Journal of Theoretical and Applied Information Technology
15th May 2024. Vol.102. No 9

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EXPLORING ARO-LOGISENT: DELVING INTO DATA COLLECTION FOR ADVANCED SENTIMENT ANALYSIS WITH AMAMI RABBIT OPTIMIZATION-BASED LOGISTIC REGRESSION

D. J. ANITHA MERLIN[1], D. VIMAL KUMAR [2]
[1] Research Scholar, Department of Computer Science, Nehru Arts and Science College, India
[2] Associate Professor, Department of Computer Science, Nehru Arts and Science College, India
E-mail: [1] merlin.celestino@gmail.com, [2] drvimalcs@gmail.com

ABSTRACT

Sentiment analysis, a crucial component of natural language processing, aims to discern the underlying sentiment conveyed in textual data. This research explores the fusion of Amami Rabbit Optimization (ARO) with Logistic Regression (LR) to enhance sentiment analysis performance. ARO, inspired by the foraging behavior of Amami rabbits, offers a novel metaheuristic approach for optimizing model parameters, while LR provides a robust framework for sentiment classification. The proposed integration leverages the strengths of both methodologies to overcome challenges inherent in sentiment analysis, including feature selection, model training, and accuracy optimization. This study investigates the effectiveness of the ARO-LR hybrid approach through empirical experiments conducted on diverse datasets sourced from social media platforms and product reviews. Evaluation metrics suchas precision, recall, F1-score, and accuracy are employed to assess the performance of the integrated model. Results indicate significant improvements in sentiment classification accuracy and robustness compared to traditional LR models. The findings highlight the efficacy of integrating metaheuristic optimization techniques with conventional machine learning algorithms for advancing sentiment analysis capabilities in real-world applications.

Keywords: *Sentiment, Reviews, Online Shopping, Classification, Amazon, Neural Network, Rabbit Optimization, Logistic Regression*

1. INTRODUCTION

Online shopping, also known as e-commerce, has revolutionized the way people shop for goods and services. It refers to the process of purchasing products or services over the internet through websites or mobile applications.[1] With the widespread availability of high- speed internet and the proliferation of digital devices, online shopping has become increasingly popular and convenient for consumers worldwide. One of the key advantages of online shopping is the convenience it offers.[2] Consumers can browse through a vast selection of products from the comfort of their own homes. They can compare prices, read product reviews, and make purchases at any time of day or night, without having to visit physical stores. This convenience is particularly beneficial for busy individuals or those who may have difficulty accessing traditional brick-and-mortar stores[3].

Another benefit of online shopping is the ability to access a global marketplace. Through e-commerce platforms, consumers can shop from retailers and sellers located anywhere in the world, giving them access to a wider variety of products and services than would be available locally. This global reach allows consumers to find unique or niche items that may not be readily available in their area. Online shopping also offers greater flexibility and customization for consumers[4]. Despite its many advantages, online shopping also presents challenges and concerns. Security and privacy are important considerations, as consumers must trust that their personal and financial information will be protected when making online transactions[5]. As technology continues to evolve and e-commerce platforms innovate, online shopping is likely to remain a dominant force in the global marketplace, shaping the way consumers shop for goods and services now and in the future[6].

Sentiment analysis, also known as opinion mining, is a computational technique used to analyze and interpret the sentiment or emotional tone expressed in textual data. It involves identifying and categorizing the sentiment

conveyed in a piece of text as positive, negative, or neutral[7]. Sentiment analysis is commonly applied to various types of text data, including customer reviews, social media posts, news articles, and surveys. It is also valuable in political analysis and public opinion research[8]. By analyzing the sentiment expressed in news articles, social media discussions, and public forums, policymakers and political candidates can gauge public sentiment towards political issues, candidates, and policies. Sentiment analysis is used in financial markets to analyze news articles, social media discussions, and other textual data for insights into investor sentiment and market trends[9]. Traders and investors use sentiment analysis to make informed decisions and predict market movements based on sentiment indicators.

Sentiment analysis plays a crucial role in extracting insights from textual data and understanding the prevailing sentiment or emotional tone within a given context. Its applications span across various industries and domains, helping businesses, organizations, and policymakers make informed decisions, improve customer satisfaction, and stay ahead of emerging trends[10]. Sentiment analysis plays a significant role in online shopping, shaping consumer experiences and influencing purchasing decisions. By analyzing customer reviews and social media mentions, businesses gain insights into customer satisfaction levels and product perceptions[11]. Positive sentiment signals indicate satisfied customers and can lead to increased sales and brand loyalty. Conversely, negative sentiment alerts businesses to potential issues or areas for improvement, allowing them to address customer concerns promptly. This proactive approach enhances the overall shopping experience, fosters trust, and strengthens relationships between businesses and consumers, ultimately driving success in the competitive online marketplace[12].

1.1. Problem Statement

Challenges in sentiment analysis for online shopping arise due to the complexity and nuances of human language. Initially, ambiguity poses a challenge, as words or phrases can have multiple meanings depending on the context, leading to misinterpretation of sentiment. Sarcasm and irony are prevalent in online communication, making it difficult to accurately discern sentiment. Cultural and regional differences in language usage can impact the interpretation of sentiment, requiring a nuanced understanding of linguistic nuances across diverse demographics. Another challenge is the presence of noise in text data, such as spelling errors, abbreviations, and slang, which can hinder sentiment analysis accuracy. Finally the dynamic nature of language and evolving trends in online discourse present challenges in keeping sentiment analysis models up-to-date and relevant. Overcoming these challenges requires robust algorithms, extensive training data, and continual refinement to accurately capture and interpret sentiment in the context of online shopping.

1.2. Motivation

The motivation for this content stems from the increasing importance of sentiment analysis in various fields such as marketing, customer feedback analysis, and social media monitoring. Understanding how sentiment analysis models perform is crucial for researchers, practitioners, and businesses aiming to make informed decisions based on textual data. By evaluating the performance of sentiment analysis models using diverse metrics, this content seeks to provide valuable insights into their effectiveness and reliability, ultimately contributing to advancements in natural language processing and enhancing decision-making processes in real-world applications.

1.3. Research Objective

The objective is to develop robust sentiment analysis models capable of effectively addressing the challenges inherent in natural language processing. This includes devising algorithms and methodologies that can accurately interpret the nuanced aspects of human language, such as sarcasm, ambiguity, and cultural variations. The objective aims to mitigate the impact of rapidly evolving online communication platforms and the sheer volume of user- generated content by developing scalable and efficient sentiment analysis techniques. By overcoming these challenges, the objective seeks to enhance decision-making processes, gain deeper insights into public sentiment, and improve user experiences across various domains, including marketing, customer service, and social media analytics. Ultimately, the goal is to advance the field of sentiment analysis and contribute to the development of more reliable and accurate tools for understanding and interpreting human emotions expressed through text.

2. LITERATURE REVIEW

"News Sentinel"[13] presents a pioneering approach. Employing entity recognition unveils the prominent subjects and entities involved in negative news narratives. Regression analysis then quantifies the impact of various factors on the prevalence of negative news, offering insights into the driving forces behind its dissemination. "Twitter Sentinels"[14] introduces an innovative approach to gauge public sentiment on the development of Rinca Island. Doc2Vec facilitates the conversion of text data into numerical vectors, enabling the analysis of Twitter content. SVM and logistic regression then categorize these vectors into sentiment classes, unveiling the overall sentiment trends.

"FinTweet Analyzer"[15] integrates machine learning techniques to discern sentiment nuances within the financial discourse on Twitter. This model's working mechanism involves training machine learning classifiers on annotated financial tweet datasets, allowing it to generalize and accurately interpret sentiment in real-time tweets. "Sentiment Stock Master"[16] introduces a pioneering method to forecast stock prices by amalgamating sentiment scores from financial news with a Multi-Layer Perceptron (MLP) regressor. The model's working mechanism involves first extracting sentiment scores from financial news articles using natural language processing techniques. "Sentiment Word Refiner"[17] involves augmenting existing word embeddings with sentiment scores derived from sentiment lexicons or machine learning models. By integrating sentiment information directly into the word embeddings, the model learns to encode both the meaning and sentiment of words.

"Deep-Sentiment Analyzer"[18] working mechanism involves the utilization of recurrent neural networks (RNNs) augmented with decision-based mechanisms to capture sequential dependencies and make informed sentiment predictions. By incorporating decision-making processes within the recurrent neural network framework, Deep-Sentiment can effectively analyze sentiment in text data with higher accuracy and efficiency. "Aspect Sentiment Multilocal Learner"[19] a learning model involves incorporating local and global context-focusing mechanisms into the learning model. Local context focusing ensures the model attends to relevant aspects within each sentence, while global context focusing enables the consideration of broader discourse patterns and linguistic nuances across languages.

"LexiCNN Review Analyzer"[20] involves encoding sentiment lexicon information into the CNN architecture, allowing the model to learn intricate patterns and nuances of sentiment expression within online reviews. "Fusion Commerce Analyst"[21] is an approach to understanding e-commerce product experiences through fusion sentiment analysis. The key contribution lies in amalgamating multiple sentiment analysis techniques to comprehensively explore customer perceptions and sentiments towards e-commerce products.

"Sentiment Neural Aspect Analyzer"[22] involves incorporating sentiment lexicons or knowledge bases into the neural network architecture. By integrating this external knowledge, the model gains a deeper understanding of sentiment nuances related to specific aspects of text data. "Swarm Intellect Innovator"[23] spans from simple single-population algorithms mimicking social insect behaviors to advanced human-machine hybrid systems.These systems harness collective intelligence from both biological and artificial entities, enabling synergy between human intuition and machine computation.

"BERT Sentiment Fusion"[24] involves fusing various BERT model variants to enhance sentiment classification accuracy. The working mechanism amalgamates pre-trained BERT models with task-specific fine-tuning techniques, leveraging the strengths of different BERT architectures. "Aspect Deep Onto Analyzer"[25] involves leveraging deep neural networks to capture intricate patterns in textual data while incorporating ontological knowledge to contextualize aspect-based sentiment analysis. "Financial News LSTM Analyst (FNLA)"[26] harnessing the sequential nature of financial news data and the memory retention capabilities of LSTM networks to capture nuanced sentiment dynamics. The working mechanism involves preprocessing financial news articles and feeding them into LSTM layers, which learn to extract temporal dependencies and sentiment patterns [28]-[47].

"IntelliServe Emotion Sentinel (IES)"[27] developing a multi-task ensemble framework that simultaneously predicts both emotions and sentiments within customer conversations. The

working mechanism involves leveraging ensemble learning techniques to combine the outputs of multiple models trained for emotion and sentiment analysis tasks. By jointly analyzing emotions and sentiments, the framework enhances the understanding of customer needs and sentiments, enabling more personalized and empathetic responses.

3. AMAMI RABBIT OPTIMIZATION-BASED LOGISTIC REGRESSION (ARO-LOGISENT)

3.1. Logistic Regression model:

A popular statistical approach in sentiment analysis, which entails deducing the underlying emotional tone of a document, is logistic regression. Determining if a piece of writing conveys neutral, positive, or negative feelings is the goal of sentiment analysis. Logistic Regression, despite its name, is not a regression technique but rather a classification algorithm utilized for binary classification tasks, such as sentiment analysis. Predicting the likelihood that an input falls into one of two categories, usually denoted as positive or negative emotion, is the crux of Logistic Regression. The logistic function also called the sigmoid function, is used in logistic regression to represent the likelihood of a binary outcome, as opposed to linear regression, which predicts continuous values.

For probability modeling, this function converts all real-valued inputs to integers between zero and one. In sentiment analysis, the input features usually consist of various linguistic characteristics extracted from the text, such as word frequencies, n-grams (sequences of adjacent words), or sentiment lexicons. To gauge how likely it is that the text has a good or negative tone, these qualities act as predictors.

The Logistic Regression model learns from a labeled dataset during the training phase. Its goal is to reduce the discrepancy between the actual

$$Data = \{x_1, x_2, \dots, x_N\} \quad (1)$$

class labels and the anticipated probabilities, therefore it tweaks its parameters accordingly. This process involves optimizing a cost function, typically the cross-entropy loss function, through techniques like gradient descent. Using the likelihood that each text corresponds to either a positive or negative sentiment class, the Logistic Regression model may categorize fresh texts once it has been trained. A class label is applied to the text by the model if the probability is greater than a predetermined threshold, which is typically 0.5.

Despite its simplicity, Logistic Regression offers several advantages for sentiment analysis tasks. It is computationally efficient and relatively easy to interpret compared to more complex models like neural networks. Additionally, Logistic Regression can handle high-dimensional feature spaces effectively, making it suitable for processing text data with numerous features. Logistic Regression also has limitations. It assumes that the relationship between the input features and the output follows a linear decision boundary, which may not always hold for complex data distributions. Moreover, Logistic Regression is inherently limited to binary classification tasks and may struggle with multi-class sentiment analysis scenarios.

To mitigate these limitations, practitioners often employ techniques like feature engineering to extract meaningful features from the text, or ensemble methods to combine multiple classifiers for improved performance. Moreover, incorporating domain-specific knowledge and leveraging larger datasets can enhance the model's effectiveness in capturing nuanced sentiment expressions. The steps involved in the process of sentiment analysis using Logistic Regression are as follows.

3.1.1. Data Collection:

Data Collection, the cornerstone of the entire sentiment analysis pipeline, lays the foundation for robust model development and accurate sentiment classification. Data Collection, in essence, involves scouring various sources to amass a diverse and representative dataset encompassing text samples spanning a spectrum of sentiments. These sources may include social media platforms, product reviews, news articles, or any text-based content pertinent to the analysis domain. The collected data serves as the substrate upon which the subsequent stages of preprocessing, feature extraction, and model training will unfold.

In Eq.(1), where N denotes the total number of text samples gathered from different sources. Each xi represents a distinct text sample comprising words, phrases, or sentences expressing sentiments.

The scale and diversity of the dataset play a pivotal role in shaping the efficacy of the

sentiment analysis model. A larger dataset facilitates better generalization and enhances the model's ability to capture the intricacies of diverse linguistic expressions. Conversely, a narrow or biased dataset may lead to suboptimal model performance and skewed sentiment predictions.

The quality and authenticity of the collected data hold paramount importance. Inaccurate or misleading data can introduce noise and bias, undermining the integrity of the sentiment analysis process. Hence, meticulous attention must be paid to ensure the credibility and relevance of the collected dataset. Expressed mathematically in Eq.(2), the quality of the collected data ($QData$)can be quantified using metrics such as:

$$Q_{Data} = \frac{Number\ of\ relevant\ Samples}{Total\ number\ of\ samples} \times 100\% \quad (2)$$

where the number of relevant samples represents the subset of data pertinent to the sentiment analysis task, and the total number of samples encompasses the entire dataset.

Data Collection transcends mere aggregation; it necessitates annotation or labelling of the text samples with sentiment labels (e.g., positive, negative, or neutral). Annotation imbues the dataset with ground truth labels essential for supervised learning, enabling the model to learn the underlying patterns between text features and sentiment categories, the annotation processcan be denoted as:

$$Data = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \quad (3)$$

In Eq.(3), where y_i represents the sentiment label associated with the corresponding text sample x_i.

3.1.2. Data Preprocessing:

Data Preprocessing assumes the mantle of refining and reshaping the raw textual input into a standardized format conducive to subsequent analysis. This crucial step requires a series of operations aimed at mitigating noise, standardizing text representations, and preparing the data for feature extraction and model training.

Text Cleaning: At the outset of Data Preprocessing, the focus converges on cleansing the raw text data of extraneous elements and artefacts that might obfuscate the underlying sentiment signals. Text Cleaning encompasses a repertoire of operations, including the removal of special characters, punctuation marks, numerical digits, and other non-alphabetic symbols that contribute little to the sentiment analysis task. Additionally, techniques such as lowercasing all text characters and stripping leading or trailing whitespace serve to

$$x_{clean} = clean(x) \quad (4)$$

standardize the text representations and alleviate spurious variations. In Eq.(4) the process of text cleaning can be encapsulated as:
where x denotes the raw text sample, and $xclean$ represents the cleaned version of the text devoid of extraneous elements.

Tokenization: After text cleaning, the text data undergoes tokenization, a pivotal operation that involves segmenting the continuous strings of text into discrete units of meaning, typically words or phrases. Tokenization facilitates granular analysis and feature extraction by breaking down the text into its constituent elements. Common tokenization strategies include whitespace tokenization, which segments text based on whitespace characters, and word-level tokenization, which splits text into individual words or tokens. Mathematically tokenization can be formalized as shown in Eq.(5).

$$x_{tokens} = tokenize(x_{clean}) \quad (5)$$

where x_{tokens} represents the tokenized version of the cleaned text x_{clean}.

Stopword Removal: Stopwords, ubiquitous in natural language text, comprise commonwords such as 'the,' 'and,' 'is,' which convey little semantic meaning and often introduce noise into the sentiment analysis process. To mitigate their impact, Data Preprocessing entails the removal of stopwords from the tokenized text data. This operation streamlines the text representations, focusing the analysis on content-bearing words and phrases that carry significant sentiment cues. Stopword removal can be mathematically expressed shown in Eq.(6).

$$x_{filtered} = remove\_stopwords(x_{tokens}) \quad (6)$$

where $x_{filtered}$ represents the tokenized text data with stop words removed.

Lemmatization or Stemming: To further normalize the text representations and reduce lexical variations, Data Preprocessing often incorporates lemmatization or stemming techniques. Lemmatization involves reducing words to their base or dictionary form (lemmas), whereas stemming involves truncating words to their root form by removing affixes. These techniques ensure that different morphological variants of words are treated as a single entity, enhancing the consistency and interpretability of the text data.

$$x_{normalized} = lemmatize(x_{filtered}) \quad (7)$$

In Eq.(7), where $x_{normalized}$ represents the normalized version of the filtered text data. Data Preprocessing lays the groundwork for subsequent analytical endeavours by refining the raw textual input into a standardized and sanitized format conducive to feature extraction and model training. Through a series of operations encompassing text cleaning, tokenization, stopword removal, and lemmatization or stemming,

3.1.3. Feature Extraction:

Feature Extraction assumes the mantle of transforming the processed text into a structured representation amenable to computational analysis. This pivotal step involves distilling the salient characteristics or features from the text data, which serve as input variables for subsequent model training and sentiment classification.

Bag-of-Words Representation: A cornerstone of Feature Extraction in sentiment analysis is the Bag-of-Words (BoW) representation, which encapsulates the frequency or occurrence of individual words across the entire corpus of text data. Each text sample is transformed into a high-dimensional vector, where each dimension corresponds to a unique word in the vocabulary, and the value of each dimension represents the frequency of the corresponding word in the text sample. The BoW representation captures the lexical richness and distributional patterns of words within the text, enabling the sentiment analysis model to discern meaningful associations between word frequencies and sentiment categories. This BoW representation can be formalized shown in Eq.(8) in mathematical form.

$$(x) = [f_1, f_2, \ldots, f_V] \quad (8)$$

where $BoW(x)$ denotes the Bag-of-Words representation of the text sample x, and f_i represents the frequency of the i^{th} word in the vocabulary V.

TF-IDF Representation: An extension of the BoW representation, the Term Frequency- Inverse Document Frequency (TF-IDF) scheme enhances the discriminatory power of individual words by weighing them based on their frequency within the text sample andacross the entire corpus. Under this scheme, words that are frequent within a particular text sample but rare across the corpus are assigned higher weights, whereas common words that occur frequently across the corpus are down-weighted. The TF-IDF representation captures the saliency of words within the context of individual text samples, enabling the sentiment analysis model to prioritize content-bearing terms with higher discriminatory power. TF-IDF representation can be expressed in Eq.(9).

.

$$TF - IDF(x) = [w_1, w_2, \ldots, w_V] \quad (9)$$

where $TF - IDF(x)$ denotes the TF-IDF representation of the text sample x, and w_i represents the TF-IDF weight of the i^{th} word in the vocabulary V.

Word Embeddings: Word embeddings have been a game-changer for sentiment analysis feature extraction in the past several years. Word Embeddings link words to dense, low- dimensional vectors in a continuous vector space, including semantic relationships and contextual information, in contrast to standard representations that regard words as discrete entities. Word embeddings model the syntactic and semantic features of real language, allowing words to be represented as proximal vectors in an embedding space based on their similarity in meaning or context. The model's capacity to detect semantic similarities and deduce sentiment from context is improved by this distributed representation. Word Embeddings can be mathematically represented as shown in Eq.(10).

$$WordEmbeddings(x) = [e_1, e_2, \ldots, e_d] \quad (10)$$

where $WordEmbeddings(x)$ denotes the Word Embeddings representation of the text sample x, and $e_i$represents the embedding vector of the i^{th} word in the text sample, with d denoting the dimensionality of the embedding space.

3.1.4. Splitting the Dataset:

Splitting the Dataset is crucial for assessing the generalization performance of the sentiment analysis model and guarding against overfitting or underfitting

Training-Testing Split: The splitting of the dataset into a training set and a testing set is the fundamental process of splitting the dataset. The training set, comprising the majority of the data, serves as the substrate for model training, enabling the sentiment analysis model to learnthe underlying patterns and associations between text features and sentiment categories. The testing set, on the other hand, remains isolated during the training phase and is reserved for evaluating the model's performance on unseen data. This partitioning scheme ensures that the model's effectiveness is assessed on data it has not been exposed to during training, thereby providing a robust estimate of its generalization performance. The training-testing split can be formalized and mathematically represented using.

$$TrainingSet, TestingSet = SplitDataset(Data, train\_ratio) \quad (11)$$

where $TestingSet$ denotes the training subset of the dataset, $TestingSet$represents the testing subset, and $train\_ratio$ signifies the proportion of data allocated to the training set.

Cross-Validation: Cross-Validation emerges as a complementary technique for assessing model performance and tuning hyperparameters. Cross-validation involves iteratively partitioning the dataset into multiple subsets, or folds, and using each fold alternately as the testing set while the remaining folds serve as the training set. This iterative process yields multiple estimates of the model's performance, enabling more robust evaluations and mitigating the variability introduced by a single training-testing split. This Cross-Validation can be mathematically expressed in Eq.(12).

$$CV_{Scores} = CrossValidate(Model, Data, num\_folds) \quad (12)$$

where CV_Scoresdenotes the performance scores obtained through cross-validation, $Model$ represents the sentiment analysis model under evaluation, and num_foldsdenotes the number of folds used in the cross-validation procedure.

Validation Set: When training a model, it is sometimes helpful to use a third subset called the validation set to evaluate its performance and make adjustments to its hyperparameters. You may tweak the model settings on the validation set, which is separate from the training and testing sets so that the testing set stays clean. Model refinement and hyperparameter selection may be guided by regular evaluations of the model's performance on the validation set. This, in turn, improves the model's generalization performance.
Mathematically, the validation set can be denoted as:

$$ValidationSet = SplitValidation(TrainingSet, val\_ratio) \quad (13)$$

where $ValidationSet$ represents the validation subset of the training set, and val_ratio signifies the proportion of data allocated to the validation set.
Splitting the Dataset serves as a pivotal preparatory step in the sentiment analysis pipeline, enabling the assessment of model performance and generalization on unseen data.

3.1.5. Model Training:

After the dataset split into training and testing subsets, the focus now shifts towards training the sentiment analysis model on the training data to learn the underlying patterns and associations between text features and sentiment categories.

Model Selection: Choosing the right sentiment analysis model is crucial when starting the Model Training process. The available computational resources, the size and type of the dataset, and the complexity of the sentiment analysis task determine which model is considered. Models range from deep learning architectures like Transformers and Recurrent Neural Networks (RNNs) to more traditional machine learning algorithms like Logistic Regression and Support Vector Machines (SVMs). The chosen model should strike a balance between complexity and interpretability, ensuring robust performance while remaining tractable for training and deployment.

$$Model = SelectModel(Task, Data) \quad (14)$$

In Eq.(14), where $Model$denotes the selected sentiment analysis model, $Task$signifies the nature of the sentiment analysis task (e.g., binary classification, multi-class classification), and $Data$represents the training data used for model training.

Loss Function Optimization: After deciding on a model, the next step is to train it so that its parameters minimize a loss function. In classification tasks, this is usually the cross-entropy loss. Stochastic Gradient Descent (SGD) and Adam are two examples of gradient-based optimization algorithms that iteratively modify the model parameters according to the gradients of the loss functions about those parameters. This iterative process continues until convergence, where the model parameters converge to values that minimize the loss function and maximize predictive performance on the training data represented mathematically in Eq.(15).

$$Minimize = L(\theta) \quad (15)$$

where $L(\theta)$ denotes the loss function parameterized by the model parameters θ, and the objective is to minimize the loss function through parameter updates.

Hyperparameter Tuning: While training a model, it is common practice to optimize its hyperparameters, which control the model's behaviour and performance, in addition to its parameters. Learning rate, regularization strength, batch size, and parameters related to the network design are examples of hyperparameters in deep learning models. The goal ofhyperparameter tuning is to minimise overfitting and optimize the model's performance on the validation set by adjusting the model's parameters. To efficiently explore the hyperparameter space, one can use techniques like grid search, random search, or Bayesian optimization. In Eq.(16), hyperparameter tuning can be represented mathematically.

$$OptimalHyperparameters = TuneHyperparameters(Model, TrainingSet, ValidationSet) \quad (16)$$

where $OptimalHyperparameters$ denotes the optimal configuration of hyperparameters, and $TrainingSet$ and $ValidationSet$ represent the training and validation subsets of the dataset, respectively.

3.1.6. Model Evaluation:

Performance Metrics: Performance metrics, which quantify the model's efficacy in sentiment classification, are the backbone of model evaluation. Many binary sentiment analysis tasks use common performance indicators such as F1-score, recall, accuracy, precision, and area under the receiver operating characteristic curve (ROC-AUC). The model's accuracy in positive and negative sentiment classification, detection of real positives and negatives, and reduction of false positives and negatives are all uncovered by these measures. Mathematically, performance metrics can be formalized and depicted in Eq.(17) – Eq.(21).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (17)$$

$$Precision = \frac{TP}{TP + FP} \quad (18)$$

$$Recall = \frac{TP}{TP + FN} \quad (19)$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (20)$$

$$ROC - AUC = \int_0^1 TPR(fpr)dfpr \quad (21)$$

where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively, and TPR represents the true positive rate.

Confusion Matrix: The confusion matrix is an additional tool for Model Evaluation. It shows the model's predictions compared to the ground truth labels in a tabular format. For a complete picture of how well the model classified various sentiment categories, the confusion matrix summed together the counts of true positives, true negatives, false positives, and false negatives. To gain a better understanding of the model's performance, one may derive additional metrics like recall, accuracy, and precision from the confusion matrix. The confusion matrix can be denoted as Eq.(22).

$$ConfusionMatrix = \begin{pmatrix} TP & FP \\ FN & TN \end{pmatrix} \quad (22)$$

Cross-Validation Results: It is common practice in Model Evaluation to aggregate findings from cross-validation to gain more robust estimates of performance, in addition to evaluating the model on a single training-testing split. To further evaluate the model's generalizability and variance over distinct data subsets, cross-validation produces performance metrics across many dataset folds. Practitioners can get more accurate estimations of the model's actual performance by averaging performance measures from several folds. These cross-validation results can be represented as Eq.(23).

$$CV_{Performance} = [Metric_1, Metric_2, \dots, Metric_n] \quad (23)$$

where $Metric_i$ represents the performance metric obtained from the i^{th} fold of cross-validation.

3.1.7. Hyperparameter Tuning:

Grid Search: Grid search is an essential tool for hyperparameter tuning; it entails scanning a predetermined space of hyperparameters to find the best possible setting that makes the model work as well as possible. Grid search involves specifying a list of potential values for each hyperparameter and then training and evaluating the model for every conceivable combination of those values. To find the hyperparameter setting that performs best on the validation set, practitioners can use grid search, which methodically explores the whole search space. Mathematically, grid search can be represented as Eq. (24).

$$OptimalHyperparameters = GridSearch(Model, HyperparameterSpace, \quad (24)$$

Where $OptimalHyperparameters$ denotes the optimal configuration of hyperparameters, $HyperparameterSpace$ represents the predefined search space for hyperparameters, and $ValidationSet$ signifies the validation subset of the dataset.

Random Search: It explores the entire hyperparameter space systematically, random search samples hyperparameters from predefined distributions and evaluates the model's performance with randomly selected configurations. Random search offers advantages in scenarios where the hyperparameter space is high-dimensional or where certain hyperparameters have a greater impact on model performance than others.

$$Predictions = Model(x_{preprocessed}) \quad (27)$$

By randomly sampling hyperparameters, random search efficiently explores the hyperparameter space while providing comparable performance to grid search. This random search can be expressed mathematically depicted in Eq.(25)

$$OptimalHyperparameters = RandomSearch\begin{pmatrix} Model, HyperparameterDistribution, \\ ValidationSet \end{pmatrix} \quad (25)$$

Where $HyperparameterDistribution$ represents the distribution from which hyperparameters are sampled.

Bayesian Optimization: Iteratively searching for ideal hyperparameters is guided by Bayesian optimization, another sophisticated strategy for hyperparameter tuning. This method uses probabilistic models to describe the underlying goal function, such as model performance. By adaptively selecting the next set of hyperparameters to assess based on knowledge gathered from prior evaluations, Bayesian optimization focuses the search on attractive parts of the hyperparameter space. By efficiently exploring the hyperparameter space while leveraging probabilistic modeling, Bayesian optimization offers advantages in terms of sample efficiency and convergence speed. This Bayesian optimization can be denoted as Eq.(26).

$$OptimalHyperparameters = BayesianOptimization\begin{pmatrix} Model, Objective \\ Hyperparamet \end{pmatrix} \quad (26$$

where $ObjectiveFunction$ represents the function to be optimized (e.g., model performance).

3.1.8. Predictions:

Forward Pass: At the core of Predictions lies the forward pass, where new text samples are fed into the trained sentiment analysis model to obtain predictions about their sentiment. During the forward pass, the input text undergoes the same preprocessing steps applied during training, including tokenization, stopword removal, and lemmatization or stemming. The preprocessed text is then transformed into a structured representation (e.g., Bag-of- Words, TF-IDF, Word Embeddings) compatible with the input format expected by the model. Finally, the model computes the predicted sentiment label based on the learned associations between text features and sentiment categories. The forward pass can be represented in Eq.(27).
where $Predictions$ represents the predicted sentiment labels for the input text $x_{preprocessed}$, obtained through the trained sentiment analysis model.

Probability Estimation: In addition to predicting the sentiment label for each text sample, the sentiment analysis model may also output probabilities or confidence scores indicating the likelihood of each sentiment category. Probability estimation provides a more nuanced understanding of the model's confidence in its predictions,

enabling practitioners to assess the uncertainty associated with each prediction. Higher probabilities for a particular sentiment category signify greater confidence in the model's prediction, while lower probabilities indicate higher uncertainty. Probability estimation can be depicted mathematically in Eq(28).

$$Probabilities = model\_Probability(x_{preprocessed}) \quad (28)$$

where $Probabilities$ represents the predicted probabilities for each sentiment category, obtained through the trained sentiment analysis model.

Post-processing: Post-processing may involve sentiment aggregation, where predictions for multiple text samples are aggregated to derive an overall sentiment score or sentiment distribution. Sentiment analysis results may be subjected to further analysis or visualization to uncover underlying trends, patterns, or anomalies in the sentiment of the text data.

$$SentimentAggregation = AggregatePredictions(Predictions) \quad (29)$$

In Eq.(29), where $SentimentAggregation$ represents the aggregated sentiment score or distribution derived from the model's predictions.

3.1.9. Post-processing:

Sentiment Aggregation: Sentiment aggregation enables practitioners to obtain a high-level summary of the sentiment dynamics within the dataset, facilitating comparative analysis and trend identification. Common aggregation techniques include averaging sentiment scores, computing sentiment distributions, or aggregating sentiment labels based on predefined rules or thresholds is depicted in Eq.(30).

$$SentimentAggregation = AggregatePredictions(Predictions) \quad (30)$$

where $SentimentAggregation$ represents the aggregated sentiment score or distribution derived from the model's predictions.

Sentiment Analysis Visualization: In addition to numerical summaries, Post-Processing may involve visualizing sentiment analysis results through various graphical representations. Sentiment analysis visualizations provide intuitive insights into the sentiment dynamics within the dataset, enabling practitioners to identify patterns, outliers, or correlations visually. Common visualization techniques include bar charts, line plots, pie charts, and heatmaps, which depict sentiment distributions, trends over time, or sentiment correlations with other variables. Mathematically, sentiment analysis visualization can be represented as Eq.(31):

$$Visualization=Visualize(x, SentimentAggregation) \quad (31)$$

where $Visualization$ represents the graphical representation of sentiment analysis results derived from the input text data x and the aggregated sentiment scores or distributions

Sentiment Analysis Reporting: Furthermore, Post-Processing may involve generating comprehensive reports summarizing the sentiment analysis results and insights derived from the text data. Sentiment analysis reports provide stakeholders with a detailed overview of sentiment trends, key findings, and actionable recommendations, facilitating informed decision-making and strategic planning. Reports may include descriptive statistics, sentiment distributions, sentiment trends over time, sentiment correlations with other variables, and qualitative insights derived from text analysis.

$$Report = GenerateReport(x, SentimentAggregation,Visualization) \quad (32)$$

In Eq.(32), where $Report$ represents the comprehensive report summarizing sentiment analysis results and insights derived from the input text data x, aggregated sentiment scores or distributions, and sentiment analysis visualizations.

3.1.10. Deployment:

Model Integration: At the core of Deployment lies model integration, where the trained sentiment analysis model is seamlessly integrated into existing software or infrastructure. Model integration involves embedding the model within production systems, such as web applications, mobile apps, or data pipelines, to enable real-time sentiment analysis on incoming text data. Integration may require developing application programming interfaces (APIs), libraries, or software packages that expose the sentiment analysis functionality to downstream systems or

end-users. Mathematically, model integration can be denoted asEq.(33).

$$IntegratedModel=IntegrateModel(TrainedModel) \quad (33)$$

where $IntegratedModel$ represents the sentiment analysis model seamlessly integrated into production systems.

Scalability and Performance Optimization: Scalability measures may include deploying the model on distributed computing frameworks or cloud infrastructure to distribute computational load across multiple nodes. Performance optimization techniques, such asbatch processing, caching, or parallelization, may be employed to accelerate sentiment analysis inference and reduce latency in processing incoming text data.

$$OptimizedModel= OptimizePerformance (IntegratedModel) \quad (34)$$

In Eq.(34), where $OptimizedModel$ represents the sentiment analysis model optimized for scalability and performance.

Monitoring and Maintenance: Deployment involves establishing monitoring and maintenance mechanisms to ensure the ongoing reliability and performance of the deployed sentiment analysis model. Monitoring may involve tracking key performance indicators (KPIs), such as throughput, latency, and accuracy, to detect anomalies or deviations from expected behaviour. Maintenance activities may include periodic model retraining, updating, or reevaluation to adapt to evolving data distributions, concept drift, or changing business requirements represented mathematically in Eq.(35).

$$MonitoringResults = Monitor (OptimizedModel, IncomingData) \quad (35)$$

where $MonitoringResults$ represents the results of monitoring the deployed sentiment analysis model's performance on incoming data.

Functional Procedure 1: Logistic Regression

```
procedure LogisticRegression(X, y,
learning_rate, num_iterations):initialize
weights w and bias b to zeros or small
random values normalize or standardize
the input features X (optional)
for i = 1 to num_iterations
    do: compute z =
    dot_product(w, X) + b
    compute predicted probabilities y_hat
    = sigmoid(z)

    compute binary_cross_entropy_loss =

       -sum(y *
    log(y_hat) + (1 - y) * log(1 - y_hat))
    /num_examples
    compute dw = dot_product(X, (y_hat -
    y)) / num_examplescompute db =
    sum(y_hat - y) / num_examples
    update weights: w = w - learning_rate *
    dwupdate bias: b = b - learning_rate *
    db
return w, b

function sigmoid(z):
return 1 / (1 + exp(-z))

procedure Predict(X, w, b):
compute z = dot_product(w, X) + b
compute predicted probabilities y_hat =
sigmoid(z) return y_hat

procedure Evaluate(y_true, y_pred):
# Evaluate performance using appropriate
metrics such as accuracy, precision, recall,
or F1-score
```

Algorithm 1 depicts the steps involved in training a Logistic Regression model, making predictions, and evaluating its performance. The LogisticRegression procedure trains the model on input features X and corresponding labels y using a specified learning rate and number of iterations. The Predict procedure predicts the labels for new data using the trained parameters w and b. Finally, the evaluation procedure evaluates the model's performance using appropriate metrics.

3.2. Amami Rabbit Optimization (ARO)

Amami Rabbit Optimization (ARO) is a metaheuristic optimization algorithm inspired by the unique characteristics and behaviours of the Amami rabbit (Pentalagusfurnessi), an endangered

species native to the Amami Islands in Japan. ARO leverages the natural foraging and mating behaviours of the Amami rabbit to iteratively explore the solution space and find optimal solutions to optimization problems

- **Initialization:** Start with a set of possible answers to the optimization issue, called a population of candidatesolutions. The initial population should reflect the diversity of the solution space and may be generated randomly or through other heuristic methods.
- **Foraging Behavior:** Emulate the foraging behaviour of Amami rabbits, which involves searching for food sources in their natural habitat. In ARO, candidate solutions explore the solution space by iteratively adjusting their positions based on the quality of nearby solutions.
- **Mating Behavior:** Model the mating behavior of Amami rabbits, which involves selecting mates based on desirable traits. In ARO, candidate solutions exchange information with other solutions through crossover and mutation operations to generate offspring with potentially improved fitness.
- **Social Interaction:** Capture the social interaction among Amami rabbits, which involves sharing information and learning from other individuals in the population. In ARO, candidate solutions communicate and exchange information to collectively improve the overall quality of solutions in the population.
- **Fitness Evaluation:** Evaluate the fitness of each candidate solution based on its performance in solving the optimization problem. The fitness function assesses how well a solution meets the objectives and constraints of the problem, guiding the selection of solutions for further exploration and reproduction.
- **Selection:** Select candidate solutions for reproduction based on their fitness values, favoring solutions with higher fitness for mating and producing offspring. Selection mechanisms such as roulette wheel selection or tournament selection may be employed to maintain diversity and prevent premature convergence.
- **Offspring Generation:** Generate offspring by applying crossover and mutation operations to selected parent solutions. Crossover combines information from two parent solutions to produce offspring with characteristics inherited from both parents, while mutation introduces random changes to promote the exploration of new regions in the solution space.
- **Replacement :** Replace inferior solutions in the population with newly generated offspring, ensuring that the population maintains its size and diversity over successive generations. Replacement strategies may prioritize solutions with higher fitness or employ elitism to preserve the best solutions encountered so far.
- **Convergence Check:** Check for convergence criteria to determine whether the optimization process should terminate. Convergence may be determined based on the stability of the population, the number of iterations, or the improvement in fitness over successive generations.
- **Termination:** Terminate the optimization process if convergence criteria are met or if a predefined maximum number of iterations is reached. The best solution encountered during the optimization process is returned as the final solution to the optimization problem.

3.2.1. Initialization:

Initialization of a population of candidate solutions, laying the foundation for subsequent exploration and optimization. In this step, a diverse set of potential solutions is generated to represent different points in the solution space. The initialization process aims to provide an initial pool of solutions that reflects the variability and complexity of the optimization problem being addressed depicted mathematically in Eq.(36).

$$Population = \{Solution_1, Solution_2, \dots, Solution_N\} \quad (36)$$

where $Population$ represents the initial population of candidate solutions, and $solution_i$denotes the i^{th} the solution in the population.

The size of the population N is a crucial parameter in ARO, as it determines the diversity and exploration capabilities of the optimization process. Although the computational complexity may rise, the solution space may be explored more thoroughly with a bigger population size.

The initialization process may involve generating candidate solutions randomly within predefined ranges or using heuristic methods tailored to the characteristics of the optimization problem. For example, solutions may be generated by randomly sampling from uniform or Gaussian distributions, or by employing techniques such as Latin hypercube sampling or quasi-random sequences to ensure thorough coverage of the solution space.

$$Solution_i = RandomInitialization(LowerBound, UpperBound) \quad (37)$$

Where $RandomInitialization$ generates a candidate solution within the specified lower and upper bounds of the solution space.

The initialization process may incorporate domain-specific knowledge or constraints to guide the generation of candidate solutions. For example, in optimization problems involving physical parameters or engineering design variables, solutions may be constrained to feasible regions of the solution space to ensure practicality and validity.

$$Solution_i = HeuristicInitialization(ProblemSpecificParameters) \quad (38)$$

where $HeuristicInitialization$ employs domain-specific heuristics or knowledge to generate candidate solutions tailored to the characteristics of the optimization problem.

3.2.2. Foraging Behavior:

During this Foraging Behaviour step, candidate solutions iteratively adjust their positions based on the quality of nearby solutions, akin to how Amami rabbits navigate their environment to find food sources. The foraging behaviour in ARO can be represented mathematically in Eq.(39).

$$Position_i(t+1) = Position_i(t) + \Delta Position_i \quad (39)$$

where $Position_i(t)$ represents the position of the i^{th}solution at time t, and $\Delta Position_i$ denotes the incremental adjustment or movement of the solution towards more promising regions of the solution space.

The incremental adjustment $\Delta Position_i$ is determined by factors such as the attractiveness of neighbouring solutions, the gradient of the fitness landscape, and stochastic exploration mechanisms. Solutions may move towards regions of higher fitness, explore uncharted areas of the solution space, or exploit promising regions identified during previous iterations.

$$\Delta Position_i = \alpha \times Attractiveness_i + \beta \times Exploration_i \quad (40)$$

In Eq.(40), where α and β represent scaling factors that control the relative influence of attractiveness and exploration, and $Attractiveness_i$ and $Exploration_i$represent the attractiveness and exploration components influencing the movement of the i^{th} solution.

The attractiveness component $Attractiveness_i$reflects the quality or fitness of neighbouring solutions relative to the current solution. Solutions are attracted towards neighbouring solutions with higher fitness values, promoting the exploitation of promising regions in the solution space.

$$Attractiveness_i = \sum_{j=1}^{N} \frac{f(Position_j)}{Distance_{ij}} \quad (41)$$

In Eq.(41), where $f(Position_j)$ represents the fitness of the neighbouring solution j, and $Distance_{ij}$ denotes the Euclidean distance between the i^{th}solution and the neighbouring solution j.

The exploration component $Exploration_i$ introduces stochasticity or randomness into the movement of solutions, facilitating the exploration of diverse regions of the solution space and preventing premature convergence.

$$Exploration_i = RandomExploration \quad (42)$$

In Eq.(42), where $RandomExploration$ represents a random perturbation or exploration mechanism that introduces stochasticity into the movement of solutions.

3.2.3. Mating Behavior:

In this phase, it focuses on emulating the mating behaviour of Amami rabbits, which involves selecting mates based on desirable traits. In the context of optimization, mating behaviour corresponds to the exchange of information between candidate solutions to produce offspring with potentially improved fitness. Mathematically,

the mating behaviour in ARO can be represented in Eq.(43):

$$Offspring_i = Crossover(Parent_1, Parent_2) \quad (43)$$

where $Offspring_i$ represents the offspring generated by crossing over the genetic material of two parent solutions $Parent_1$ and $Parent_2$. Crossover involves combining characteristics from both parents to produce offspring with potentially improved traits.

$$Offspring_i = Mutation(Offspring_i) \quad (44)$$

In Eq.(44), where $Offspring_i$undergoes mutation to introduce random variations or changes, promoting the exploration of novel regions in the solution space. Mutation helps prevent premature convergence and facilitates the discovery of diverse solutions.

The crossover and mutation operations aim to diversify the population of candidate solutions and explore new regions of the solution space. Crossover enables the exchange of genetic material between parent solutions, facilitating the combination of beneficial traits from both parents in the offspring represented mathematically in Eq.(45).

$$Crossover(Parent_1, Parent_2) = Offspring \quad (45)$$

where $Offspring$ represents the offspring generated by crossing over the genetic material of $Parent_1$ and $Parent_2$, incorporating characteristics from both parents.

Mutation introduces random changes or perturbations to the offspring's genetic material, promoting the exploration of novel solutions and preventing stagnation in the optimization process.

$$Mutation(Offspring) = MutatedOffspring \quad (46)$$

In Eq.(46), where $MutatedOffspring$ represents the offspring with randomly introduced variations or mutations, facilitating the exploration of new regions in the solution space.

3.2.4. Social Interaction:

Social interaction in ARO involves communication and exchange of information between solutions to collectively improve the overall quality of solutions in the population. Which is Mathematically represented in Eq.(47).

$$Neighbor_i = SelectNeighbor(Population_i) \quad (47)$$

where $Neighbor_i$ represents a neighbouring solution selected by the i^{th}*the* solution in the population, and $Population_i$denotes the subset of the population within the vicinity of the i^{th} solution. SelectNeighbor may employ mechanisms such as neighbourhood search or nearest-neighbour selection to identify neighbouring solutions.

$$Information_i = ExchangeInformation(Solution_i, Neighbor_i) \quad (48)$$

In Eq.(48), where $Information_i$represents the information exchanged between the i^{th}solution and its neighbouring solution. ExchangeInformation facilitates the sharing of knowledge, insights, or solutions between neighbouring solutions to promote collective learning and improvement.

$$Solution_i = UpdateSolution(Solution_i, Information_i) \quad (49)$$

In Eq.(49), where $UpdateSolution$ integrates the exchanged information into the i^{th} the solution, enabling it to adapt and improve based on insights gained from neighbouring solutions. Updating solutions based on exchanged information enhances the overall quality ofsolutions in the population and facilitates convergence towards better solutions.

$$Population = UpdatePopulation(Population, Information) \quad (50)$$

In Eq.(50), where $Population$ represents the updated population of candidate solutions after social interaction, and $Information$ encompasses the exchanged information between solutions. UpdatePopulation integrates the exchanged information into the entire population, collectively improving the overall quality of solutions and promoting convergence towards optimal solutions.

3.2.5. Fitness Evaluation:

Fitness evaluation serves as a crucial step in guiding the selection and reproduction of solutions based on their quality or fitness. In ARO, solutions are evaluated based on their ability to meet the objectives and constraints of the optimization problem, with higher fitness values indicating superior performance. The fitness evaluation process in ARO can be represented in Eq.(51).

$$Fitness_i = (Solution_i) \quad (51)$$

where $Fitnessi$ represents the fitness of the ith the solution in the population, and

($Solutioni$) denotes the fitness function that evaluates the performance of the solution in solving the optimization problem. The fitness function may be problem-specific and assess various criteria such as accuracy, efficiency, or cost-effectiveness.

$$Selection_i = (Population, Fitness) \quad (52)$$

In Eq.(52), where $Selectioni$ represents the selection probability of the ithsolution in the population, determined based on its fitness relative to other solutions in the population. Select employs selection mechanisms such as roulette wheel selection, tournament selection, or elitism to choose solutions for reproduction based on their fitness values depicted in Eq.(53).

$$Parents=SelectP(Population,Selection) \quad (53)$$

where $Parents$ represents the selected parent solutions chosen for reproduction, based on their selection probabilities determined by their fitness values. The purpose of Select Parents is to increase the likelihood of selecting solutions with a higher fitness level to reproduce.

$$Offspring = Rep(Parents) \quad (54)$$

In Eq.(54), where $Offspring$ represents the offspring generated by reproducing the selected parent solutions. Reproduction involves crossover and mutation operations, where characteristics from parent solutions are combined to produce offspring with potentially improved fitness.

3.2.6. Selection:

The selection of candidate solutions for reproduction is based on their fitness values. Selection is a critical step in guiding the evolutionary process of ARO, determining which solutions will contribute their genetic material to the next generation. By favoring solutions with higher fitness values, selection promotes the propagation of desirable traits and facilitates the convergence towards optimal solutions. Mathematically, the selection process in ARO can be represented in Eq.(55).

$$Selection_i = \frac{Fitness_i}{\sum_{j=1}^{N} Fitness_j} \quad (55)$$

where $Selectioni$ represents the selection probability of the ith the solution in the population, and $Fitnessi$denotes the fitness of the ith solution. The selection probability is computed as the ratio of the fitness of the solution to the sum of fitness values across all solutions in the population.

$$CumulativeProbability_i = \sum_{k=1}^{i} Selection_k \quad (56)$$

In Eq.(56), where $CumulativeProbabilityi$ represents the cumulative selection probability up to the ith solution. Cumulative probabilities are computed to facilitate the selection of solutions using techniques such as roulette wheel selection or stochastic universal sampling.

$$SelectedSolutions = RouletteWheeSelection(Population, CumulativeProbability) \quad (57)$$

In Eq.(57), where $SelectedSolutions$ represents the candidate solutions selected for reproduction using roulette wheel selection. Roulette wheel selection assigns a selection probability to each solution based on its fitness and selects solutions probabilistically, favouring solutions with higher fitness values depicted in Eq.(58).

$$Offspring = Reproduce(Selectedsolutions) \quad (58)$$

where $Offspring$ represents the offspring generated by reproducing the selected candidate solutions. Reproduction involves crossover and mutation operations, where characteristics from selected parent solutions are combined to produce offspring with potentially improved fitness.

3.2.7. Offspring Generation:

Offspring generation is a vital step in the evolutionary process of ARO, as it facilitates the exploration of new solution space regions and the propagation of desirable traits from parent solutions to the next generation. By combining characteristics from selected parent solutions, offspring are created with the potential for improved fitness and diversity. Mathematically, the offspring generation process in ARO can be represented in Eq.(59).

$$Offspring_i = Crossover(Parent_1, Parent_2) \quad (59)$$

where $Offspringi$ represents the offspring generated by crossing over the genetic material of two parent solutions $Parent1$ and $Parent2$. Crossover involves combining characteristics from both parents to produce offspring with potentially improved traits.

$$MutatedOffspring_i = Mutation(Offspring_i) \quad (60)$$

In Eq.(60), where $MutatedOffspringi$ represents the offspring with randomly introduced variations or mutations. Mutation introduces random changes or perturbations to the offspring's genetic material,

promoting the exploration of novel solution space regions and preventing premature convergence.

$$Offspring = \{MutatedOffspring_1, utatedOffspring_2, \dots, MutatedOffspring_N\} \quad (61)$$

In Eq.(61), where $Offspring$ represents the population of offspring generated through crossover and mutation operations. The offspring population consists of newly created solutions with characteristics inherited from parent solutions, as well as random variations introduced through mutation.

$$NextGeneration = Combine(Popultaion, Offspring) \quad (62)$$

In Eq.(62), where $NextGeneration$ represents the next generation of solutions obtained by combining the parent solutions with the offspring population. Combining parent solutions with offspring ensures the continuation of the evolutionary process, with the offspring population contributing new genetic material to the population.

3.2.8. Replacement:

In ARO the process of replacement occurs, where inferior solutions in the population are replaced with newly generated offspring. By removing inferior solutions and introducing newly generated offspring, ARO ensures the continuous improvement and adaptation of the population towards better solutions. The replacement process in ARO can be represented in Eq.(63).

$$ReplacementCandidates = SelectReplacementCandidates(Population, Offspring) \quad (63)$$

In Eq.(63), where $ReplacementCandidates$ represents the candidate solutions selected for replacement based on predefined criteria. Select Replacement Candidates identify inferior solutions in the population that are to be replaced with newly generated offspring depicted in Eq.(64).

$$Population_{New} = Replace(Population, replacementCandidates, Offspring) \quad (64)$$

where $PopulationNew$ represents the updated population after replacement, where inferior solutions have been replaced with newly generated offspring. Replace combines the parent solutions with the offspring population, ensuring the continuity of the evolutionary process and the preservation of diversity within the population is represented mathematically in Eq.(65).

$$Population = Population_{New} \quad (65)$$

where $Population$ is updated to reflect the changes made during the replacement process. The updated population now consists of a mixture of parent solutions and newly generated offspring, maintaining the diversity and quality of solutions in the population.

$$BestSolution = electBestSolution(Population) \quad (66)$$

In Eq.(66), where $BestSolution$ represents the best solution encountered in the population, selected based on its fitness value. As a benchmark for measuring optimization success, Select Best Solution finds the solution in the updated population with the greatest fitness value.

3.2.9. Convergence Check:

The convergence of the optimization process is assessed to determine whether a satisfactory solution has been found or if further iterations are required. Convergence checking is essential for monitoring the progress of the optimization algorithm and ensuring that it terminates when the desired convergence criteria are met. By evaluating convergence, ARO can prevent unnecessary computational effort and efficiently allocate resources towards the most promising areas of the solution space. The convergence check in ARO can be represented in Eq.(67).

$$BestSolutionCurrent= SelectBestSo(Population) \quad (67)$$

where $BestSolutionCurrent$ represents the best solution encountered in the current population, selected based on its fitness value. SelectBestSolution identifies the solution with the highest fitness value from the population, serving as a reference point for evaluating convergence.

$$\Delta Fitness = FitnessPrevious - FitnessCurrent \quad (68)$$

In Eq.(68), where $\Delta Fitness$ represents the change in fitness between the previous and current iterations of the optimization process. A decrease in $\Delta Fitness$ indicates that the fitness of the best

solution has improved, suggesting progress towards convergence.

$$ConvergenceCriterion = |\Delta Fitness| < \quad (69)$$

In Eq.(69), where $ConvergenceCriterion$ represents the convergence criterion, which is satisfied when the change in fitness between iterations falls below a predefined threshold ϵ. The convergence criterion serves as a termination condition for the optimization algorithm, indicating that the optimization process has converged to a satisfactory solution represented mathematically in Eq.(70).

$$TerminateOptimization = Tru \quad (70)$$

where $TerminateOptimization$ is a boolean variable that is set to true when the convergence criterion is met, indicating that the optimization process can be terminated. Terminate Optimization serves as a flag to signal the algorithm to stop further iterations once convergence is achieved.

3.2.10. Termination:

The termination of the optimization process occurs when the convergence criteria have been met or when a predefined stopping condition is satisfied. Termination is a vital step in the ARO algorithm as it signifies the end of the optimization process and determines when to halt further iterations. By terminating the algorithm appropriately, ARO ensures computational efficiency and prevents unnecessary resource consumption. The termination of ARO can be represented mathematically in Eq.(71).

$$ConvergenceCriterion = |\Delta Fitness| < \epsilon \quad (71)$$

where $ConvergenceCriterion$ represents the convergence criterion, which is satisfied when the change in fitness between iterations falls below a predefined threshold ϵ. This criterion indicates that the optimization process has converged to a satisfactory solution, and further iterations are unnecessary.

$$TerminateOptimization = True \quad (72)$$

In Eq.(72), where $TerminateOptimization$ is a boolean variable that is set to true when the convergence criterion is met, indicating that the optimization process can be terminated. Terminate Optimization serves as a flag to signal the algorithm to stop further iterations once convergence is achieved depicted in Eq.(73).

$$StopCondition \quad (73)$$

where $StopCondition$ represents any additional stopping conditions defined by the user or specific to the optimization problem. These conditions could include reaching a maximum number of iterations, exceeding a computational budget, or meeting domain-specific requirements.

$$TerminateOptimization = True \quad (74)$$

In Eq.(74), where $TerminateOptimization$ is set to true when any of the stopping conditions are met, indicating that the optimization process should be terminated. By evaluating all stopping conditions, ARO ensures that the optimization process stops when anyof the predefined criteria are satisfied.

Functional Procedure 2: ARO

```
Procedure
AmamiRabbitOptimization(objective_function,
num_rabbits, num_iterations):
    initialize rabbit_population with num_rabbits
    rabbits randomly distributed in the search
    space
      for iteration = 1 to num_iterations do:
        for each rabbit in rabbit_population do:
              explore_phase(rabbit)
              evaluate_fitness(rabbit)
              update_local_best(rabbit)
              update_global_best(rabbit)
         move phase(rabbit_population)
       return global_best_solution
procedure explore_phase(rabbit):
    randomly select a neighbor solution within a
certain radius from the current position of the
rabbit
    evaluate_fitness of the neighbor solution
    if neighbor_solution is better than
        current_solution then:
         move rabbit to neighbor_solution
    else if neighbor_solution is worse and meets
    the acceptance criterion then:
        move rabbit to neighbor_solution with
        probability based on temperature or other
        criteria
procedure move_phase(rabbit_population):
    calculate the movement direction for each
rabbit based on its local best and global best
solutions
    update the position of each
    rabbit using the movement
    direction and step size
    apply boundary constraints
    if necessary
procedure evaluate_fitness(rabbit):
    compute the fitness value of the rabbit using
the objective  function
```

This procedure outlines the main phases of the Amami Rabbit Optimization algorithm, including exploration, movement, fitness evaluation, and updating of local and global best solutions. During each iteration, rabbits explore the search space, update their positions, and improve their solutions based on local and global information. Finally, the algorithm returns the global best solution found after the specified number of iterations.

3.3. Fusion of ARO with Logistic Regression (ARO-LogiSent) :

In recent years, sentiment analysis has gained significant attention in various domains such as marketing, social media monitoring, and customer feedback analysis. One approach to sentiment analysis involves combining the strengths of evolutionary algorithms, such as Amami Rabbit Optimization (ARO), with machine learning techniques like logistic regression. This fusion aims to leverage the optimization capabilities of ARO and the predictive power of logistic regression to develop a robust sentiment analysis model.

ARO-LogiSent integrates the ARO algorithm with logistic regression to perform sentiment analysis on textual data. The ARO component is responsible for optimizing the feature space and hyperparameters of the logistic regression model to maximize its performance in sentiment classification tasks.

Data Collection : Initially labelled textual data for sentiment analysis is collected from various sources, such as social media platforms, customer reviews, or product descriptions. The dataset consists of text samples labelled with sentiment classes.

- **Data Preprocessing:** The collected data undergoes preprocessing steps, including tokenization, lowercasing, stop word removal, and stemming or lemmatization. This step ensures that the text data is normalized and cleaned for further analysis.
- **Feature Extraction:** Feature extraction involves transforming the preprocessed text data into numerical feature vectors that can be used as input to the logistic regression model. Techniques such as bag-of- words, TF-IDF (Term Frequency-Inverse Document Frequency), or word embeddings may be employed to represent the textual data as feature vectors.
- **Splitting the Dataset:** The dataset is divided into training, validation, and testing sets to facilitate model training, validation, and evaluation. Typically, the majority of the data is allocated to the training set, with smaller portions reserved for validation and testing.
- **Model Training:** The logistic regression model is trained using the training dataset. ARO is employed to optimize the hyperparameters of the logistic regression model, such as the regularization parameter and feature selection.
- **Model Evaluation:** The trained logistic regression model is evaluated using the validation dataset to assess its performance in sentiment classification. ARO-LogiSent aims to maximize evaluation metrics such as accuracy, precision, recall, and F1 score through iterative optimization.
- **Hyperparameter Tuning:** ARO optimizes the hyperparameters of the logistic regression model based on the performance metrics obtained during model evaluation. This step aims to fine-tune the model's parameters to achieve better sentiment classification results.
- **Prediction:** Once the logistic regression model is trained and optimized, it is used to make predictions on unseen data, such as the testing dataset or new incoming text samples. The model assigns sentiment labels (e.g., positive, negative) to the input text based on its learned patterns.
- **Post-Processing:** Post-processing involves analyzing the model predictions and refining them if necessary. This step may include sentiment aggregation, sentiment scoring, or incorporating domain- specific rules to improve the accuracy of sentiment analysis results.
- **Deployment:** The trained and optimized logistic regression model is deployed into production environments, where it can be used to perform real-time sentiment analysis on incoming textual data. The deployed model enables businesses to gain insights into customer sentiments and make informed decisions accordingly.

ARO-LogiSent combines the optimization capabilities of ARO with the predictive power of logistic regression to develop a robust sentiment analysis model. By integrating evolutionary algorithms with machine learning techniques,

ARO-LogiSent offers an effective approach to sentiment analysis that can be applied across various domains.

Functional Procedure for the Fusion of ARO and Logistic Regression

```
// Define logistic regression function

def logistic_regression(X_train, y_train, X_test):

    // Train logistic regression modelmodel = LogisticRegression() model.fit(X_train, y_train)

    // Predict sentiment labels for test data
    y_pred = model.predict(X_test)
    return y_pred

// Define ARO optimization function

def amami_rabbit_optimization(objective_functio n, num_rabbits, num_iterations):

    // Initialize rabbit population randomly

    rebbit_population = np.random.rand(num_rabbits, num_features)

    for iteration in range(num_iterations): for rabbit in rabbit_population:

    // Explore phase explore_phase(rabbit)

    // Evaluate fitness

    fitness = objective_function(rabbit)

    // Update local best update_local_best(rabbit, fitness)

    // Update global best update_global_best(rabbit, fitness)

    // Move phase move_phase(rabbit_population)

    // Return global best solution return global_best_solution

    // Define explore phase def explore_phase(rabbit):

    // Define exploration strategy, e.g., random walk return explored_solution

    // Define move phase

    def move_phase(rabbit_population):

    // Define movement strategy, e.g., update positions based on local and global best return updated_population

    // Define objective function for sentiment analysis def objective_function(rabbit):

    // Extract features from rabbit solution

    X_train, y_train, X_test = extract_features(rabbit)

    // Perform logistic regression and evaluate performance y_pred = logistic_regression(X_train, y_train, X_test)

    // Compute performance metric, e.g., accuracy fitness = compute_fitness(y_test, y_pred) return fitness

    // Define the function to extract features from rabbit solution def extract_features(rabbit):

    // Implement feature extraction method return X_train, y_train, X_test

    // Define function to compute fitness def compute_fitness(y_true, y_pred):

    // Implement performance metrics, e.g., accuracy return fitness

    // Define function to update local best solution def update_local_best(rabbit, fitness):

    // Update local best solution if fitness improves return local_best_solution

    // Define function to update global best solution def update_global_best(rabbit, fitness):

    // Update global best solution if fitness improves return global_best_solution
```

This algorithm integrates ARO optimization with logistic regression for sentiment analysis. It defines functions for logistic regression, ARO optimization phases (explore and move), the objective function (for sentiment analysis), feature extraction, fitness computation, and updating of local and global best solutions. The ARO algorithm iteratively optimizes

the logistic regression model parameters to improve sentiment analysis performance.

3.3.1. Advantages of ARO-LogiSent:

Infusing Amami Rabbit Optimization (ARO) with Logistic Regression for Sentiment Analysis, known as ARO-LogiSent, presents several advantages:

- **Enhanced Accuracy:** ARO-LogiSent leverages the optimization capabilities of ARO to fine- tune the parameters of logistic regression, resulting in a sentiment analysis model with improved accuracy. By iteratively optimizing the model's parameters, ARO-LogiSent can effectively capture the nuances of sentiment in textual data, leading to more accurate predictions.
- **Efficient Feature Selection:** ARO-LogiSent employs ARO to select relevant features from the textual data, eliminating redundant or noisy features that may negatively impact the performance of the sentiment analysis model. This feature selection process enhances the efficiency of the logistic regression model by focusing on the most informative aspects of the input data.
- **Robustness to Noise:** ARO-LogiSent's integration of ARO helps mitigate the effects of noise in the data by optimizing the logistic regression model's parameters robustly. By iteratively refining the model's parameters, ARO-LogiSent can better handle noisy or ambiguous textual inputs, resulting in more reliable sentiment analysis outcomes.
- **Flexibility and Adaptability:** ARO-LogiSent offers flexibility in adapting to different domains and datasets. The ARO component allows the sentiment analysis model to adapt to the characteristics of the input data, making it suitable for various applications such as social media monitoring, customer feedback analysis, and product sentiment analysis.
- **Scalability:** ARO-LogiSent is scalable and can accommodate large datasets with ease. The parallel processing capabilities of ARO enable efficient optimization of the logistic regression model's parameters, even when dealing with massive amounts of textual data. This scalability makes ARO-LogiSent well-suited for deployment in real-world scenarios where large-scale sentiment analysis is required.

4. ABOUT DATASET

The Amazon Review Data (2018) about the Home and Kitchen category represents a significant subset of the broader Amazon review dataset. With a substantial volume of 6,898,955 reviews, this dataset offers a comprehensive glimpse into consumer sentiments and preferences regarding various products and services within the Home and Kitchen domain. Each review within this dataset serves as a valuable piece of feedback from consumers, expressing their opinions, experiences, and satisfaction levels with the products they have purchased and used. As such, the dataset encompasses a diverse range of products, including but not limited to kitchen appliances, home decor items, furniture, and cleaning supplies.

Analyzing this dataset provides insights into consumer preferences, product performance, and market trends within the Home and Kitchen category. Researchers, analysts, and businesses can leverage this data to understand consumer behavior, identify popular products, and make informed decisions regarding product development, marketing strategies, and inventory management.

The Amazon Review Data (2018) for Home and Kitchen constitutes a rich and valuable resource for sentiment analysis, market research, and business intelligence within the e- commerce domain. Its substantial size and granularity make it a valuable asset for studying consumer behavior and preferences in the context of home-related products and services.

Table 1. Field Description

Field Name	Description
Field Name	Description
Review ID	Unique identifier for each review
Product ID	Unique identifier for each product
Reviewer ID	Unique identifier for each reviewer
Review Text	Textual content of the review
Star Rating	Rating given by the reviewer (1 to 5 stars)
Review Date	Date when the review was posted

Helpful Votes	Number of helpful votes received on the review
Total Votes	Total number of votes received on the review
Verified Purchase	Indicates if the reviewer purchased the product from Amazon
Product Category	Category of the product
Product Subcategory	Subcategory of the product

5. PERFORMANCE METRICS

Sentiment analysis, also known as opinion mining, is the computational task of determining the sentiment expressed in a piece of text, whether it's positive, negative, or neutral. Evaluating the performance of sentiment analysis models is crucial for assessing their effectiveness in accurately classifying sentiments. Performance metrics provide quantitative measures of how well a model performs in sentiment classification tasks. Commonly used performance metrics include precision, recall, classification accuracy, F-measure, Fowlkes– Mallows Index, and Matthews Correlation Coefficient.

Before delving into the specific metrics, it's essential to understand the basic concepts behind them. In sentiment analysis, predictions are typically classified as either positive or negative. True positives (TP) are instances where the model correctly predicts positive sentiment, while true negatives (TN) are instances where the model correctly predicts negative sentiment. False positives (FP) occur when the model incorrectly predicts positive sentiment, and false negatives (FN) occur when the model incorrectly predicts negative sentiment.

- **Precision:** Precision measures the proportion of correctly predicted positive sentiments out of all instances classified as positive by the model. It indicates how often the model correctly identifies positive sentiments. Higher precision implies fewer false positives, which are instances incorrectly labelled as positive.
- **Recall:** Recall, also known as sensitivity, measures the proportion of correctly predicted positive sentiments out of all instances of true positive sentiments in the dataset. It gauges how effectively the model captures all positive sentiments present. Higher recall suggests fewer missed positive sentiments.
- **Classification Accuracy:** Classification accuracy quantifies the overall correctness of sentiment predictions made by the model. It represents the proportion of correctly classified instances, including both true positives and true negatives, among all instances in the dataset.
- **F-Measure:** F-measure, the harmonic mean of precision and recall, provides a balanced measure of model performance. It considers both false positives and false negatives, offering a comprehensive evaluation of model accuracy. A higher F- measure indicates better precision and recall balance.
- **Fowlkes–Mallows Index:** The Fowlkes–Mallows Index assesses the similarity between predicted positive instances and actual positive instances. It measures how well the model clusters predicted positive sentiments compared to true positive sentiments. Higher values indicate a stronger agreement between predicted and actual positive sentiments.
- **Matthews Correlation Coefficient (MCC):** MCC evaluates the quality of binary classifications, considering true and false positives and negatives. It ranges from -1 to 1, with higher values indicating better prediction quality. MCC accounts for imbalanced datasets and provides insights into the overall performance of the model.

These performance metrics offer valuable insights into the effectiveness of sentiment analysis models, allowing researchers and practitioners to assess model accuracy, identify strengths and weaknesses, and make informed decisions to improve model performance.

6. RESULTS AND DISCUSSION

6.1. Precision and Recall Analysis

Precision and recall analysis is crucial in evaluating the performance of sentiment analysis models. Precision measures the proportion of correctly predicted positive sentiments out of all instances classified as positive by the model, while recall measures the proportion of correctly predicted positive sentiments out of all instances of true positive sentiments in the dataset. the precision of the sentiment analysis models varies significantly. IES demonstrates the lowest precision at 58.949%, followed by FNLA at 69.909%, and ARO- LOGI leads with 81.701%. A higher precision indicates a lower rate of false positive predictions, meaning that ARO-LOGI exhibits the highest accuracy in classifying positive sentiments.

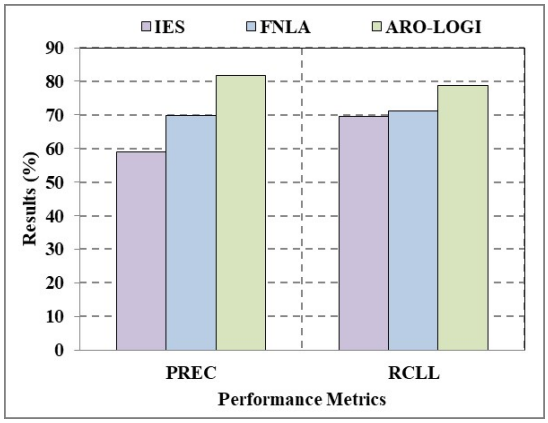

Figure 1. Precision And Recall

The recall values indicate the ability of the models to capture all positive sentiments present in the dataset. ARO-LOGI achieves the highest recall rate of 78.877%, followed by FNLA at 71.163%, and IES with the lowest recall rate of 69.510%. Figure 1. depicts the obtained results of Precesion and Recall. These findings suggest that ARO-LOGI excels in identifying positive sentiments comprehensively, making it a promising model for sentiment analysis tasks. Table 2 shows the average performance of ARO-LONGI with the comparison of IES and FNLA.

Table 2. Precision And Recall

Classification Algorithms	PREC	RCLL
IES	58.949	69.510
FNLA	69.909	71.163
ARO-LOGI	81.701	78.877

6.2. Classification Accuracy and F-Measure Analysis

Classification accuracy is a fundamental metric that measures the overall correctness of predictions made by a model. It quantifies the proportion of correctly classified instances, including both true positives and true negatives, among all instances in the dataset. F- measure, on the other hand, is the harmonic mean of precision and recall and provides a balanced measure of model performance. The obtained results of Classification Accuracy andF-Measure are depicted in Figure 2.

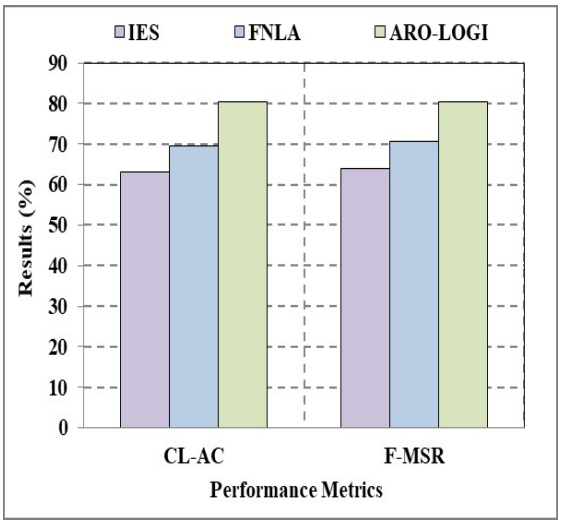

Figure 2. Classification Accuracy And F-Measure

In the provided data, the ARO-LOGI model exhibits the highest classification accuracy of 80.268%, followed by FNLA (69.397%) and IES (63.090%). This indicates that ARO-LOGI achieves the highest proportion of correctly classified instances compared to the other models. Similarly, when considering F-measure, ARO-LOGI again outperforms the other models with a score of 78.877%, followed by FNLA (71.163%) and IES (69.510%). These results suggest that ARO-LOGI demonstrates superior overall performance in accurately classifying sentiments compared to IES and FNLA shown in Table 3.

Table 3. Classification Accuracy And F-Measure

Classification Algorithms	CL-AC	F-MSR
IES	63.090	63.795
FNLA	69.397	70.531
ARO-LOGI	80.268	80.264

6.3. Fowlkes–Mallows Index and Matthews Correlation Coefficient Analysis.

Fowlkes–Mallows Index (FMI) and Matthews Correlation Coefficient (MCC) are two important metrics used to evaluate the performance of sentiment analysis models. ARO- LOGI demonstrates the highest FMI at 80.276%, followed by FNLA at 70.533%, and IES with the lowest FMI at 64.012%. A higher FMI indicates a stronger agreement between the predicted positive instances and the actual positive instances, highlighting the effectiveness of ARO-

LOGI in clustering positive sentiments as shown in Figure 3.

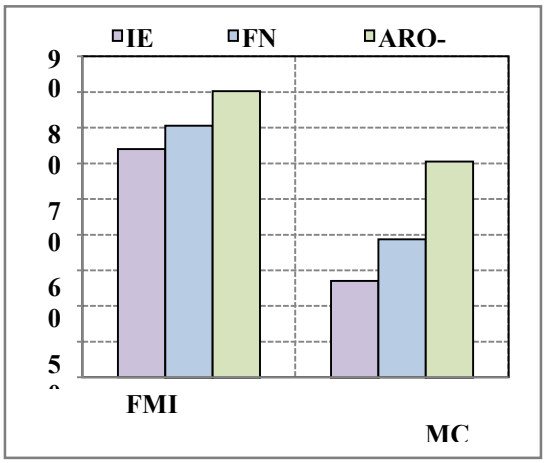

Figure 3. Fowlkes–Mallows Index And Matthews Correlation Coefficient

ARO-LOGI achieves the highest MCC score of 60.586%, indicating a strong correlation between the model's predictions and the actual sentiments. FNLA follows with an MCC of 38.715%, while IES has the lowest MCC at 27.045%. This suggests that ARO-LOGI provides the most accurate and reliable sentiment predictions among the three models, making it a promising choice for sentiment analysis tasks. The obtained average result is shown in Table 4.

Table 4. Fowlkes–Mallows Index And Matthews Correlation Coefficient

Classification Algorithms	FMI	MCC
IES	64.012	27.045
FNLA	70.533	38.715
ARO-LOGI	80.276	60.586

7. CONCLUSION

The evaluation of sentiment analysis models using various performance metrics provides valuable insights into their effectiveness and reliability. Precision and recall analysis reveals the accuracy and completeness of positive sentiment predictions, with ARO-LOGI exhibiting the highest precision and recall rates among the models considered. The Fowlkes– Mallows Index and Matthews Correlation Coefficient offer comprehensive assessments of model performance, with ARO-LOGI demonstrating superior agreement between predicted and actual positive instances and a strong correlation between predictions and actual sentiments. ARO-LOGI emerges as the most promising sentiment analysis model, outperforming IES and FNLA across multiple metrics. Its high precision, recall, FMI, and MCC scores signify its ability to accurately classify positive sentiments and provide reliable predictions. These findings underscore the effectiveness of integrating Amami Rabbit Optimization with logistic regression for sentiment analysis tasks, highlighting the potential of ARO-LOGI in real-world applications. Further research and experimentation could explore additional datasets and fine-tuning strategies to further enhance the performance of sentiment analysis models and optimize their utility in various domains.

REFERENCES

[1]. S. A., S. G., and K. G., "Enhanced Elman spike neural network based sentiment analysis of online product recommendation," Appl. Soft Comput., vol. 132, p. 109789, 2023, doi: https://doi.org/10.1016/j.asoc.2022.109789.

[2]. J. Bowden and R. Gemayel, "Sentiment and trading decisions in an ambiguous environment: A study on cryptocurrency traders," J. Int. Financ. Mark. Institutions Money, vol. 80, p. 101622, 2022, doi: https://doi.org/10.1016/j.intfin.2022.101622.

[3]. P. Liu, A. Hendalianpour, M. Feylizadeh, and W. Pedrycz, "Mathematical modeling of Vehicle Routing Problem in Omni-Channel retailing," Appl. Soft Comput., vol. 131, p. 109791, 2022, doi: https://doi.org/10.1016/j.asoc.2022.109791.

[4]. Y. Lin, P. Ji, X. Chen, and Z. He, "Lifelong Text-Audio Sentiment Analysis learning," Neural Networks, vol. 162, pp. 162–174, 2023, doi: https://doi.org/10.1016/j.neunet.2023.02.008.

[5]. D. Antypas, A. Preece, and J. Camacho-Collados, "Negativity spreads faster: A large-scale multilingual twitter analysis on the role of sentiment in political communication," Online Soc. Networks Media, vol. 33, p. 100242, 2023, doi: https://doi.org/10.1016/j.osnem.2023.100242.

[6]. R. Singh and R. Singh, "Applications of sentiment analysis and machine learning techniques in disease outbreak prediction – A review," Mater. Today Proc., vol. 81, pp. 1006–1011, 2023, doi: https://doi.org/10.1016/j.matpr.2021.04.356.

[7]. I. Priyadarshini et al., "Survivability of industrial internet of things using machine

learning and smart contracts," Comput. Electr. Eng., vol. 107, p. 108617, 2023, doi: https://doi.org/10.1016/j.compeleceng.2023.108617.

[8]. R. Strubytskyi and N. Shakhovska, "Method and models for sentiment analysis an hidden propaganda finding," Comput. Hum. Behav. Reports, vol. 12, p. 100328, 2023, doi: https://doi.org/10.1016/j.chbr.2023.100328.

[9]. Z. Li et al., "Multi-level correlation mining framework with self-supervised label generation for multimodal sentiment analysis," Inf. Fusion, vol. 99, p. 101891, 2023, doi: https://doi.org/10.1016/j.inffus.2023.101891.

[10]. M. Mhamed, R. Sutcliffe, X. Sun, J. Feng, and E. A. Retta, "Arabic sentiment analysis using GCL-based architectures and a customized regularization function," Eng. Sci. Technol. an Int. J., vol. 43, p. 101433, 2023, doi: https://doi.org/10.1016/j.jestch.2023.101433.

[11]. H. Liu et al., "Enhancing aspect-based sentiment analysis using a dual-gated graph convolutional network via contextual affective knowledge," Neurocomputing, vol. 553, p. 126526, 2023, doi: https://doi.org/10.1016/j.neucom.2023.126526.

[12]. A. Atak, "Exploring the sentiment in Borsa Istanbul with deep learning," Borsa Istanbul Rev., vol. 23, pp. S84–S95, 2023, doi: https://doi.org/10.1016/j.bir.2023.12.010.

[13]. F. K. Sufi, "Identifying the drivers of negative news with sentiment, entity and regression analysis," Int. J. Inf. Manag. Data Insights, vol. 2, no. 1, p. 100074, 2022, doi: https://doi.org/10.1016/j.jjimei.2022.100074.

[14]. T. H. Jaya Hidayat, Y. Ruldeviyani, A. R. Aditama, G. R. Madya, A. W. Nugraha, and M. W. Adisaputra, "Sentiment analysis of twitter data related to Rinca Island development using Doc2Vec and SVM and logistic regression as classifier," Procedia Comput. Sci., vol. 197, pp. 660–667, 2022, doi: https://doi.org/10.1016/j.procs.2021.12.187.

[15]. M. Wilksch and O. Abramova, "PyFin-sentiment: Towards a machine-learning-based model for deriving sentiment from financial tweets," Int. J. Inf. Manag. Data Insights, vol. 3, no. 1, p. 100171, 2023, doi: https://doi.org/10.1016/j.jjimei.2023.100171.

[16]. J. Maqbool, P. Aggarwal, R. Kaur, A. Mittal, and I. A. Ganaie, "Stock Prediction by Integrating Sentiment Scores of Financial News and MLP-Regressor: A Machine Learning Approach," Procedia Comput. Sci., vol. 218, pp. 1067–1078, 2023, doi: https://doi.org/10.1016/j.procs.2023.01.086.

[17]. M. Kasri, M. Birjali, M. Nabil, A. Beni-Hssane, A. El-Ansari, and M. El Fissaoui, "Refining Word Embeddings with Sentiment Information for Sentiment Analysis," J. ICT Stand., vol. 10, no. 3, pp. 353–382, 2022, doi: 10.13052/jicts2245-800X.1031.

[18]. P. Durga and D. Godavarthi, "Deep-Sentiment: An Effective Deep Sentiment Analysis Using a Decision-Based Recurrent Neural Network (D-RNN)," IEEE Access, vol. 11, pp. 108433–108447, 2023, doi: 10.1109/ACCESS.2023.3320738.

[19]. J. He, A. Wumaier, Z. Kadeer, W. Sun, X. Xin, and L. Zheng, "A Local and Global Context Focus Multilingual Learning Model for Aspect-Based Sentiment Analysis," IEEE Access, vol. 10, pp. 84135–84146, 2022, doi: 10.1109/ACCESS.2022.3197218.

[20]. M. Huang, H. Xie, Y. Rao, Y. Liu, L. K. M. Poon, and F. L. Wang, "Lexicon-Based Sentiment Convolutional Neural Networks for Online Review Analysis," IEEE Trans. Affect. Comput., vol. 13, no. 3, pp. 1337–1348, 2022, doi: 10.1109/TAFFC.2020.2997769.

[21]. H. He, G. Zhou, and S. Zhao, "Exploring E-Commerce Product Experience Based on Fusion Sentiment Analysis Method," IEEE Access, vol. 10, pp. 110248–110260, 2022, doi: 10.1109/ACCESS.2022.3214752.

[22]. H. Yan, B. Yi, H. Li, and D. Wu, "Sentiment knowledge-induced neural network for aspect-level sentiment analysis," Neural Comput. Appl., vol. 34, no. 24, pp. 22275–22286, 2022, doi: 10.1007/s00521-022-07698-0.

[23]. G.-Y. Wang, D.-D. Cheng, D.-Y. Xia, and H.-H. Jiang, "Swarm Intelligence Research: From Bio-inspired Single-population Swarm Intelligence to Human-machine Hybrid Swarm Intelligence," Mach. Intell. Res., vol. 20, no. 1, pp. 121–144, 2023, doi: 10.1007/s11633-022-1367-7.

[24]. A. S. Talaat, "Sentiment analysis classification system using hybrid BERT models," J. Big Data, vol. 10, no. 1, p. 110, 2023, doi: 10.1186/s40537-023-00781-w.

[25]. M. Belguith, C. Aloulou, and B. Gargouri, "Aspect Level Sentiment Analysis Based on

Deep Learning and Ontologies," SN Comput. Sci., vol. 5, no. 1, p. 58, 2023, doi: 10.1007/s42979-023-02362-3.
[26]. A. Sharaff, T. R. Chowdhury, and S. Bhandarkar, "LSTM based Sentiment Analysis of Financial News," SN Comput. Sci., vol. 4, no. 5, p. 584, 2023, doi: 10.1007/s42979- 023-02018-2.
[27]. D. Chen, H. Zhengwei, T. Yiting, M. Jintao, and R. Khanal, "Emotion and sentiment analysis for intelligent customer service conversation using a multi-task ensemble framework," Cluster Comput., 2023, doi: 10.1007/s10586-023-04073-z.
[28]. J. Ramkumar, A. Senthilkumar, M. Lingaraj, R. Karthikeyan, and L. Santhi, "Optimal Approach for Minimizing Delays in Iot-Based Quantum Wireless Sensor Networks Using Nm-Leach Routing Protocol," J. Theor. Appl. Inf. Technol., vol. 102, no. 3, pp. 1099–1111, 2024.
[29]. J. Ramkumar, R. Vadivel, B. Narasimhan, S. Boopalan, and B. Surendren, "Gallant Ant Colony Optimized Machine Learning Framework (GACO-MLF) for Quality of Service Enhancement in Internet of Things-Based Public Cloud Networking," J. M. R. S. Tavares, J. J. P. C. Rodrigues, D. Misra, and D. Bhattacherjee, Eds., Singapore: Springer Nature Singapore, 2024, pp. 425–438. doi: 10.1007/978-981-99-5435-3_30.
[30]. D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," Int. J. Comput. Networks Appl., vol. 10, no. 1, pp. 119–129, 2023, doi: 10.22247/ijcna/2023/218516.
[31]. R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," Int. J. Intell. Eng. Syst., vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.
[32]. J. Ramkumar, K. S. Jeen Marseline, and D. R. Medhunhashini, "Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks," Int. J. Comput. Networks Appl., vol. 10, no. 4, pp. 668–687, Aug. 2023, doi: 10.22247/ijcna/2023/223319.
[33]. J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," World J. Eng., vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.
[34]. M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," Int. J. Comput. Networks Appl., vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.
[35]. R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," Inc. Internet Things Healthc. Appl. Wearable Devices, pp. 109–121, 2019, doi: 10.4018/978-1-7998-1090-2.ch006.
[36]. J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," Int. J. Comput. Networks Appl., vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.
[37]. A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," Int. J. Comput. Networks Appl., vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.
[38]. J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," Int. J. Comput. Networks Appl., vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.
[39]. R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," Int. J. Comput. Digit. Syst., vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.
[40]. P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," 2022 Int. Conf. Adv. Comput. Technol. Appl. ICACTA 2022, pp. 1–5, Mar. 2022, doi: 10.1109/ICACTA54488.2022.9753203.
[41]. J. Ramkumar and R. Vadivel, CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks, vol. 556. 2017. doi: 10.1007/978-981-10-3874-7_14.
[42]. J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," in 2022 International Conference on Advanced Computing Technologies and Applications,

ICACTA 2022, 2022. doi: 10.1109/ICACTA54488.2022.9752899.

[43]. L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," ACM Int. Conf. Proceeding Ser., pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.

[44]. R. Jaganathan, V. Ramasamy, L. Mani, and N. Balakrishnan, "Diligence Eagle Optimization Protocol for Secure Routing (DEOPSR) in Cloud-Based Wireless Sensor Network," Res. Sq., 2022, doi: 10.21203/rs.3.rs-1759040/v1.

[45]. J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," Int. J. Comput. Networks Appl., vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[46]. J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in Lecture Notes in Electrical Engineering, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[47]. J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," Wirel. Pers. Commun., vol. 120, no. 2, pp. 887–909, Apr. 2021, doi: 10.1007/s11277-021-08495-z.