International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

TSAO: Twitter Sentiment Analysis Using Deep Learning with Optimization Techniques

[1]C. Suresh Kumar, [2]Dr. J. K. Kanimozhi

Submitted: 03/10/2023 **Revised**: 20/11/2023 **Accepted**: 03/12/2023

Abstract: Sentiment analysis is essential for understanding people's attitudes and views about different entities. In this research, proposed Twitter sentiment analysis with optimization (TSAO). Initially the dataset has collected from benchmark datasets. In the data set gathering module, a thorough collection of labeled sentiment data is gathered in order to train and assess the model. The pre-processing module then employs enhanced Part of Speech (POS) labeling with Natural language processing (NLP) and WORDNET to increase the semantic understanding of the text input. To extract relevant features from text for the deep learning training module, a CNN and Dense architecture with Resnet50 are employed. As a result, the model is able to learn sophisticated sentiment patterns and representations. To improve the feature selection method, ensemble feature selection approaches are applied. To find the most relevant and informative features, Elastic Net, Recursive Feature, and Hybrid ML Classifier, which includes the Decision Tree and Random Forest algorithms, are utilized. Optimization approaches are used to fine-tune the model's parameters and improve its effectiveness using the Alternative Least Square (ALS) algorithm. The Ensemble Stacking and Ensemble Voting algorithms are used in the classification phase to combine predictions from several models and enhance overall classification accuracy.

Keyword: *Alternative Least Square, Convolutional Neural Network, Language Processing, Sentiment Analysis, WORDNET*

1. Introduction

Sentiment analysis, often known as opinion mining, is a popular NLP task that involves extracting subjective information from text [1]. The goal here is to determine if a particular piece of text has an upbeat, downbeat, or neutral emotional tone [2]. As a result of its adaptability, sentiment analysis has been more popular in recent years [3]. Some of the ways it has been put to use include the interpretation of customer feedback, the monitoring of social media, the management of company reputation, and market research. Traditional methods to sentiment analysis often depend on rule-based systems or feature engineering techniques, which need explicitly written rules or preexisting lexical resources [4]. While these techniques yielded respectable results, they typically failed to capture complicated verbal patterns and context-dependent emotional expressions [5]. Deep learning has considerably moved sentiment analysis toward more complex and data-driven methodologies [6].

Deep learning models, notably neural networks, have exhibited extraordinary effectiveness in a variety of NLP applications, including sentiment analysis [7]. These models excel in learning hierarchical representations automatically from unstructured text data, allowing them to grasp deep connections between words, sentences, and context. By using large-scale labeled datasets, deep learning-based sentiment analysis algorithms may efficiently understand the underlying sentiment patterns and generalize well to unseen data [8-10]. However, owing to a variety of reasons, training deep learning models for sentiment analysis may be challenging [11]. First, a scarcity of labeled data for sentiment analysis in certain topics or languages is a severe challenge [12]. Large datasets may be time-consuming and costly to collect and annotate. Second, deep learning models are susceptible to over fitting, which happens when they remember the training data instead of acquiring substantial generalizations [13]. When the dataset is restricted or uneven, this issue is worsened. To overcome these challenges and increase the performance of sentiment analysis algorithms, many optimization approaches [14] have been developed. These strategies are designed to improve the learning process, model generalization, and overall performance. Popular optimization approaches include transfer learning, data augmentation, regularization methods, and hyperparameter optimization [15].

Transfer learning allows sentiment analysis algorithms to apply previously acquired language information to large-scale general language tasks like language modeling [16]. By starting with pretrained weights, the sentiment analysis model may take use of the learnt representations and possibly achieve better performance even with less labeled sentiment data [17]. Data augmentation strategies alleviate the scarcity of labeled sentiment data by

[1]*Assistant Professor, Department of Computer science, Cheran Arts Science College, Kangeyam.*
san.sureshmca@gmail.com
[2]*Assistant Professor, P.G. Department of Computer Science, Senguthar Arts & Science College, Tiruchengode.*
drjkkanimozhi@gmail.com

producing synthetic data samples. These approaches require transforming current data samples using changes like as synonym substitution, paraphrasing, and back-translation [18]. Models may gain more robust representations and improve their ability to cope with fluctuations in sentiment expression by enriching the dataset. Dropout and weight decay are two regularization approaches used to reduce overfitting and increase model generalization [19]. During the training phase, these strategies impose limitations on the model, encouraging it to acquire more meaningful and generalizable representations [20]. The goal of hyperparameter optimization is to find the best configuration of sentiment analysis model hyperparameter. Hyperparameter govern several features of the model, such as learning rate, sample size, and network design [21]. Models may uncover optimum settings by methodically studying the hyperparameter space using approaches such as grid search or Bayesian optimization [22].

This manuscript's main contributions and aims may be summed up as follows.

- Pre-Processing using Enhanced POS Tagging with NLP and WORDNET
- Training using CNN and Dense architecture with Resnet50
- Ensemble Feature Selection using Elastic Net, Recursive Feature and Hybrid ML Classifier (Decision tree with Random forest algorithm)
- Optimization using ALS (Alternative Least Square Algorithm)
- Classification using Ensemble Stacking and Ensemble Voting algorithm

For the remainder of this paper, the structure will look like this. In Section 2, several writers discuss different approaches to Sentiment Analysis. In Section 3, we can see the suggested model. The inquiry findings are presented in Section 4. The conclusion and recommendations for further research are presented in Section 5.

2. Background Study

Abburi et al. [1] developed a method for extracting emotion from audio and text data. They developed DNN and GMM-based sentiment models based on audio MFCC characteristics. The DNN classifier using 65-dimensional MFCC features outperforms the GMM classifier in terms of accuracy. Sentiment analysis was much enhanced when audio and text were used together. Day and Lin [5] Utilizing data from Google Play reviews, trained a deep neural network to improve the precision of sentiment analysis. They took samples at random and utilized them to create a dictionary of feelings. The efficiency of the deep learning system was enhanced by data that was not representative of the whole. Demirci et al. [7] employing a mix of three supervised machine learning models and one deep learning model, they zeroed in on Turkish sentiment. Using a variety of text properties, they discovered that deep learning algorithms may be put to use for binary text categorization. Heikal et al. [10] shown the efficacy of a deep learning model in assessing the emotional content of Arabic Twitter postings. They created a model that relied on a pre-trained word vector representation to outperform previous models in terms of F1-score and accuracy. Jangid et al. [12] showed how to use deep learning models to analyze financial tweets and headlines from several perspectives. They suggested a new method for calculating the strength of a sentiment using deep learning. They want to look at how ensemble models function, and how well they do it, using data from various fields. Mittal et al. [13] addressed image sentiment analysis using deep learning techniques. They discussed past studies and expected more efficient techniques to emerge in the future. Pasupa and Seneewong Na Ayutthaya [15] experimented on sentiment classification of Thai children tales using LSTM, Bi-LSTM, and CNN deep learning models. By examining all bits of information concurrently and eliminating the memory issue, the CNN model fared the best. Roshanfekr et al. [17] studied deep neural network architectures for sentiment classification of documents. Word vector representations may have contributed to the superior performance of deep learning approaches compared to NBSVM. Additionally, a new Persian sentiment categorization dataset was presented. Shilpa et al. [19] developed a model for analyzing the tone of tweets using deep learning tools like RNN and LSTM. They incorporated feature selection methods like TF-IDF and Doc2Vec, achieving better classification results. They suggested investigating the analysis of users' personality from their tweets in future work.

Table 1: comparison table for existing work

Author	Year	Methodology	Advantage	Limitation
Abburi, H. et al.	2017	Gaussian Mixture Models	the approach extracts textual features from audio transcripts using Doc2vec	This limits the applicability of the approach to situations where audio data is present.

			vectors and employs a Support Vector Machine (SVM) classifier to develop a sentiment model based on these features.	Additionally, the extraction of MFCC features from audio inputs may introduce challenges related to noise, varying audio quality, and the need for robust audio processing techniques.
Al-Dabet, S. et al.	2021	Deep learning	Two primary goals are well addressed by the proposed deep learning models for aspect-based sentiment analysis in Arabic reviews. Aspect categorization and aspect-emotion labeling.	The models require a sufficient amount of labeled data for effective training, which may be limited or unavailable for certain domains or specific aspects within the reviews.
Day, M.-Y., & Lin, Y.-D.	2017	Deep Learning	Researchers employed deep learning to improve the accuracy of sentiment analysis based on evaluations from Chinese consumers in this study.	The possible bias generated by the data collecting process is one drawback of the research on sentiment analysis of Chinese customer reviews using deep learning.
Demirci, G. M.	2019	Machine Learning	The ability to extract meaningful information from unstructured textual data when using machine learning and deep learning techniques for sentiment analysis on social media data during times of tragedy is a benefit of using machine learning and deep learning methods for sentiment analysis on social media data.	The study has limitations due to the potential bias in the dataset used to examine the sentiment of social media postings made during times of disaster using machine learning and deep learning.

3. Materials and Methods

We begin by gathering data sets that are representative and varied. Preprocessing approaches such as increased POS labeling with NLP and WordNet inclusion may improve the quality and understanding of text data. We use CNN and Dense architectures with ResNet50 as the backbone for deep learning training, allowing for fast feature extraction. Elastic Net, Recursive Feature Elimination, and a Hybrid ML Classifier that combines Decision Tree and Random Forest are just few of the ensemble feature selection algorithms used to identify the most informative features. Furthermore, ALS (Alternative Least Square Algorithm) optimization is carried out. Finally, to achieve exact predictions, categorization is performed using ensemble stacking and ensemble voting algorithms.

3.1 Dataset collection

The dataset is titled "Sentiment Analysis Dataset" and is available on Kaggle. **https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset**.

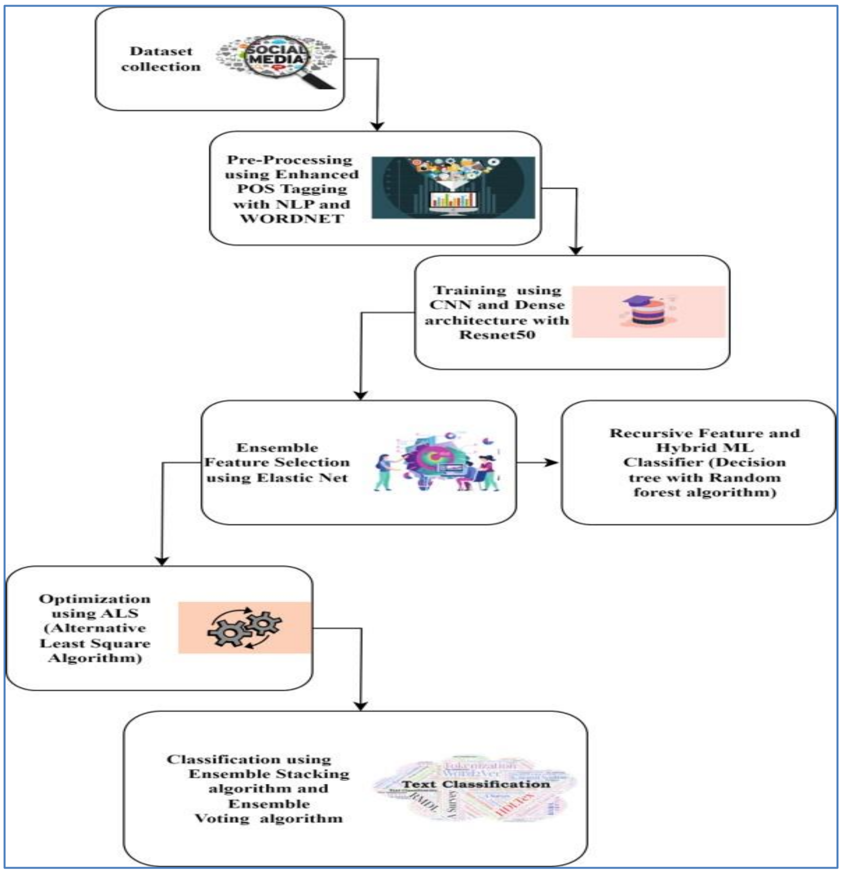

Dataset collection
SOCIAL MEDIA
Pre-Processing using Enhanced POS Tagging with NLP and WORDNET
Training using CNN and Dense architecture with Resnet50
Ensemble Feature Selection using Elastic Net
Recursive Feature and Hybrid ML Classifier (Decision tree with Random forest algorithm)
Optimization using ALS (Alternative Least Square Algorithm)
Classification using Ensemble Stacking algorithm and Ensemble Voting algorithm
Text Classification

Fig 1: overall architecture

3.2 Preprocessing using enhanced POS tagging with NLP and wordnet

3.2.1: POS tagging

Assigning grammatical labels or tags to words in a phrase to indicate their syntactic functions and categories, known as part-of-speech (POS) tagging, is a key job in natural language processing (NLP). Labels like "noun," "verb," "adjective," "adverb," "pronoun," "preposition," "conjunction," and "other" are all examples of POS tags. Syntactic parsing, information extraction, sentiment analysis, and machine translation are just some of the downstream NLP applications that rely on accurate POS tagging. The inherent uncertainty of language makes POS labeling a difficult undertaking. In many cases, the context in which a word is used will determine which POS label is most appropriate. The word "run" may function as a verb in the phrase "I run every day" or as a noun in "I went for a run." Language expertise, statistical models, and machine learning algorithms are used by POS taggers to determine the most probable POS label for each word in a phrase. Rule-based systems or manually produced lexical resources with predetermined mappings between words and POS tags were formerly used in POS tagging. These approaches produced passable output, but they had difficulty dealing with the complexities and variations of natural language.

While a syntax parsing model may be able to anticipate the POS tag directly, we nevertheless choose to use a specialized POS tagger for two reasons. Massive amounts of data annotated just with POS tags and no syntax may be found in the People's Daily corpus and the SIGHAN bakeoff corpus. Using this data to train a parser model is strictly forbidden; nevertheless, they may be used to train POS taggers. A bigger training set is typically used to improve POS tagging accuracy. Second, modern implementations of POS tagging often use sequence labelling models during training, which is not the case for older systems.

$w_{i-2}t_i, w_{i-1}t_i, w_it_i, w_{i+1}t_i, w_{i+2}t_i$ ----- (1)

$w_{i-2}w_{i-1}t_i,\ \ w_{i-1}w_it_i,\ \ w_iw_{i+1}t_i, w_{i+1}w_{i+2}t_iw_{i-1}w_{i+1}t_i$ ------ (2)

Each word in a phrase must be assigned a value, or tag, from a list of potential labels, or tags, t 2 T, in order to perform POS tagging. We also use a linear chain of CRFs for POS labeling. Algorithm 1 shows striking similarities between the feature templates used in successful CTB corpus experiments and those used in (Qian et al., 2010). There are three distinct kinds of features: Word-level data include things like the average word length and the frequency of nearby unigrams and bigrams; character-level features include things like word borders and starting and ending letters. iii) characteristics of the changing situation.

Algorithm 1:POS tagging

```
Input: Sentence (a sequence of words)
Algorithm: CRF-based POS Tagging
1. Preprocessing:
   • Tokenize the input sentence into individual words.
   • Optional: Perform any necessary normalization or cleaning steps on the words (e.g., lowercase, remove
     punctuation).
2. Feature Extraction:
   • Define a set of feature templates based on word-level, character-level, and transition characteristics.
   • For each word in the sentence, extract the relevant features based on the defined templates.
   • Convert the features into a numerical representation (e.g., feature indices).
3. Inference:
      o Initialize the sequence of predicted POS tags for each word in the sentence.
      o For each word in the sentence:
           ▪ Compute the scores of each possible POS tag for the current word based on the extracted
             features.
           ▪ Take into account the scores of neighboring words and their tags using the defined transition
             features.
           ▪ Apply the Viterbi algorithm or other inference techniques to find the most likely sequence of
             POS tags for the current word.
           ▪ Update the predicted POS tag sequence accordingly.
4. Output:
   • Return the POS-tagged sentence with each word assigned its corresponding POS tag.
```

3.3 Enhanced POS Tagging

Part-of-speech (POS) tagging is the act of labeling or tagging words in a sentence according to their grammatical function, and enhanced POS tagging refers to the use of cutting-edge methods and techniques to increase the precision, reliability, and contextual understanding of this process. Traditional POS tagging techniques have attained respectable accuracy; nevertheless, the increasing complexity and variety of language has prompted the need for more advanced methodologies. To overcome obstacles like ambiguity, context, and domain-specific language changes, improved POS tagging methods combine the strengths of cutting-edge machine learning algorithms, statistical models, and linguistic expertise.

Pipelining allows for sequential processing of separate operations like POS labeling and dependency parsing.

At first, we choose the optimal placement of point-of-sale tags, indicated by t.

$t = argmaxScore_{pos}(x,t)$ ------ (3)

Then, we choose the optimal dependency network d from x and t.

$d = argmaxScore_{syn}(x,t,d)$ ------ (4)

Tags at the Point of Sale Determined by Customer Feedback Forms Our POS tagger is based on the first-order conditional random field (CRF). The probability of a tag sequence is defined in CRF, a conditional log-linear probabilistic model, as

$p(t|x) = \frac{exp\left(Score_{pos}(x,t)\right)}{\sum_t exp\left(Score_{pos}(x,t)\right)}$ ------ (5)

$Score_{pos}(x,t) = w_{pos}.f_{pos}(x,t) = \sum_{1\le i\le n} W_{pu}.f_{pu}(x,t) + W_{pb}.f_{pb}(x,t_{i-1},t_i)$ ------- (6)

Where$f_{pos}/f_{pu}/f_{pb}(.)$ represents the feature vectors and $w_{pos}/W_{pu}/W_{pb}(.)$ represents the weight vectors. Features for POS unigrams are represented by$f_{pu}(x,t)$, whereas features for POS bigrams are represented by $f_{pb}(x,t_{i-1},t_i)$. These imputed Chinese traits are appreciated nevertheless. They employ the Chinese characters for a word to build elaborate characteristics, which is very useful for foreign or unusual words.

Second-order graph-based dependency parsing Using the graph-based interpretation of dependency parsing is similar to looking for the tree in a directed graph that has the highest score. We choose the second-order model of since it has been proved to provide the highest parsing accuracy across a wide range of languages.

$$Score_{syn}(x,t,d) = w_{syn}.f_{syn}(x,t,d) = \sum_{\{(h,m,l)\}d} W_{dep}.f_{dep}(x,t,h,m,l) \text{ ------ (7)}$$

Algorithm 2:Enhanced POS Tagging

Input: Sentence (a sequence of words)

1. POS Tagging: a. Preprocessing:

- Tokenize the input sentence into individual words.
- Optional: Perform any necessary normalization or cleaning steps on the words (e.g., lowercase, remove punctuation).

b. POS Tagging:

- Assign a POS tag to each word in the sentence based on its grammatical role and category.
- Utilize advanced machine learning algorithms, statistical models, or linguistic knowledge to predict the most likely POS tag for each word.
- Apply any relevant techniques such as CRFs, sequence labeling models, or rule-based systems to improve accuracy and handle language ambiguity.

2. Dependency Parsing: a. Dependency Parsing:

- Analyze the grammatical relationships between words in the sentence.
- Assign each word a dependency label and identify the head word it depends on.
- Utilize graph-based or transition-based parsing algorithms to construct the dependency parse tree.
- Consider advanced models and techniques that capture contextual information, word dependencies, and syntactic structures to improve parsing accuracy.

3. Output:

- Return the POS-tagged sentence with each word assigned its corresponding POS tag.
- Return the dependency parse tree representing the grammatical relationships between the words.

3.4 NATURAL LANGUAGE PROCESSING

Despite its popularity for NLP problems, it has found use in many other fields as well. When it comes to data classification, the Naive Bayes Classifier takes a purely probabilistic approach. To arrive at an appropriate prediction of the text label, Bayes' theorem is employed, together with the assumption that each pair of characteristics is conditionally independent. By using the Bayes theorem (Equation 8), we can calculate the probability that a given document will be assigned to a certain outcome categoryT. Chen et al. (2022).

$$\boldsymbol{P}\left(\frac{\boldsymbol{co}}{\boldsymbol{cd}}\right) = \frac{\boldsymbol{P}\left(\frac{\boldsymbol{cd}}{\boldsymbol{co}}\right)*\boldsymbol{P}(\boldsymbol{co})}{\boldsymbol{P}(\boldsymbol{cd})} \text{ ------ (8)}$$

The naive Bayes classifier has several specific uses, such as in text analysis. Text categorization refers to the process of labeling texts with categories based on their content. Both categorizing and labeling texts fall under the umbrella term "classification".

"Relational data extraction" (RE) refers to the process of identifying and understanding the relationships between items in a text. Relation extraction is crucial because data from disorganized sources often appears as unstructured text. Access to an organization's internal semantic links is useful in several areas, including information extraction, language acquisition, and knowledge discovery. Equation 9 provides a mathematical expression for this connection.

$$t = (e_1, e_2, \dots, e_n) \text{ ------ (9)}$$

Wheree_i represents the parties to a connection R that has already been established in a document D.

In the sentence "NLP implemented in logistics," for example, the term "implemented in" demonstrates the connection between the two ideas. In the form of triples (NLP, utilized by, Logistics), this association is recorded and kept track of. The first method is often used to identify connections between objects, and it is open and may potentially extract meaningful information directly from raw text with little involvement. Supervised information extraction, which draws on previously acquired wisdom, may achieve the same goal.

3.5 Enhanced POS Tagging with NLP

To that end, natural language processing (NLP)researchers have developed a cutting-edge technique called enhanced Part-of-Speech (POS) tagging. Assigning grammatical tags to each word in a sentence (a process known as "POS tagging") is a crucial operation in natural language processing that makes it possible to analyze the syntactic structure and meaning of text. Words are given POS tags based on their context and linguistic features using traditional POS tagging approaches, which depend on pre-defined sets of rules and statistical models. These approaches have reached a decent level of accuracy, but they often struggle when confronted with ambiguous vocabulary, complicated sentence structures, and linguistic differences. To get over these restrictions, advanced POS tagging methods use natural language processing to make POS taggers more precise and reliable. Improved tagging accuracy is achieved via the use of these methods by using sophisticated linguistic characteristics, machine learning algorithms, and extensive linguistic resources N. R, P. M. S, P. P. Harithas and V. Hegde (2022).

There are a few crucial measures in using NLP for improved POS tagging. At the outset, we amass a huge, multilingual, multi-genre corpus of annotated text data. For the purpose of creating reliable POS taggers, this corpus is used as both training and assessment data. Tokenization, sentence segmentation, and morphological analysis are some of the pre-processing methods used on the text data next. These procedures guarantee that the text is properly divided into words or tokens and those linguistic properties like word forms and affixes are recorded. Machine learning algorithms and deep learning models are only two examples of the cutting-edge NLP approaches being used to improve POS tagging. Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs) are only two examples of machine learning algorithms that may be trained on an annotated corpus to discover the statistical patterns and correlations between words and their POS tags. There is also encouraging evidence from the use of deep learning models, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and transformer models, to the task of POS tagging. These algorithms are able to accurately anticipate POS tags because they are trained on massive annotated corpora, which captures subtle linguistic patterns and context.

3.6 Word net

WordNet is quite helpful since it is one of the biggest and most in-depth dictionaries of the English language. It aims to simulate the vocabulary of a native English speaker. More than 150,000 nouns, verbs, adjectives, and adverbs are included in the database as synsets. The synsets provide each word a tree-like hierarchical structure by classifying links between terms in terms of hyponym/hypernym (i.e. Is-A) and meronym/holonym (i.e. Part-Of) pairs. Many studies have examined WordNet's flexibility with respect to IR techniques, focusing on the semantic similarity of recovered words and their relationship to clustering approaches. They choose hypernyms/synonyms in 'levels' to extend the bisecting k-means using WordNet data and fuzzy association rules, however they find that noise is skewing their clustering findings. We believe that, in contrast to their approach, you may be able to sidestep this problem by using a proper weighting mechanism for WordNet hypernyms. The authors of explore WordNet's potential as a disambiguation resource by categorizing stemmed terms. While their method does improve the performance of existing clustering techniques, it seems to over generalize the affected words. Although Abdelmalek et al. recognize the possibility of information loss owing to the ambiguity inherent in matching words to concepts in ontology; they still seek to minimize dimensionality. We argue that none of these methods adequately take into account WordNet hypernyms to increase the scope of search terms.

Labeling is the sole task that can be performed by existing systems that utilize external sources like Wikipedia. There have been significant advancements in document clustering algorithms, including methods like "fuzzy frequent item set-based document clustering" integrating the contextual information provided by WordNet hypernyms with fuzzy association rule mining. The authors attribute the long running time (despite if it grows linearly with the number of documents) to fuzzy association mining and the first stages of clustering. However, we are almost finished with a system that can rapidly and concurrently build clusters and labels.

3.7 Training using CNN and dense architecture with resnet50

3.7.1 CONVOLUTIONAL NEURAL NETWORK

To achieve this, a convolutional layer takes the area of interest and applies it to a collection of filters, which are kernel functions. In most cases, the input is a picture with the dimensions M x N x C. The image's dimensions, M and N, are shown. The number of color channels, denoted by C, is typically three (RGB). A feature map (with dimensions W x Q x K) is the product of a convolutional layer. Equations (10) and (11) may be used to calculate W and Q, respectively H. T. Phan et al. (2022).

$$W = \frac{M-F+(2\times ZP)}{S} + 1 \text{ ------ } (10)$$

$$Q = \frac{N-F+(2\times ZP)}{S} + 1 \text{ -------- } (11)$$

Figure 3.4 shows a visual representation of the operation of a convolutional layer. After zero padding is applied to the outside borders of the blue-colored input, the volume increases to 7 × 7 x 3 (M = N = 5, C = 3).

There are two red filters shown, each with a volume of three cubic units (F = three, K = two). The three levels are shown sequentially. The input matrix is convolved with the filter at ZP = 0 and S = 2 where ZP is the zero padding and S is the stride. The feature map's dimensions are calculated using Equation (12) with these inputs.

$$W = Q = \frac{M-F+(2\times ZP)}{S} + 1 = \frac{5-3+(2\times 1)}{2} + 1 = 3 \quad \text{-------} \ (12)$$

The feature map's green color indicates that its dimensions are 3 by 3 by 2, or W x Q x K. To separate less-important characteristics from more-important ones, the kernel matrix includes a variety of weights. Values in the kernel matrix are used as filter weights. The likelihood of a pattern occurring is shown not just through the use of weights, but also via the provision of bias values. Parameters of a network are referred to as weights and biases. All additional characteristics are referred to as hyper parameters and are learnt by the network during the training phase. Equation (13) is used to determine the output matrix's values.

$$Output = (InputxFilter) + \ bias \ \text{------} \ (13)$$

CNNs are special because numerous neurons use the same filter. By recycling the region's bias and weight vectors, we can drastically cut down on data storage needs.

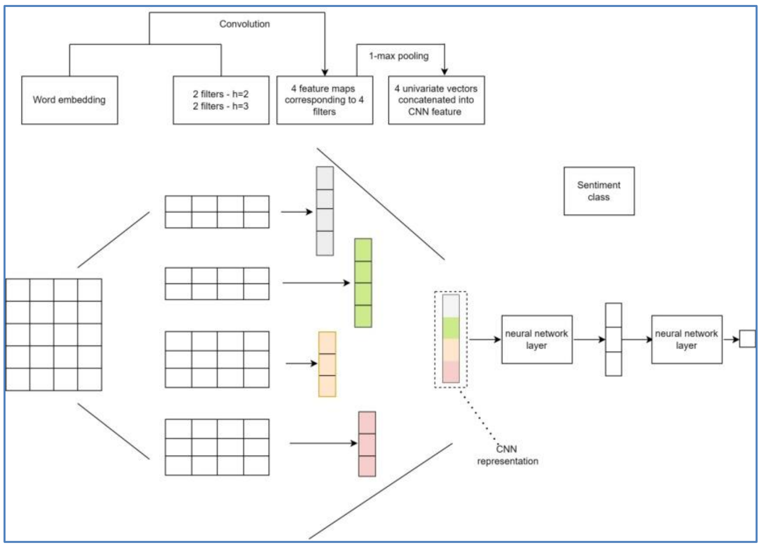


Fig 2: CNN architecture

3.7.2Dense architecture

Using the strengths of the residual attention mechanism to emphasize the relevant characteristics and downplay the noise, the DenseNet network model on the dataset of power equipment may achieve better classification accuracy. DenseNet is a network model that makes use of the residual attention mechanism. In the above diagram, x represents the input, which is down-sampled by the mask branch on the left in order to facilitate the extraction of more valuable information, and then up-sampled so that it is on par in size with the input. The final result is a feature map designated by T that combines the attention map M (x) with the right-hand trunk branch (x). Important characteristics are brought to the forefront while less crucial ones are hidden using the mask branch's attention graph, which incorporates pixel values weighted according to their relevance in the original feature graph. Point-multiplying the feature map outputs from the mask and stem branches yields an equation for the Attention Mechanism network model output:

$$H_{i,c}(x) = \left(1 + M_{i,c}(x)\right) * T_{i,c}(x) \ \text{-----} \ (14)$$

, where (x) I c M ranges from 0 to 1, and (x) I c T represents the position of the ith pixel point if the feature map from the trunk branch matches the location of the cth channel; incorporating this mask branch allows the model to pick up information from new inputs.

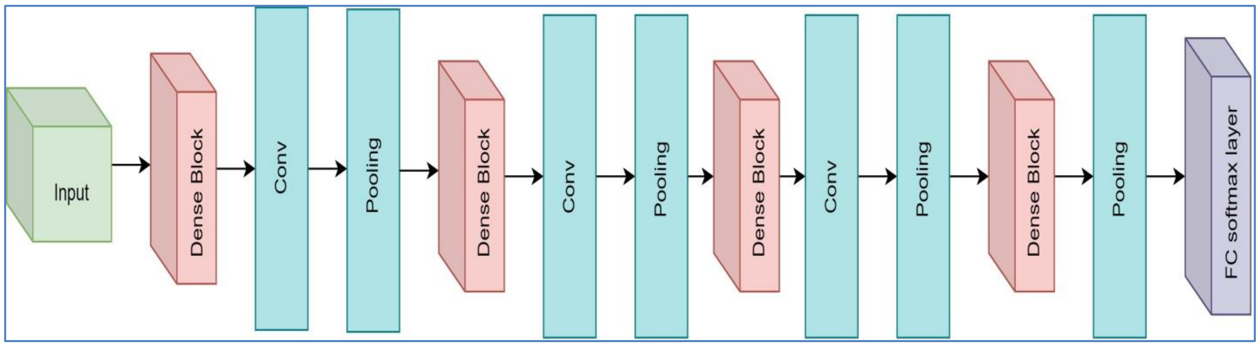


Fig 3: Dense architecture

3.7.3RESNET 50

In order to standardize the input, it employs a label encoder. If a label does not include a number, it will be replaced with one that does. Tokenize converts each word in a text into a series of integers or a vector with a binary coefficient based on tools like the tf-idf or word counts.

Token frequency, abbreviated tf_{ij} , is the sum of all occurrences of a token in a content record. The fraction of equation 15's total tokens that this token occurs in is the number of times this token appears in the content record.

$$\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{ij}} \text{------ (15)}$$

The Inverse Data Frequency (IDF) statistic is used by statisticians to calculate the frequency with which unexpected tokens appear in historical files. Tokens that appear seldom (16 times total) in the record document are more likely to be significant.

$$\mathrm{df}(w) = \log\left(\frac{N}{\mathrm{df}_i}\right) \text{ ------- (16)}$$

The TF-IDF score for a word is the sum of its TF score and its IDF score. In particular, I'm referring to equation 17,

$$W_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{N}{\mathrm{df}_i}\right) \text{ ------- (17)}$$

$\mathrm{tf}_{i,j}$= counting the occurrences of I in j

df_i= records where I is the id value

N = the whole count of files

The model is trained by first converting tokens to word sequences using a text to sequence tool.

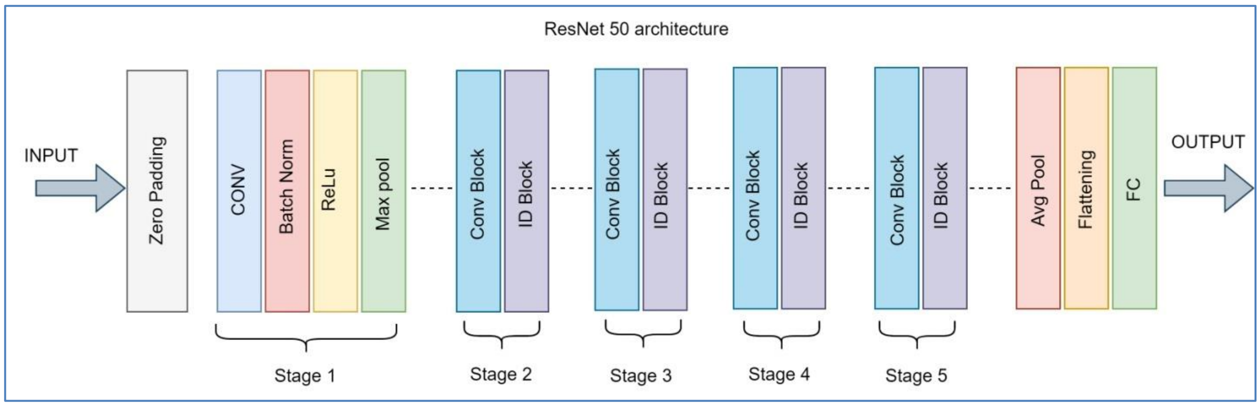


Fig 4: ResNet 50 architecture

3.7.4DenseNet architecture with Resnet50

Different deep neural network topologies, such as DenseNet and ResNet50, have both proven useful in computer vision applications. However, textual information may also benefit from them if it is seen as a series of inputs. DenseNet and ResNet50 both excel in feature extraction, and by combining them we can develop a robust architecture for use in sentiment analysis. By introducing dense connections, DenseNet ensures that every layer receives inputs from the layers above it. This allows the model to capture both local and global relationships within the text data since it encourages feature reuse and information flow. As a result of the dense connection, more discriminative characteristics may be learned with a smaller set of network parameters.To combat the issue of disappearing gradients in deep networks, ResNet50 makes use of residual connections. ResNet50 permits training of deeper networks while minimizing the degradation issue by providing skip connections that transfer the input straight from one layer to a subsequent layer. In the case of text data, this is especially helpful for capturing complicated patterns and hierarchical representations.

Utilizing the strengths of both DenseNet and ResNet50 allows us to achieve optimal performance. While DenseNet's dense connections help with feature reuse and capturing textual relationships, ResNet50's residual connections allow for the training of deeper networks and help with the vanishing gradient issue. This integrated design facilitates the rapid identification of relevant and distinguishing characteristics in text data for sentiment analysis.

Combining DenseNet and ResNet50 for Sentiment Analysis:

- In the context of sentiment analysis, we can adapt DenseNet and ResNet50 to process text data.
- Text data is treated as a sequential input, and the convolutional layers are used to extract features.
- DenseNet's dense connections capture dependencies and promote feature reuse within the text data.
- ResNet50's residual connections allow for the training of deeper networks and capture complex patterns.
- By combining DenseNet and ResNet50, we can leverage the advantages of both architectures to efficiently extract meaningful and discriminative features from text data for sentiment analysis.

Algorithm3:DenseNet architecture with Resnet50

Input:

- Dataset of power equipment with labeled examples for training and evaluation.
- Preprocessed input data, represented by x, which could be down-sampled or transformed as described in the previous explanation.
- Label encoder to normalize the input and convert labels to corresponding numerical values.
- TF-IDF or other tokenization techniques to transform words into integer sequences or vectors.

Output:

- The output of the model is represented by H(i,c)(x) and denotes the feature map incorporating both the attention map M(x) and the trunk branch output T(x) as described in equation (10).
- H(i,c)(x) represents the feature map output for the ith pixel point and cth channel location.
- The attention map M(x) includes pixel values weighted according to their importance in the original feature map, highlighting important features and downplaying less essential ones.
- The trunk branch output T(x) represents the feature map obtained from the right side of the architecture, which may capture additional information.
- The algorithm output can be further used for classification or other downstream tasks related to power equipment analysis.

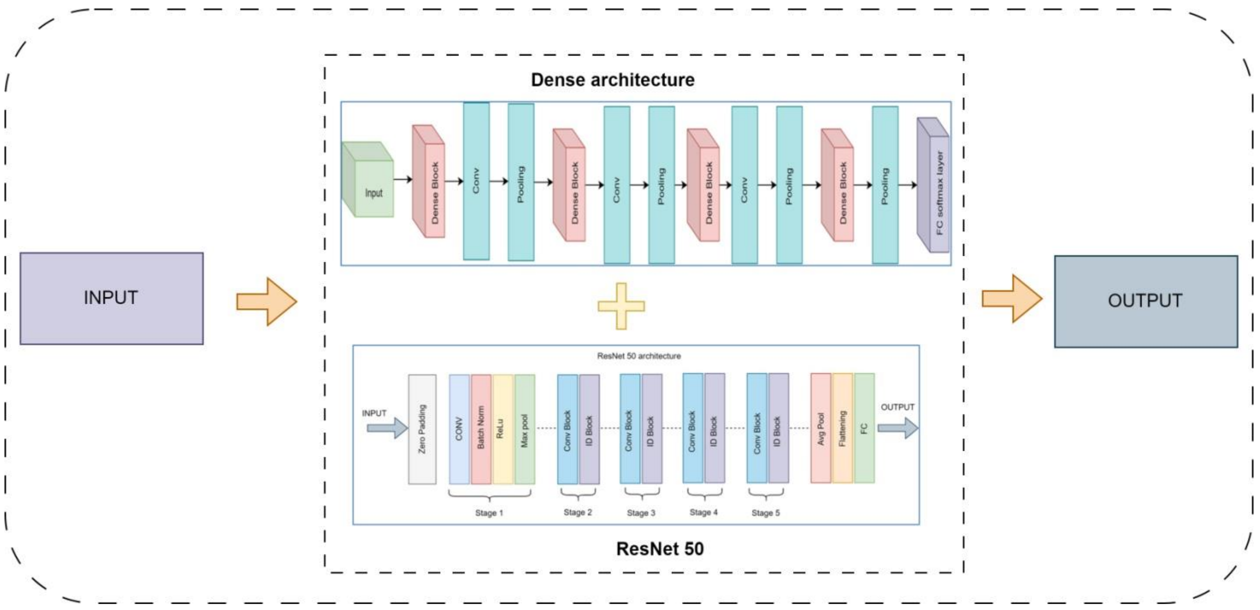


Fig 5: DenseNet architecture with Resnet50

3.8 Ensemble feature selection using Elastic Net

3.8.1 Elastic Net

In the sentence "NLP implemented in logistics," for example, the term "implemented in" demonstrates the connection between the two ideas. In the form of triples (NLP, utilized by, Logistics), this association is recorded and kept track of. The first approach is open and may extract significant information straight from raw text with minimum intervention; it is often used to find links between items. To this end, supervised information extraction, which makes use of previously learned knowledge, might be useful. 1, 22, and 23several researchers in the fields of pattern recognition and machine vision have turned to the EN approach as a clustering algorithm26 to solve difficulties with appropriate pairing. 27 While the EN method is stiff and prone to produce longer TSP tours when the number of cities in the problem exceeds 100, it outperforms simulated annealing, k-opt exchange heuristics22, and the Lin-Kernighan17 algorithm. 28 Because of its adaptability, the EN method is very competitive in structural optimization. We will talk about the benefits of this strategy and how they stack up against those of other stochastic methods like simulated annealing.

A salesperson only has to visit each city once, thus he can easily do it by following a set route that takes him there and back again. $R = R_1, \ldots R_n$Represents the whole route, with n tour points; c represents the J^{th} tour point.

$l = \sum_{j=1}^{n} |R_{j+1} - R_j|$ ------ (18)

is at an all-time low throughout the world. In the elastic net (EN) approach, this is accomplished by using two groups of harmonic springs with zero beginning lengths. A complete circuit between the tour nodes is established by the initial gathering of n springs. Therefore, these points of connection along a tour route are referred to as tour springs. Each tour spring stores energy equal to t $R_{j+1} - R_j$, where R_j and R_{j+1} are the spring's radii. Having access to such a beautiful natural spring certainly makes the journey seem shorter. The energy levels of the second set of mn springs link the cities Si and R_j on the tour route. Sometimes you need to make a salesman's trip to get to these city havens.

3.9 DECISION TREE ALGORITHM

When it comes to making decision trees, there is a tried and true method that is used as a standard. ID3 evaluates the splitting function using a concept called information gain, which is derived from the idea of information entropy. In physics, the entropy of a thermodynamic system is a measure of how much disorder there is. This instrument allows for the quantification of analytical uncertainty. As the order of a system grows, the entropy of the information it contains decreasesAlahmarys, R. M. et al. (2019).

However, an increase in chaos results in a rise in information entropy. The entropy of information may be utilized in this manner to quantify a system's degree of organization. The data set s has n observations, evenly distributed across m categories. The total number of samples in group i is denoted by ni, thus we may divide them into m separate groups and label them C_i (I = 1, 2,...). The required information entropy for efficiently assigning n samples to m categories is

$S(q) = \sum_{i=1}^{m} P_i \log_2 p_i$, ------ (19)

The formula pi = ni / n may be used to calculate the likelihood that every data point in sample s belongs to class C_i. The sample will be divided into subsets based on the values of property A if the splitting attribute is specified. We calculate the entropy of the information utilized to split the samples into each group.

$S(D_j) = \sum_{i=1}^{m} P_i \log_2 P_i$,------- (20)

whereD_j is some subset of D, and pi is the chance that any random element from D_j belongs to some class C_i. Therefore, attribute D, the minimal entropy of information required to identify all samples, is

$S(D) = \sum_{v=1}^{v} \frac{sv}{s} S(D_j)$,------- (21)

The importance of an attribute value is represented by the ratio of the number of samples having that value $\sum_{v=1}^{v} \frac{sv}{s}$ to the total number of samples (q). For this reason, data has

$gain = S(q) - S(D)$-------- (22)

ID3 choose between two split attributes using gain-based criteria. Previous studies have indicated that the character with the highest number of attribute values benefits the most from this newfound information. A larger number of attribute values does not always indicate a higher quality decision tree. Consequently, it is crucial to settle the conflict between attribute and value selection.

Algorithm: 4 DECISION TREE ALGORITHM

Input:

- Dataset D, consisting of n observations and their corresponding attributes
- Target attribute to be predicted

- Set of attributes A

Output:

- Decision tree that predicts the target attribute

```
1. Create a new node N.
2. If all observations in D belong to the same class C, assign N as a leaf node labeled with class C and return.
3. If the set of attributes A is empty, assign N as a leaf node labeled with the majority class in D and return.
4. Calculate the entropy of the current dataset D using Equation (19).
5. For each attribute in A, calculate the information gain using Equation (22).
6. Select the attribute with the highest information gain as the splitting attribute and assign it to N.
7. For each possible value v of the selected attribute: a. Create a new branch from N labeled with attribute = v. b. Split the dataset D into subsets based on the value v of the selected attribute. c. If the subset for value v is empty, assign the branch with a leaf node labeled with the majority class in D and return. d. Otherwise, remove the selected attribute from the set A. e. Recursively call the ID3 algorithm on the subset with the updated set of attributes A. f. Attach the resulting subtree to the corresponding branch from step 7a.
8. Return the decision tree rooted at N.
```

3.10RANDOM FOREST

Random Forest is a supervised learning method that may be used as an ensemble learning strategy for these and other issues by training a large number of decision trees for a specific objective (classification, regression, etc.), and then aggregating their predictions into a single output.

A decision tree is a kind of decision-making diagram that graphically represents a set of alternatives in the shape of a tree. Every leaf node in the tree stands in for a unique category, while every other node represents a feature test and every branch displays the attribute result over numerous values. The decision tree starts at the root node and works its way out to the leaf node, all the while considering the attributes of the target category and their related features. The category stored at the leaf node is used to make the ultimate decisionDay, M.-Y., & Lin, Y.-D. (2017).

A random forest is a collection of independent decision trees. The Gini index was used to classify the properties of the decision tree, and the d parameter of the algorithm determined the length of each branch.

$$G(X_i) = \sum_{j=1}^{j} pr\left(X_i = L_j\right)\left(1 - pr\left(X_i = L_j\right)\right) \text{------ (23)}$$

$$1 - \sum_{j-1}^{j} pr\left(X_i = L_j\right)^2 \text{-------- (24)}$$

Random Forest outperforms other machine learning algorithms in the following ways, which lead us to choose it:

It's a great method for missing data forecasting that works even when a lot of information is lacking.

Algorithm 5:RANDOM FOREST

Input:

- Dataset D, consisting of n observations and their corresponding attributes
- Number of decision trees to be created, num_trees
- Number of randomly selected features at each node, num_features
- Output:
- Random Forest ensemble of decision trees

```
1. Create an empty list, forest, to store the decision trees.
2. Repeat steps 3 to 8 for t = 1 to num_trees: a. Create a bootstrap sample, D_t, by randomly selecting n observations with replacement from D. b. Create a new decision tree T_t using the dataset D_t and the following modifications: i. For each node in T_t: - Randomly select a subset of num_features features from the available attributes. - Evaluate the best attribute and split point based on a criterion (e.g., Gini index or information gain). - Split the node into child nodes based on the selected attribute and split point. c. Add the decision tree T_t to the forest.
3. Return the forest containing the ensemble of decision trees.
```

3.11Optimization using ALS

3.11.1 ALTERNATIVE LEAST SQUARE

When both q_i and P_u are undefined, Eq. 25 is not convex. When one of these issues is resolved, however, optimization becomes a tractable quadratic problem. As a result, ALS method alternates between mending the qi and the P_u. In order to recalculate the q_i once all the P_u have been adjusted, the system solves a least-squares issue. Consequentially reducing Eq. 25 in this manner guarantees convergence.

$min(p,q)\sum_{(u,i)\in k}(r_{ui}-q_i^T p_u)^2+Y\left(\left||q_i|\right|^2+\left||P_u|\right|^2\right)$ ----- (25)

Even though ALS is more computationally intensive than SGD, it provides advantages in at least two scenarios. The first is seen in systems that rely heavily on implicit data and involve working with matrices that are densely packed. Looping through each training example (as in the case of SGD) is impractical since the training set cannot be deemed sparse. The second scenario is when parallelization is a viable option for the system. The method is amenable to parallelization since each q_i and p_u may be computed independently of one another and from other item and user characteristics.

Example: if we want to recomputed the ith row pi of the user feature matrix P, we need to look at the ith row of R (which includes user i's interactions) and the ith column q_i^T of Q (which corresponds to non-zero values inq_i^T). As long as efficient data access to the rows of R and the required columns from Q is controlled properly, it is simple to parallelize P's re-computation, since the re-computation of pi does not depend on the re-computation of any other rows in P. In ALS, one iteration is defined as the process of first recalculating P and then recalculating Q. The ALS algorithm's main components are outlined in Algorithm 1.

In order to compute ALS, we must connect the interaction data R and Q (the item attributes) at the same time that we recomputed the rows of P. Then, Q is recalculated by performing a parallel join on R and P (the user features). Since the limitation of any parallel solution is the available network bandwidth, it is essential to develop an effective execution strategy for these joins.

3.12 Classification using ensemble stacking algorithm and ensemble voting algorithm

3.12.1 ENSEMBLE STACKING

Time I for T samples of real numbers x1,...,xT is the answer (1 I T). By analyzing past data from samples x1...xw (w+h T) (1 I h), time series forecasting attempts to reduce the deviation between the predicted and observed xw+i values. Both the historical window, denoted by w, and the forecast horizon, denoted by h, must be specified in this formulation.

Traditionally, time series is divided into three parts:

In statistics, a "trend" refers to the overall direction of change over time. There are several possible patterns in a time series (square root), including linear, logarithmic (power), exponential, and polynomial.

Second, the "trend" of a time series reveals its general direction. A time series may display a variety of interesting patterns. Polynomial (square root) and exponential (power) trends are two further examples of nonlinear patterns.

What little time series data we have left is shown here. Importantly, it shows how seasonality and patterns may be obscured by huge, irregular piecesPasupa, K. et al. (2019).

Persistent tendencies, including seasonal effects, might be shown by further breakdown patterns. Real-world time series are notoriously difficult to predict because of their intrinsic unpredictability. - Test for the stationarity of a time series. The goal is to see whether the mean and standard deviation of a time series stay the same throughout time. Before incorporating a no stationary time series into a forecasting model, it must first be transformed. Studies of time series may be classified as either univariate or multivariate, depending on the number of independent variables being compared. In a univariate time series, the information is presented in a linear fashion. In multivariate time series, numerous variables' values are tracked across all points in time. It is essential to grasp the relationship between these variables. There are a number of techniques that may be used to make predictions about time series. There are both linear and nonlinear options available for techniques. The time series is linearly expressed via linear techniques. Even if there is a strong correlation between individual observations and their combination, a linear combination of this historical data, leaving out additive noise, may still be able to predict the future. Nonlinear techniques are increasingly used in machine learning. These techniques try to predict the course of a time series using nonlinear data. Although conventional methods may still provide sufficient outcomes for linear circumstances, there has been a rise in the usage of machine learning techniques for nonlinear modeling prediction.

3.12.2 Ensemble voting algorithm

By combining their strengths, several approaches may provide a more robust outcome in a voting system. Compared to using only one model as a starting point, it may improve accuracy. Classification using this approach has been used to boost efficiency in a variety of fields. This voting approach will be used in this research instead of a single model since it produces more trustworthy findings. For the single model, you may use whatever machine learning approach you like, such as logistic regression, decision trees, support vector machines, and so on. The base model is the sole input model of a voting ensemble approach.

The results of votes are used to determine groups. Using the training dataset, different categorization models are first created. Here, the foundational algorithm may be created in a few different ways, either by splitting the same method across many datasets or by employing a different algorithm on the same dataset.

By having each model cast a prediction (vote) for each test instance, Majority Voting determines which value has the most support and uses it as the final anticipated output. Unstable prediction occurs when no one value receives more than 50% of the vote. Each classifier's predictions are treated as a 'vote' in the final tally. The majority vote from each classifier is counted as the final prediction.

Maximum voting, often known as hard voting, and minimum voting are the two voting methods available. In the hard voting procedure, each classifier makes a separate prediction and the majority vote is tallied.

If there are two classes (0, 1) and three classifiers (A, B, C) predict them, then after training, Classifier A predicts class 0 while Classifiers B and C predict class 1, then two out of three classifiers (A, B, C) predict class 1, and the ensemble choice will be class 1.

However, with soft voting, we average the class probabilities over all classifiers to arrive at an estimate of the chance of a certain class being predicted. In this example, we will use the soft voting method with group 1.

Weighted voting is another method of voting. In contrast to majority voting and soft voting, where all models are given equal weight, we may give greater weight to certain models than others. In weighted voting, the forecast of a stronger model may be counted more than once.

4. Results and Discussion

The proposed model has implemented in Python programming.

$$ACCURACY = \frac{(TP+TN)}{(TP+FP+FN+TN)} \text{ ----------- } (26)$$

$$Precision \frac{TP}{TP+FP} \text{ ------------ } (27)$$

$$Recall \frac{TP}{TP+FN} \text{ ------------ } (28)$$

$$F - Measure \frac{2\, x\, (precisionxrecall)}{preceision+recall} \text{----------}(29)$$

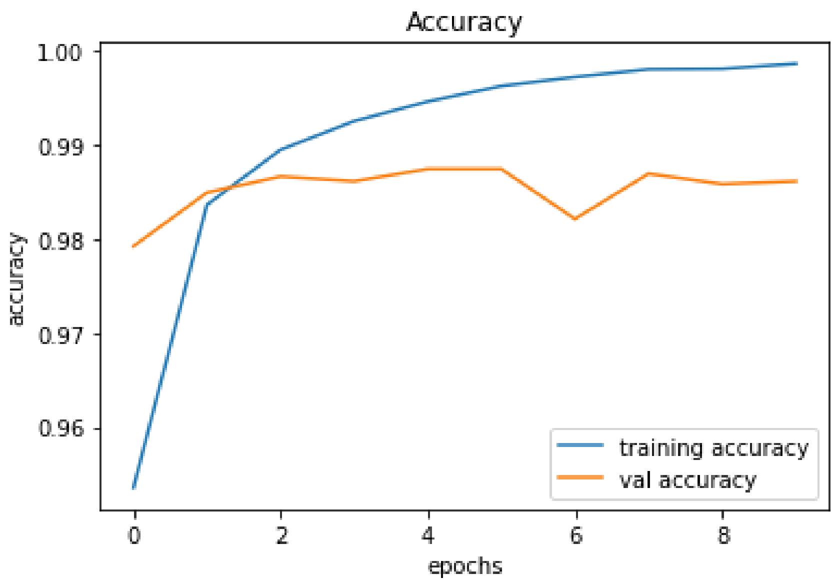


Fig 6: Training accuracy

Figure 6 depicts the training accuracy at various epochs throughout the model training procedure. The epochs are shown by the x-axis and may be thought of as a single loop of the full dataset through the model during the training phase. Each epoch entails running all of the training data through the model, updating its internal parameters (weights and biases), and optimizing them to minimize the training error. The training accuracy at each epoch is shown by the y-axis. Training accuracy is a measure of how well a model performs on training data at a given epoch. It is obtained by comparing the model's predictions to the training dataset's actual labels. A greater training accuracy implies that the model is learning and fitting the training data better as more epochs pass.

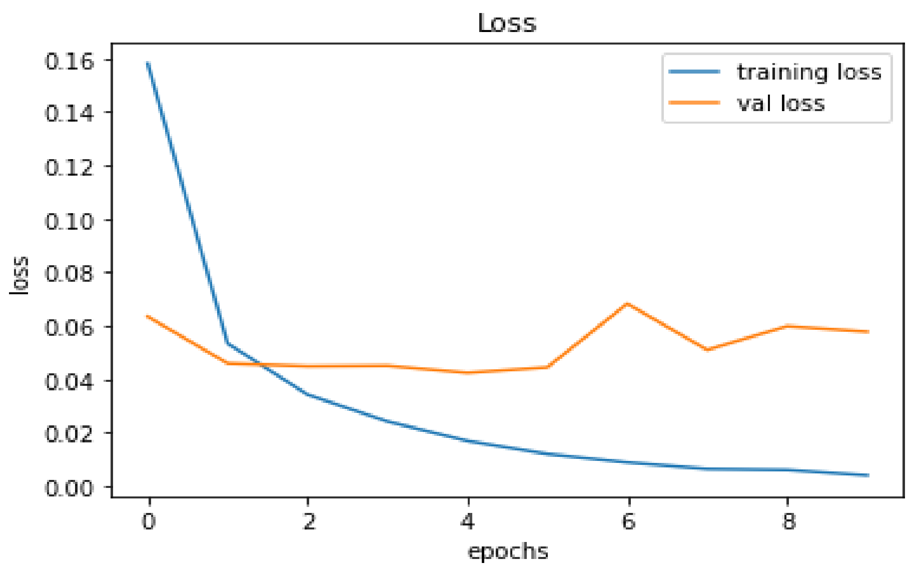


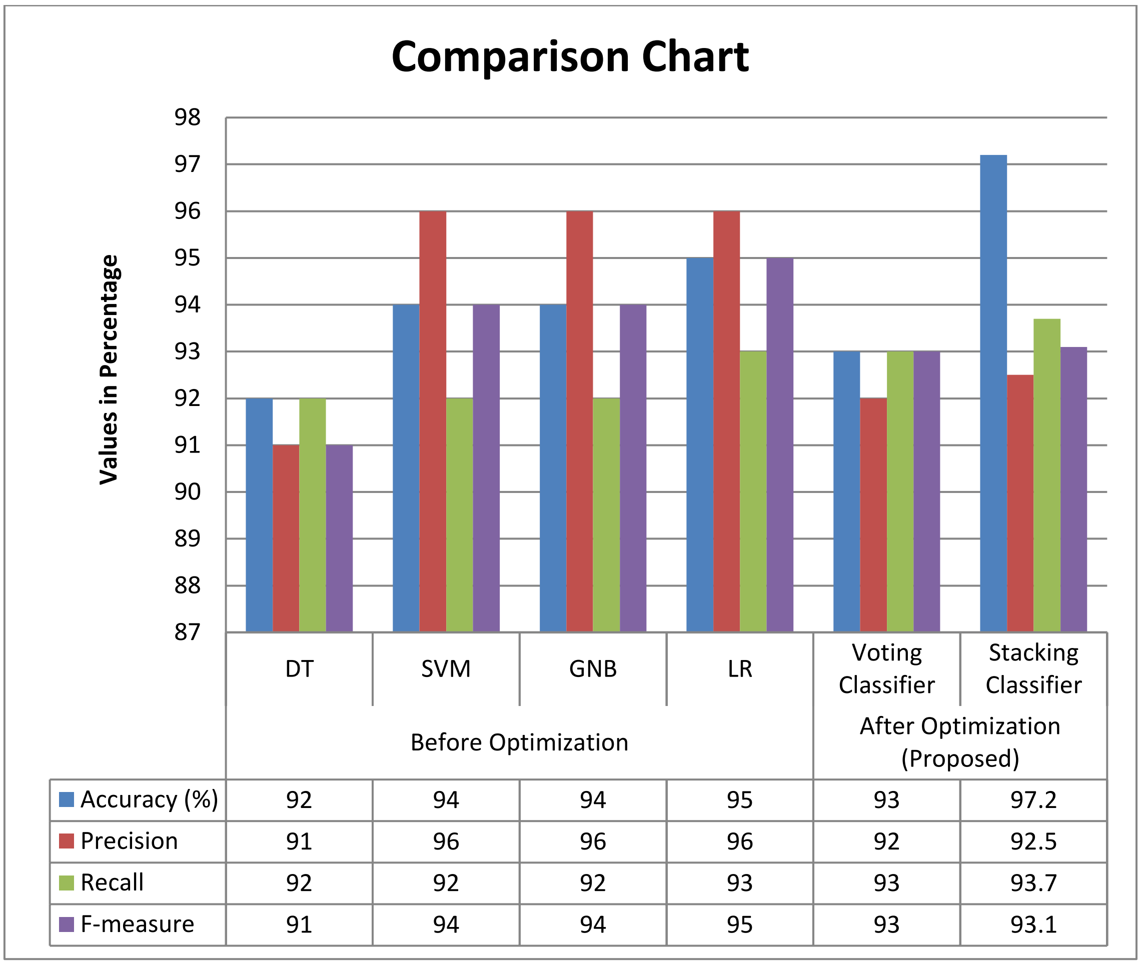
Fig 7: Training loss

Figure 7 depicts the training loss at various epochs throughout the model training procedure. The x-axis shows epochs, which are the repetitions of the full dataset that the model goes through throughout training. The model produces predictions on the training data at each epoch, compares those predictions to the actual labels, and determines the training loss. The training loss at each epoch is shown by the y-axis. The training loss is a measure of how closely the model's predictions match the training data labels. It measures the discrepancy between expected and actual values. The aim during training is to minimize this loss, since lower training loss shows that the model is better precisely fitting the training data.

Table 2: performance metrics comparison

Method	Model	Accuracy (%)	Precision	Recall	F-measure
Before Optimization	DT	92	91	92	91
	SVM	94	96	92	94
	GNB	94	96	92	94
	LR	95	96	93	95
After Optimization (Proposed)	Voting Classifier	93	92	93	93
	Stacking Classifier	97.2	92.5	93.7	93.1

Table 2 shows the performance characteristics of several machine learning models both before and after optimization. Four models were assessed prior to optimization: Decision Tree (DT), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), and Logistic Regression (LR). SVM, GNB, and LR all showed high accuracy levels ranging from 94% to 95%, with SVM having the best precision of 96%. DT, on the other hand, performed somewhat worse, with 92% accuracy, precision, recall, and F-measure. Following optimization, two new ensemble models, the Voting Classifier and the Stacking Classifier, were introduced. The Voting Classifier performed consistently well, achieving 93% accuracy, precision, recall, and F-measure. In contrast, the Stacking Classifier improved significantly, obtaining an excellent accuracy of 97.2% and a recall of 93.7%. The accuracy, however, was significantly lower at 92.5%, resulting in an overall F-measure of 93.1%. The optimization efforts resulted in considerable gains in model performance, especially with the development of the Stacking Classifier, which outperformed all prior models in accuracy while maintaining an acceptable balance of precision and recall.



	DT	SVM	GNB	LR	Voting Classifier	Stacking Classifier
	Before Optimization				After Optimization (Proposed)	
🟦 Accuracy (%)	92	94	94	95	93	97.2
🟥 Precision	91	96	96	96	92	92.5
🟩 Recall	92	92	92	93	93	93.7
🟪 F-measure	91	94	94	95	93	93.1

Fig 8: performance metrics comparison

The figure 8 shows performance metrics the x axis shows methods and the y axis shows value.

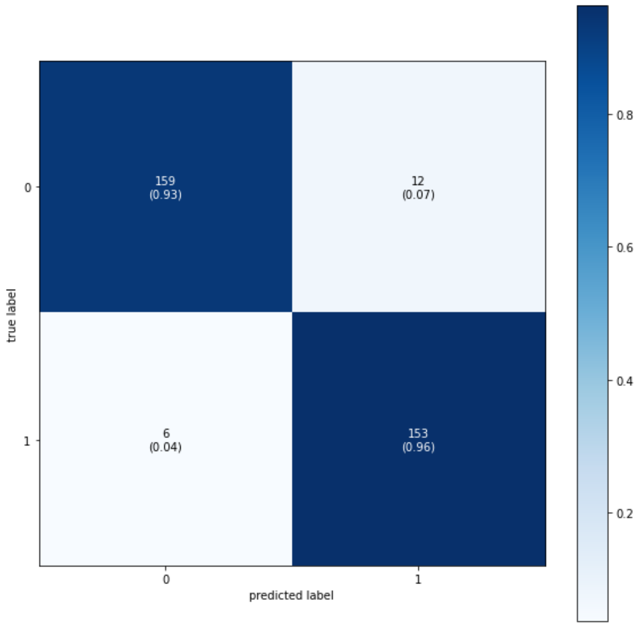


Fig 9: stacking classifier Confusion matrix

The figure 9 shows stacking classifier Confusion matrix TP is 159 and TN is 153, FP is 12 and FN is 6.

5. Conclusion

This paper introduced a deep learning sentiment analysis approach based on optimization. To measure text sentiment, we created a multi-module system. The data set gathering module offered labeled sentiment data for model training and assessment. The pre-processing module uses NLP and WORDNET to enhance text semantics and sentiment analysis. We employed a CNN and Dense architecture with Resnet50 to extract key text properties for the deep learning training module. The model acquired complex sentiment patterns and representations, increasing the accuracy of sentiment analysis. The selection of ensemble features enhanced. The most significant and relevant features were found using Elastic Net, Recursive Feature, and Hybrid ML Classifier to increase sentiment analysis model efficiency. It was critical to optimize model parameters. The model was fine-tuned using the Alternative Least Squares (ALS) approach, which improved performance and sentiment analysis. In classification, the Ensemble Stacking and Ensemble Voting algorithms were utilized. To conclude the stacking ensemble classification has achieved highest accuracy as 97.2 percentages. These strategies improved sentiment analysis classification accuracy by using the strengths of many models. Deep learning-based sentiment analysis that employs optimization techniques is both robust and accurate. To capture complicated attitudes in text data, we developed a full sentiment analysis system that employs advanced deep learning architectures, feature selection, and optimization techniques. Future research may expand the system to support several languages, investigate novel optimization strategies, and include domain-specific data to enhance sentiment analysis in real-world contexts.

Reference

[1] Abburi, H., Prasath, R., Shrivastava, M., &Gangashetty, S. V. (2017). Multimodal Sentiment Analysis Using Deep Neural Networks. Lecture Notes in Computer Science, 58–65. doi:10.1007/978-3-319-58130-9_6

[2] Alahmary, R. M., Al-Dossari, H. Z., & Emam, A. Z. (2019). Sentiment Analysis of Saudi Dialect Using Deep Learning Techniques. 2019 International Conference on Electronics, Information, and Communication (ICEIC). doi:10.23919/elinfocom.2019.8706408

[3] Al-Dabet, S., Tedmori, S., & AL-Smadi, M. (2021). Enhancing Arabic aspect-based sentiment analysis using deep learning models. Computer Speech & Language, 69, 101224. doi:10.1016/j.csl.2021.101224

[4] Chandra, Y., & Jana, A. (2020). Sentiment Analysis using Machine Learning and Deep Learning. 2020 7th International Conference on Computing for Sustainable Global Development (INDIACom). doi:10.23919/indiacom49435.2020.9083703

[5] Day, M.-Y., & Lin, Y.-D. (2017). Deep Learning for Sentiment Analysis on Google Play Consumer Review. 2017 IEEE International Conference on Information Reuse and Integration (IRI). doi:10.1109/iri.2017.79

[6] De Oliveira Carosia, A. E., Coelho, G. P., & da Silva, A. E. A. (2021). Investment strategies applied to the Brazilian stock market: A methodology based on Sentiment Analysis with deep learning. Expert Systems with Applications, 184, 115470. doi:10.1016/j.eswa.2021.115470

[7] Demirci, G. M., Keskin, S. R., & Dogan, G. (2019). Sentiment Analysis in Turkish with Deep Learning. 2019 IEEE International Conference on Big Data (Big Data). doi:10.1109/bigdata47090.2019.9006066

[8] Hassan, A., & Mahmood, A. (2017). Deep Learning approach for sentiment analysis of short texts. 2017 3rd International Conference on Control, Automation and Robotics (ICCAR). doi:10.1109/iccar.2017.7942788

[9] H. T. Phan, N. T. Nguyen and D. Hwang, "Aspect-Level Sentiment Analysis Using CNN Over BERT-GCN," in IEEE Access, vol. 10, pp. 110402-110409, 2022, doi: 10.1109/ACCESS.2022.3214233.

[10] Heikal, M., Torki, M., & El-Makky, N. (2018). Sentiment Analysis of Arabic Tweets using Deep Learning. Procedia Computer Science, 142, 114–122. doi:10.1016/j.procs.2018.10.466

[11] Jain, K., & Kaushal, S. (2018). A Comparative Study of Machine Learning and Deep Learning Techniques for Sentiment Analysis. 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO). doi:10.1109/icrito.2018.8748793

[12] Jangid, H., Singhal, S., Shah, R. R., & Zimmermann, R. (2018). Aspect-Based Financial Sentiment Analysis using Deep Learning. Companion of the The Web Conference 2018 on The Web Conference 2018 - WWW '18. doi:10.1145/3184558.3191827

[13] Mittal, N., Sharma, D., & Joshi, M. L. (2018). Image Sentiment Analysis Using Deep Learning. 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI). doi:10.1109/wi.2018.00-11

[14] N. R, P. M. S, P. P. Harithas and V. Hegde, "Sentimental Analysis on Student Feedback using NLP & POS Tagging," 2022 International Conference on Edge Computing and Applications (ICECAA), Tamilnadu, India, 2022, pp. 309-313, doi: 10.1109/ICECAA55415.2022.9936569.

[15] Pasupa, K., & Seneewong Na Ayutthaya, T. (2019). Thai sentiment analysis with deep learning techniques: A comparative study based on word embedding, POS-tag, and sentic features. Sustainable Cities and Society, 50, 101615. doi:10.1016/j.scs.2019.101615

[16] Ramadhani, A. M., & Goo, H. S. (2017). Twitter sentiment analysis using deep learning methods. 2017 7th International Annual Engineering Seminar (InAES). doi:10.1109/inaes.2017.8068556

[17] Roshanfekr, B., Khadivi, S., & Rahmati, M. (2017). Sentiment analysis using deep learning on Persian texts. 2017 Iranian Conference on Electrical Engineering (ICEE). doi:10.1109/iraniancee.2017.7985281

[18] Roy, A., & Ojha, M. (2020). Twitter sentiment analysis using deep learning models. 2020 IEEE 17th India Council International Conference (INDICON). doi:10.1109/indicon49873.2020.9342279

[19] Shilpa, P. C., Shereen, R., Jacob, S., & Vinod, P. (2021). Sentiment Analysis Using Deep Learning. 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV). doi:10.1109/icicv50876.2021.9388382

[20] Singh, J., Singh, G., & Singh, R. (2017). Optimization of sentiment analysis using machine learning classifiers. Human-Centric Computing and Information Sciences, 7(1). doi:10.1186/s13673-017-0116-3

[21] T. Chen, Y. Liang, T. Huang, J. Huang and C. Liu, "Agricultural Product Recommendation Model and E-Commerce System based on CFR Algorithm," 2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI), Changchun, China, 2022, pp. 931-934, doi: 10.1109/ICETCI55101.2022.9832175.

[22] Yadav, A., & Vishwakarma, D. K. (2019). Sentiment analysis using deep learning architectures: a review. Artificial Intelligence Review, 53(6), 4335–4385. doi:10.1007/s10462-019-09794-5

[23] Ekinci, E., & Ilhan Omurca, S. (2018). An Aspect-Sentiment Pair Extraction Approach Based on Latent Dirichlet Allocation. *International Journal of Intelligent Systems and Applications in Engineering*, *6*(3), 209–213. https://doi.org/10.18201/ijisae.2018644779

[24] Coban, O., Ozyildirim, B. M., & Ozel, S. A. (2018). An Empirical Study of the Extreme Learning Machine for Twitter Sentiment Analysis. *International Journal of Intelligent Systems and Applications in Engineering*, *6*(3), 178–184.

[25] Geethanjali, D. ., & Suresh, P. . (2023). Subjectivity Sentence Level Sentiment Analysis and Classification using Correlation Based Embedded Feature Subset using Machine Learning. *International Journal of Intelligent Systems and Applications in Engineering*, *11*(4), 556–562