entertainment, and Politics. It contains 510 text documents from the business category, 511 text documents from the sports category, 401 text documents from the tech category, 386 text documents from the entertainment category, and 417 articles from the politics category.

There are a total number of 456,542 words in all the documents while containing only 29,126 unique words. This means that on average every word appears average about 15 times throughout the whole corpus. The average document length is 205 words. Note however that all these statistics are reported after the removal of all stop words. Each document in reality is about 1500 words long so clearly stopwords comprise a great amount of the document.

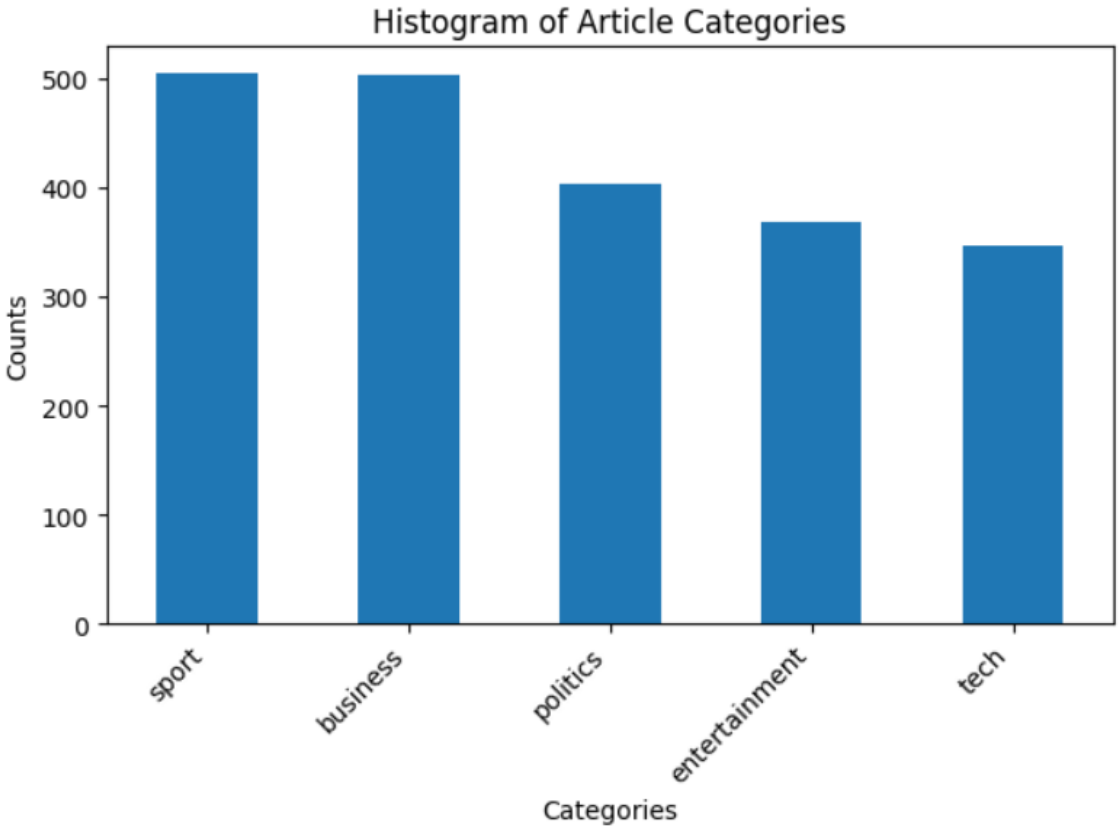


Figure 3: BBC News Histogram

4.1.1 Text Preprocessing

Text preprocessing involves eliminating noise such as punctuation, numbers, and irrelevant words (stop words) from the dataset. The goal is to make the text more uniform and structured, thereby helping NLP models focus on meaningful information and improving their efficiency and effectiveness. Python libraries like NumPy, pandas, and scikit-learn are commonly used to streamline these tasks, ensuring efficient data handling and analysis.

1. **Conversion to Lowercase:** All text data is converted to lowercase to ensure uniformity and prevent redundancy from multiple occurrences of words with different casing.

2. **Removal of Punctuation:** Punctuation such as full stops, commas, question marks, and exclamation marks is eliminated from the corpus to simplify the text.

3. **Elimination of Numbers:** Numeric characters are removed from the text as they do not contribute much to the analysis.

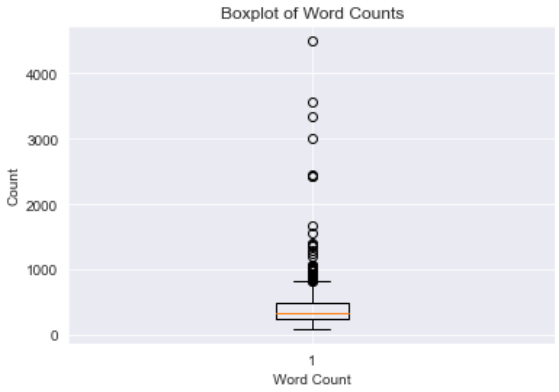
4. **Exclusion of Stop Words:** Common words like "the", "a", "and", etc., are removed from the corpus as they do not add much meaning or information to the text.

5. **Omitting Short Words:** Words containing two or fewer characters are filtered out from the text corpus as they often do not provide useful information.

6. **Removal of Extra Spaces:** Excess whitespace, including multiple consecutive spaces, is condensed into a single space to ensure text uniformity.

4.2 Feature Engineering

Feature engineering is a crucial step in the machine learning pipeline where you create new features or modify existing ones to improve the performance of your model. Effective feature engineering can lead to better model interpretability, generalization, and overall predictive performance



We can see there are most of text column contains upto 500 words and some are consisting too many words kinda outliers as we have seen most of words containing text or lets say articles most of articles containing words under 1000 so we remove the outliers from our dataset.

4.2.1 Feature Extraction

It is very important to represent the text in which machine can easily understand. Therefore, Vectorizer has been used which converts the sentence or text into an array or vector of numbers.

To transform the term into a representation of number which has meanining we use TF-IDF.

Term Frequency is used for normalizing the occurrence of each word with size of the data set.

Inverse Document Frequency is used for removing the words which do not contribute much for deciding the meaning of the sentence.

TF-IDF algorithm's main focus is on the word that has the higher frequency in the text, and at the same time, it appears in the corpus in a smaller range. Then, this word has the strongest capability to distinguish different classes in the text.

Formula: $\text{tfidf}(t, d, D) = \text{tf}(t, d) \times \text{idf}(t, D)$

$$\text{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

$$\text{IDF}(t, D) = \log_e \left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t} \right)$$

4.3 Model Selection And Training

In the process of model selection and training, the first crucial step is to clearly define the problem at hand, whether it be classification, regression, clustering, or another type of task. Following this, a comprehensive understanding of the dataset is essential, involving analysis of feature types, distributions, and relationships, as well as addressing any missing values or outliers. The dataset is then split into training, validation, and test sets. Choosing an appropriate algorithm comes next, ranging from linear models and tree-based methods to support vector machines and neural networks. Hyperparameters are fine-tuned through techniques like grid search or random search. The model is trained on the training set and evaluated on the validation set, iterating through hyperparameter adjustments if necessary. Once satisfied, the final model is trained on both the training and validation sets and evaluated on the test set to ensure robust generalization to unseen data.

Our Problem is a Text Classification problem with 5 classes and we have converted the text into TF-IDF feature vectors in feature extractions. We have used following models for the project:

1. Random Forest Classifier
2. Logistic Regression Classifier
3. KNN Classifier

4. SVM Classifier

Before we get into evaluation let's understand each model first.

4.3.1 Ensemble Learning

Ensemble learning is a supervised learning technique that combines multiple models to build a more powerful and robust model and then to combine their predictions for the classification of unseen instances using some form of voting (such as majority voting. . .).

There are mainly two types of ensemble classifiers.

- Bagging (Bootstrap Aggregation) - Bagging is a type of ensemble learning in which multiple base(weak) models are trained independently in parallel on different subsets of the training data. subset is generated using bootstrap sampling (Randomly 'n' subsets of original training data are sampled with replacement.). In bagging classifier final prediction using majority voting. In bagging regression final prediction is made by averaging.

- Boosting - Boosting is a machine learning technique that combines multiple weak learners into a single strong learner.Boosting works by sequentially improving the performance of individual weak learners by assigning more weight to the data points that are misclassified or have high error.

Random Forest Classifier A random forest model is a extension of bagging technique that combines multiple decision trees to create a more accurate and robust prediction. It works by selecting random subsets of the data and features, also known as feature bagging, generates a random subset of features, which ensures low correlation among decision trees, and building a decision tree for each subset. Then, it aggregates the predictions of all the trees to produce the final output.

Used Parameters:

1. **n_estimators**: The number of trees in the forest.

2. **max_depth**: The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

4.3.2 Logistic Regression

Logistic regression is a statistical method used for binary classification tasks. However, it can be extended to handle multiclass classification by employing techniques such as one-vs-all or multinomial logistic regression.

In logistic regression, the probability that an instance belongs to a particular class is modeled using the logistic function (also known as the sigmoid function). For multiclass logistic regression, the probability of an instance belonging to each class is calculated, and the class with the highest probability is predicted.

The logistic function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Where z is a linear combination of the input features and model parameters:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Here, $\sigma(z)$ represents the probability of the positive class (class 1), e is the base of the natural logarithm, and $\theta_0, \theta_1, \ldots, \theta_n$ are the model parameters (coefficients) learned during training.

In multiclass logistic regression, the model computes probabilities for each class c using the softmax function:

$$P(y = c|x; \theta) = \frac{e^{\theta_c^T x}}{\sum_{k=1}^{K} e^{\theta_k^T x}}$$

Where $P(y = c|x; \theta)$ is the probability that the instance x belongs to class c given the parameters θ, K is the total number of classes, and θ_c is the parameter vector for class c.

During training, the parameters θ are learned by minimizing a suitable loss function, such as the cross-entropy loss, using optimization algorithms like gradient descent.

Let x be the input feature vector and β be the parameter vector. The logistic regression model can be represented as:

$$P(y = 1|x) = \frac{1}{1 + e^{-x^T \beta}}$$

where:

- $P(y = 1|x)$ is the probability that the output y is 1 given the input x,
- e is the base of the natural logarithm,
- x^T denotes the transpose of x,
- β is the parameter vector, and
- $\frac{1}{1+e^{-x^T \beta}}$ is the logistic (sigmoid) function.

The logistic function maps any real-valued number to the range $[0, 1]$, making it suitable for modeling probabilities. During training, the parameters β are estimated using methods such as maximum likelihood estimation or gradient descent to minimize the error between the predicted probabilities and the actual labels.

Used Paremeters:

1. **C** : Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.
2. **penalty**: Specify the norm of the penalty:
   (a) None: no penalty is added;
   (b) 'l2': add a L2 penalty term and it is the default choice;
   (c) 'l1': add a L1 penalty term;
3. **max_iter**: Maximum number of iterations taken for the solvers to converge.

4.3.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. In the context of classification, SVM aims to find the optimal hyperplane that best separates the data points into different classes, with the largest margin between the classes.

For binary classification, SVM finds the hyperplane by maximizing the margin between the nearest data points from each class, which are called support vectors.

For multiclass classification, SVM can be extended using techniques such as one-vs-one or one-vs-all. In one-vs-one, SVM constructs a binary classifier for each pair of classes and then combines their outputs to make the final multiclass prediction. In one-vs-all, SVM trains a binary classifier for each class against the rest of the classes.

Mathematically, the decision function of SVM for binary classification can be represented as:

$$f(x) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Where: - $\mathbf{w}$ is the weight vector. - $\mathbf{x}$ is the input feature vector. - b is the bias term. - sign$(\cdot)$ is the sign function.

For multiclass classification using one-vs-one or one-vs-all, the decision function is typically combined with voting or other aggregation methods to determine the final predicted class.

During training, SVM optimizes the margin and finds the hyperplane parameters $\mathbf{w}$ and b by solving the optimization problem, often formulated as the minimization of the hinge loss function regularized by the L2 norm of the weight vector.

Used Parameters:

1. **C** :Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.

2. **Kernel** : Specifies the kernel type to be used in the algorithm. If none is given, 'rbf' will be used. If a callable is given it is used to pre-compute the kernel matrix from data matrices; that matrix should be an array of shape (n_samples, n_samples).

3. **gamma** Kernel coefficient for 'rbf', 'poly' and 'sigmoid'.

4. **degree** Degree of the polynomial kernel function ('poly'). Must be non-negative. Ignored by all other kernels.

4.3.4 k-Nearest Neighbors (kNN)

k-Nearest Neighbors (kNN) is a simple and intuitive machine learning algorithm used for classification and regression tasks. In kNN classification, the class of a new data point is determined based on the class labels of its nearest neighbors in the feature space.

For multiclass classification with kNN, the class label of a new data point is often determined by a majority vote among its k nearest neighbors. Each neighbor

contributes equally to the decision, and the class with the highest count among the neighbors is assigned to the new data point.

Mathematically, the class assignment for a new data point x in a kNN classifier can be represented as:

$$\hat{y} = \arg\max_{c} \sum_{i=1}^{k} I(y_i = c)$$

Where: - $\hat{y}$ is the predicted class label for x. - y_i is the class label of the ith nearest neighbor of x. - c iterates over all possible class labels. - $I(\cdot)$ is the indicator function that returns 1 if its argument is true and 0 otherwise.

During prediction, the algorithm calculates the distance between the new data point and all training data points using a distance metric such as Euclidean distance. It then selects the k nearest neighbors based on these distances and assigns the class label based on the majority vote.

1. **n_neighbours**: Number of neighbors to use by default for kneighbors queries.
2. **weights**: Weight function used in prediction. Possible values: 'uniform', 'distance'.
3. **algorithm**: Algorithm used to compute the nearest neighbours.

Splitting Data

- Train-Test Split: Description: The dataset is divided into two subsets: a training set used to train the model and a test set used to evaluate its performance. Use Case: Commonly used when there is a sufficient amount of data. A typical split might be 80% for training and 20% for testing.

We initially employed train-test split of 80-20 and trained all the models on **default parameters** and then employed **GridSearchCV** to find best hyperparameters and evaluated the models on these hyperparameters.

4.4 Results

4.4.1 Random Forest

The following were results of Random Forest Classifier on **default parameters**.

Table 1: Performance Metrics

Metric	Value
Accuracy	0.9501
Precision	0.9458
Recall	0.9490
F1 Score	0.9471

The following were results after finding hyperparameters using GridSearchCV

Table 2: Performance Metrics

Metric	Value
Accuracy	0.9596
Precision (Macro)	0.9554
Recall (Macro)	0.9572
F1-Score (Macro)	0.9560

The Random Forest Classifier showed an improvement in all performance metrics after hyperparameter tuning using GridSearchCV. The accuracy increased from 0.9501 to 0.9596, indicating that the tuned model made more correct predictions. Additionally, there were improvements in precision, recall, and F1-score, demonstrating an overall enhancement in the model's ability to classify instances across different classes.

4.4.2 Logistic Regression

The following were results of Logistic Regression on **default parameters**.

Table 3: Performance Metrics

Metric	**Value**
Accuracy	0.988123
Precision (Macro)	0.98938
Recall (Macro)	0.986024
F1-Score (Macro)	0.987469

The following were results after finding **hyperparameters** using **GridSearchCV**

Table 4: Performance Metrics

Metric	Value
Accuracy	0.9881
Precision (Macro)	0.9894
Recall (Macro)	0.9860
F1-Score (Macro)	0.9875

The Logistic Regression Classifier showed consistent performance between default parameters and after hyperparameter tuning using GridSearchCV. There were only minor changes in the evaluation metrics, indicating that the default parameters already resulted in a highly effective model. Both accuracy and F1-score remained high, demonstrating the model's robustness in correctly classifying instances across different classes.

4.4.3 KNN Classifier

The following were results of KNN Classifier on **default parameters**.

Table 5: Performance Matrix

Metric	**Value**
Accuracy	0.9406
Precision (Macro)	0.9394
Recall (Macro)	0.9411
F1-Score (Macro)	0.9400

The following were results after finding **hyperparameters** using **GridSearchCV**

Metric	**Value**
Accuracy	0.9406
Precision	0.9405
Recall	0.9416
F1 Score	0.9406

The KNN Classifier showed consistent performance between default parameters and after hyperparameter tuning using GridSearchCV. There were only slight changes in the evaluation metrics, indicating that the default parameters already resulted in a reasonably effective model. The accuracy and F1-score remained stable, suggesting that the model's ability to classify instances across different classes did not significantly improve with hyperparameter tuning.

4.4.4 SVM

The following were results of SVM on **default parameters**.

Table 6: Performance Matrix

Metric	**Value**
Accuracy	0.9857
Precision (Macro)	0.9876
Recall (Macro)	0.9836
F1-Score (Macro)	0.9853

The following were results after finding **hyperparameters** using **GridSearchCV**

Metric	**Value**
Accuracy	0.9881
Precision	0.9887
Recall	0.9875
F1 Score	0.9880

The SVM classifier showed improvement in accuracy, precision, recall, and F1-score after hyperparameter tuning using GridSearchCV. The accuracy increased from 0.9857 to 0.9881, indicating that the tuned model made more correct predictions. Additionally, there were improvements in precision, recall, and F1-score, demonstrating an overall enhancement in the model's ability to classify instances across different classes.

5 Conclusion

In conclusion, all models displayed competence in text classification tasks, with each having its advantages and areas for improvement. Logistic Regression exhibited consistent high performance, while Random Forest and SVM showed significant improvements after hyperparameter tuning. KNN Classifier, although stable, didn't exhibit notable enhancements with tuning.However, the final selection should consider various factors like computational resources, interpretability, and specific requirements of the application.

6 References

1. Multi-category news classifcation using Support Vector Machine based classifers Pooja Saigal1 · Vaibhav Khanna2

2. A Comparative Analysis of Logistic Regression, Random Forest and KNN Models for the Text Classification

3. Topic Classification of Online News Articles Using Optimized Machine Learning Models. Computers 2023, 12, 16. https://doi.org/10.3390/computers12010016

4. Text classification of BBC news articles and text summarization using text rank

5. News Classification and Its Techniques – A Review