ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

A Comparative Analysis of Machine Learning Models for Domain Adaptation in Multiclass Sentiment Classification

Thawatwong Lawan[1], Jantima Polpinij[2], Chunpong Chan[3], Dolsun Singhakam[4], Theeraya Uttha[5], Anirut Chotthanom[6],Woraphot Watthusin[7], Bancha Luaphol[8] and Khanista Namee[9]

ABSTRACT

This study presents a comparative evaluation of machine learning models for domain adaptation in multiclass sentiment classification. While sentiment analysis aims to categorize opinions as positive, neutral, or negative, adapting models across domains remains a significant challenge due to differences in vocabulary, writing style, and sentiment expression. Models trained on a specific domain often fail to generalize effectively to others. To solve this problem, we evaluate how well six models—logistic regression, support vector machine (SVM) with a linear kernel, random forest, convolutional neural network (CNN), long short-term memory (LSTM), and BERT—perform on sentiment data from books, beauty & personal care, and automotive categories. The evaluation uses Amazon review data and measures performance via accuracy, F1 score, and Area Under the ROC Curve (AUC). Results indicate that BERT consistently outperforms all other models due to its attention-based transformer architecture, which captures nuanced contextual information across diverse domains. CNN and LSTM models also perform well, particularly in domain-specific settings, with CNN excelling in extracting local features and LSTM in modeling sequential relationships. Traditional models, such as logistic regression and SVM, show limitations in generalizability, while random forest demonstrates stable yet moderate performance. These findings highlight the strengths and trade-offs of each approach for effective cross-domain sentiment classification.

Article information:

Keywords: Multiclass Sentiment Classification, Machine Learning Models, Deep Learning Models, BERT

Article history:
Received: October 14, 2024
Revised: January 6, 2025
Accepted: April 17, 2025
Published: April 26, 2025 (Online)

DOI: 10.37936/ecti-cit.2025192.258824

1. INTRODUCTION

Sentiment classification offers several benefits [1-4]. For instance, it enables businesses to quickly gather feedback on their products and services. Detecting negative feedback highlights areas for improvement, while positive feedback emphasizes a company's strengths. Additionally, sentiment analysis tracks shifts in public opinion over time or in response to events like product launches or public relations issues. This analysis helps businesses address customer concerns more efficiently, enhancing customer satisfaction. Sentiment classification can also process large datasets, providing near real-time insights or automated actions based on public sentiment. Furthermore, it supports product development by identifying features that users frequently praise or criticize.

The lexicon-based approach is widely recognized as an effective technique for sentiment classification across various domains, to the best of our knowledge [1-10]. In this approach, the sentiment expressed in

[1,2,3,4,5,6]The authors are with the Faculty of Informatics, Mahasarakham University, Mahasarakham, Thailand, Email: thawatwong@msu.ac.th, jantima.p@msu.ac.th, 65011212003@msu.ac.th, 65011212011@msu.ac.th, theeraya.u@msu.ac.th and anirut@msu.ac.th
[7]The author is with the Division of Registrar, Mahasarakham University, Mahasarakham, Thailand, Email: woraphot.wat@msu.ac.th
[8]The author is with the Faculty of Administrative Science, Kalasin University, Kalasin, Thailand, Email: bancha.lu@ksu.ac.th
[9]The author is with the Faculty of Industrial Technology and Management, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand, Email: khanista.n@fitm.kmutnb.ac.th
[2]Corresponding author: jantima.p@msu.ac.th

text is determined using predefined sets of words, or lexicons, making it both comprehensible and interpreted. Furthermore, classification accuracy has improved with the integration of diverse machine learning algorithms alongside this lexicon-based approach. However, identifying the most suitable algorithm for a specific sentiment classification task remains complex. The challenge lies in the varying performance of different algorithms based on the dataset's nature and the specific problem context. We often conduct a comparative analysis of multiple machine learning models to address this issue. The most suitable model can be identified by evaluating and benchmarking the performance of various algorithms on the dataset in question. By considering factors such as precision, recall, accuracy, and computational efficiency, researchers and practitioners can make informed decisions about the algorithm that will yield the most reliable and actionable results for sentiment classification in their specific scenarios. Consequently, this systematic comparison provides a more data-driven approach to selecting the most suitable sentiment classifier for real-world applications.

Several traditional machine learning algorithms have been extensively evaluated for multiclass sentiment classification tasks [2-3, 11-14]. While these models often yield promising results, they face challenges in consistently performing well across various scenarios. One significant challenge is domain adaptation [15-16]. For example, a model trained on customer reviews for technology products may struggle when applied to beauty product reviews due to differences in vocabulary and sentiment expression. A term like "light," which describes the weight of a smartphone in a technology context, might instead refer to the subtlety of a color in beauty product reviews. This discrepancy highlights the difficulty models face in adapting to new domains with unique linguistic features and sentiment nuances that differ from their training data.

This study tackles the important issue of domain adaptability in multiclass sentiment classification, examining how effectively machine learning models generalize when trained on product reviews from one domain and tested on reviews from an entirely different domain. The primary goal is to create a robust, domain-agnostic text representation that can accurately identify sentiment across various contexts. To achieve this, we test five different models: three machine learning models (logistic regression, random forest, and support vector machine), two deep learning models (convolutional neural network and long short-term memory), and one transformer model (BERT) to see how well they work for sentiment analysis across different areas. Our experiments utilize a diverse dataset of Amazon reviews, and model performance is rigorously assessed using multiple metrics, including accuracy, F1 score, and Area Under the Curve (AUC). This research enhances our understanding of how machine learning models can be adapted for domain transferability in sentiment classification tasks, an increasingly important area for practical applications.

2. DATASETS AND DATA SEPARATION

2.1 Datasets

This study used four datasets, each detailed as follows. We developed a multiclass sentiment classification model using the first dataset. It comprises product reviews sourced from the Amazon website, specifically focusing on electronics and technology products. These reviews, written in English, use a five-star rating system to indicate the overall sentiment expressed by users. For sentiment classification in this research, reviews rated 5 stars were categorized as the positive class, those rated 3 stars as the neutral class, and those rated 1 star as the negative class. This classification scheme was chosen because a 5-star rating typically reflects a high level of satisfaction and a positive experience with the product, while a 1-star rating indicates significant dissatisfaction. A 3-star rating generally conveys ambivalence or a mixed opinion, often reflecting a balance of both positive and negative aspects, making it suitable as the neutral class.

To ensure the dataset's quality and reliability, a rigorous refinement process was undertaken. Language experts were consulted to correct grammatical and lexical errors and to eliminate reviews containing sarcasm, which could distort sentiment analysis outcomes. This meticulous curation was crucial for minimizing noise and enhancing the accuracy of subsequent analyses. Only reviews with a minimum length of 50 words were included to provide sufficient context for sentiment analysis, ensuring that the classification is based on comprehensive and meaningful information.

A structured data preparation pipeline was implemented to effectively visualize, analyze, and clean an acquired dataset in the context of sentiment classification. This process included data cleaning, exploratory data analysis (EDA), and pre-processing techniques aligned with natural language processing (NLP) best practices. First, data cleaning was performed to ensure high-quality input for model training. This step involved removing XML tags, special characters, and excessive whitespace and correcting grammatical and lexical errors with the assistance of language experts. Reviews containing sarcasm were also excluded, as they could introduce ambiguity into sentiment interpretation. Additionally, only reviews with a minimum length of 50 words were retained to ensure adequate contextual information. We conducted EDA using Python's Seaborn library, which is known for its powerful and aesthetically pleasing visualization capabilities. EDA provided insights into the class distribution, review lengths, and word fre-

quencies. Visualizations such as bar plots for sentiment class counts, histograms for review lengths, and word clouds were used to detect potential biases, imbalances, or anomalies within the dataset. These analyses helped confirm the balanced class distribution for both training and testing datasets, as shown in Table I. Third, the text analysis and pre-processing stages included lowercasing, stop-word removal, tokenization, and stemming. These steps helped to clean up the data, make it consistent, and improve how well techniques like TF-IDF and Word2Vec worked during model training. Overall, this combined method of cleaning and analyzing data made sure that the input data was organized, relevant, and reflected real feelings, which helped make the models more reliable.

We consulted language experts to correct grammatical and lexical errors and eliminate reviews containing sarcasm, which could distort sentiment analysis outcomes. To further increase the diversity of the training data and support better generalization across domains, we experimented with simple text-based data augmentation. One technique we applied was synonym replacement using WordNet, a lexical database that helps identify semantically similar words. For instance, we replaced adjectives like "*excellent*" in some reviews with synonyms like "*superb*" or "*outstanding*" when contextually appropriate. This method helped generate additional training instances without altering the core sentiment of the original text. We limited synonym replacement to non-critical words and avoided sentiment-bearing words to prevent changing the polarity. This lightweight augmentation was particularly useful in adding variation to common review patterns while maintaining data quality. This augmentation contributed to improving model robustness and generalization, especially when handling reviews from varied domains with differing expressions of sentiment.

This meticulous curation was crucial for minimizing noise and enhancing the accuracy of subsequent analyses. Only reviews with a minimum length of 50 words were included to provide sufficient context for sentiment analysis, ensuring that the classification is based on comprehensive and meaningful information. The cleaned and structured dataset is saved in .csv file format, and a detailed summary of our dataset is presented in Table 1, and examples of product reviews can be presented as Fig. 1.

Table 1: *Summary of the first dataset.*

Class Labels	Total of Product Reviews
Positive	10,000
Neutral	10,000
Negative	10,000

2.2 Data Separation for the First Dataset

10-fold cross-validation is employed to partition the data for developing a robust multiclass sentiment classification model. This method ensures balanced evaluation by using all data for both training and testing, reducing the likelihood of biased results and enhancing the overall reliability of the model's performance assessment.

To make sure the data splitting process is reliable and fair, we used stratified 10-fold cross-validation, which keeps the same balance of sentiment labels (positive, neutral, and negative) in each part. This approach minimizes bias in training and evaluation by preserving the proportion of each sentiment class across all folds. Additionally, we used a fixed random seed during data shuffling to ensure reproducibility across all model runs. All six models—logistic regression, SVM with linear kernel, random forest, CNN, LSTM, and BERT—were trained and validated on the same sets of data to keep the experiments consistent. Furthermore, we closely monitored the variance of performance metrics (accuracy, F1 score, and AUC) across the folds to detect any potential overfitting or instability. This consistent evaluation protocol confirmed the robustness of our data partitioning strategy and ensured reliable model comparison.

I've been using this phone for a month now, and it's incredible! The display is vibrant, and the performance is top-notch. I can multitask with several apps open, and it doesn't lag at all. The camera quality is amazing, especially in low light conditions. The battery life is decent, lasting me about a day and a half with regular usage. I also love the fast charging feature – it's super convenient. Overall, I'm very satisfied with my purchase and would recommend it to anyone looking for a premium smartphone experience.

a. An example of positive review

These wireless earbuds have decent sound quality, but they're not as comfortable as I expected. They tend to fall out when I'm running or working out, and the controls are a bit difficult to use. The battery life is good, lasting around 4 hours per charge. The connectivity is stable, but I occasionally experience some dropouts. They're fine for casual listening, but I wouldn't recommend them for more active use.

b. An example of neutral review

This laptop is a complete waste of money. It's extremely slow, even with basic tasks like browsing the web or checking emails. The keyboard stopped working properly after just a month, and now certain keys don't respond. The screen quality is terrible, with dull colors and poor viewing angles. I regret buying this and would strongly advise others to stay away.

c. An example of negative review

Fig.1: *Examples of product reviews.*

3. RESEARCH DESIGN

This section details our research design, in which we aim to identify the most suitable multiclass sentiment classification model that can effectively generalize for domain adaptation.

3.1 Tools

This work primarily utilizes Python and its libraries, including NumPy, NLTK (Natural Language Toolkit), scikit-learn, TensorFlow, and Hugging Face Transformers.

3.2 Multiclass Sentiment Classification Model-based Machine Learning

(1) Text pre-processing [17] – This stage is crucial for effectively handling tasks related to natural language processing (NLP) and text analytics. It involves transforming raw text data through cleaning and formatting to make it more suitable for subsequent analysis. In this study, the pre-processing of text generally includes the following procedures:

Lowercasing: Transforming all text to lowercase is essential for maintaining consistency across the dataset. This step ensures that words with the same meaning are not mistakenly treated as distinct entities due to variations in capitalization. By converting the entire text to lowercase, words like "*Computer*" and "*computer*" are recognized as identical, preventing misinterpretation as separate terms. This process is particularly important in text analysis for reducing redundancy and improving the accuracy of word matching and frequency calculations.

Stop-word Removal: Removing frequently used stop words such as "*and*," "*on*," and "*are*" is a crucial step in text pre-processing for sentiment analysis. While necessary for the grammatical structure of a sentence, these words generally carry little semantic weight and do not significantly contribute to understanding the underlying sentiment or context. By filtering out these common terms, the focus shifts to more meaningful words that are likely to influence overall sentiment, thereby enhancing the efficiency and accuracy of the analysis.

Tokenization: Tokenization is the process of segmenting text into smaller, individual units called tokens, which can be words, phrases, or symbols. This step is fundamental for text analysis as it breaks down the continuous stream of text into manageable pieces, making it easier to examine and process. By breaking sentences into separate parts—like turning "*sentiment analysis model*" into "*sentiment*," "*analysis*," and "*model*"—tokenization helps us analyze each piece more accurately, making it easier to count word frequency, parse text, and do more language-related analysis.

Stemming: Stemming involves converting words to their base or root form to manage different variations without altering their core meaning. For instance, words like "*computer*," "*computing*," and "*computes*" are reduced to the common stem "*comput*." This process consolidates similar words that share the same meaning but differ in tense, form, or suffix. By doing so, stemming simplifies text data and ensures that variations of a word are treated as a single entity during analysis, ultimately improving the consistency and relevance of the results.

(2) Vectorization and Term Weighting –This stage involves converting preprocessed text data into numerical format using methods such as TF-IDF (term frequency-inverse document frequency). TF-IDF is used to assign weights to words based on their significance within a document relative to their frequency across the entire corpus. TF quantifies how often a specific word appears in a document.

$$TF(w) = \log(1 + freq(w)\_within\_document\_d) \quad (1)$$

where $freq(w)$ is the frequency of the term w within a document.

While IDF is used to assess how uncommon a word is by considering its distribution across an entire collection of documents.

$$IDF(w, N) = \log\left(1 + \frac{N}{df(w)}\right) \quad (2)$$

where N represents the total number of documents in the dataset, while $df(w)$ denotes the number of documents that contain the term w.

(3) Feature Selection [18] – This concise process effectively selects and reduces features from TF-IDF vectors, making the data easier to analyze while retaining the most important information. We applied principal component analysis (PCA) to identify the most significant words for developing a multiclass sentiment classification model. After turning the cleaned reviews into numerical vectors with TF-IDF—where each word gets a score based on how often it appears and how important it is—we need to adjust the TF-IDF matrix so that each feature has an average of 0 and a standard deviation of 1, allowing for fair comparison in PCA. This process ensures that each feature has a mean of 0 and a standard deviation of 1, making all features comparable for PCA. Once standardized, the next step is to compute the covariance matrix of the standardized TF-IDF data to reveal the relationships between different terms (or words). Next, we perform eigen decomposition on the covariance matrix to obtain eigenvalues and eigenvectors. The eigenvalues indicate the amount of variance captured by each principal component, while the eigenvectors show the directions of maximum variance. Subsequently, we select the principal terms (or words) that account for the most variance in the data, typically using a threshold such as 95% cumulative variance. The selected components are then used to transform the original standardized TF-IDF data, reducing the dimensionality of the feature set. This reduced set of features can be leveraged for further analysis, such as classification or clustering, resulting in improved performance and lower complexity.

(4) Development of Multiclass Sentiment Classification model – This study compares three algorithms: logistic regression (LR), random forest (RF), and support vector machine (SVM).

Logistic Regression (LR) [19-20] – This algorithm is a widely used statistical method primarily designed for binary classification tasks, where the objective is to model the probability of one of two possible outcomes based on a set of predictor variables. It leverages the logistic function, or sigmoid function, to transform input features into a probability value between 0 and 1, making it suitable for distinguishing between two classes. To extend this capability to multiclass classification, techniques such as One-vs-Rest (OvR) or Softmax Regression are employed. In this study, the OvR strategy was chosen for multiclass classification because each model specializes in distinguishing one class from the rest, with the class having the highest predicted probability selected as the final output. This method is intuitive, easy to implement, and often performs well across various multiclass problems, making it a reliable choice for sentiment classification tasks.

In this study, we have a dataset of product reviews with three sentiment classes: positive, neutral, and negative. For the OvR strategy, we will train three separate logistic regression (LR) models:

Model-1: Predicts "Positive" vs. "Not Positive" (i.e., Neutral and Negative combined).

Model-2: Predicts "Neutral" vs. "Not Neutral" (i.e., Positive and Negative combined).

Model-3: Predicts "Negative" vs. "Not Negative" (i.e., Positive and Neutral combined).

As mentioned above, it provides specifics regarding the training steps as follows:

For each class k in the set of classes $\{1, 2, \ldots, K\}$, transform the problem into a binary classification problem. It can define the binary target variable y_k as follows. $y_k = 1$ if the instance belongs to class k, and $y_k = 0$ if the instance belongs to any other class. This transformation enables the training of a binary classifier that differentiates class k from all other classes.

To train the Logistic Regression model for class k; Initially, it trains a LR model to minimize the binary cross-entropy loss function for class k. The formula can be written as follows.

$$L(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_{i,k}\log(\hat{y}_{i,k}) + (1-y_{i,k})\log(1-\hat{y}_{i,k})] \quad (3)$$

where N is number of training samples, $y_{i,k}$ actual binary label for class k (1 or 0) for the i-th instance, $\hat{y}_{i,k}$ is redicted probability of the i-th instance belonging to class k, and $\hat{y}_{i,k}$ is calculated using the logistic function (sigmoid function):

$$\hat{y}_{i,k} = \sigma(z_i) = \frac{1}{1+e^{-z_i}} \quad (4)$$

where $z_i = \beta_0 + \beta_1 x_{i,1} + \beta_1 x_{i,2} + \ldots + \beta_n x_{i,n}$, where β_j are the coefficients learned during training, and $x_{i,j}$ are the feature values for the i-th instance.

In order to prevent overfitting, incorporate a regularization term into the loss function.

$$L(\beta) = -\frac{1}{N}\sum_{i=1}^{N}[y_{i,k}\log(\hat{y}_{i,k}) + (1-y_{i,k})\log(1-\hat{y}_{i,k})] + \frac{\lambda}{2}\sum_{j=1}^{n}\beta_j^2 \quad (5)$$

where λ is regularization parameter and β_j is coefficients of the model. The regularization term imposes a penalty on major coefficients to minimize overfitting.

In model training, LR also employs Gradient Descent Optimization to minimize the loss function and determine the ideal coefficients β. It can update the coefficients iteratively using the following formula:

$$\beta_j = \beta_j - \eta\frac{\partial L(\beta)}{\partial \beta_j} \quad (6)$$

where η is learning rate and $\partial L(\beta)/\partial\beta_j$ is Gradient of the loss function with respect to coefficient β_j.

The training stops when either the maximum number of iterations is reached or the change in the loss function falls below a predefined threshold. After training, you will have K separate binary classifiers. For prediction, pass the feature vector of a new instance through each of the K classifiers. Each classifier k will output a probability $\hat{y}^k$ indicating the likelihood that the instance belongs to class k. The predicted class is then chosen as the one with the highest probability.

Support Vector Machine (SVM) [21-22] –SVM is a type of supervised learning algorithm designed to find an optimal hyperplane that separates classes within a feature space. It performs particularly well in high-dimensional settings and is resistant to overfitting, making it useful when the number of features exceeds the number of samples. Although SVM was initially designed for linear classification tasks, it can be adapted for non-linear cases through the "kernel trick," which enables the model to construct complex decision boundaries. When using a linear kernel, SVM can also work for problems with multiple classes by using the OvR method, which treats each class as its own binary classification task, just like logistic regression does for multiple classes.

The purpose of SVM is to identify a hyperplane that maximizes the margin between two classes (positive and negative). The decision function for a linear SVM is as follows:

$$f(x) = w \cdot x + b \quad (7)$$

where w is the weight vector (coefficients), x is the feature vector, and b is the bias term.

The purpose of SVM is to minimize the hinge loss function while incorporating a regularization element to prevent overfitting.

$$L(x,b) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\max(0,1 - y_i(w.x_i + b)) \quad (8)$$

where $\|w\|^2$ is the regularization term that penalizes large weights to ensure a large margin, C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing classification errors, and $\max(0, 1 - y_i(w.x + b))$ is the hinge loss, which penalizes points that are not correctly classified or fall within the margin.

The SVM algorithm aims to find a hyperplane that maximizes the margin between the two classes. The margin is the distance between the hyperplane and the closest data points (support vectors). This is mathematically achieved by minimizing $\|w\|^2$, which ensures the hyperplane is as far away as possible from the nearest data points. Using the binary labels y_k, the SVM model is trained to learn the optimal hyperplane that distinguishes class k from all other classes. The decision boundary for class k is defined by the following formula:

$$w_k.x + b_k = 0 \quad (9)$$

where w_k and b_k are the weight vector and bias term learned during training for class k.

By iterating this procedure for each class, we generate a collection of binary classifiers— one for each class in the multiclass problem. The class with the highest decision value or probability is designated as the predicted class for any new case.

Random Forest (RF) [21-22] – RF is an ensemble learning method that combines multiple decision trees to improve classification performance. For multiclass sentiment classification (e.g., positive, neutral, negative), RF can be directly applied to handle multiple classes by training several decision trees and aggregating their results.

First, we randomly sample the training data with replacement to generate multiple subsets. Each subset is then used to train an individual decision tree, promoting diversity within the forest. At each node of a tree, Random Forest selects a random subset of features to determine the best split. The most suitable feature and threshold for splitting are chosen based on criteria such as Gini impurity or entropy.

RF utilizes Gini impurity to assess the likelihood of misclassification at each node. The formula is

$$G = 1 - \sum_{i=1}^{K} p_i^2 \quad (10)$$

Additionally, RF employs entropy to evaluate the information gain achieved by each split.

$$H(S) = -\sum_{i=1}^{K} p_i \log(p_i) \quad (11)$$

The tree continues to grow until it reaches a predefined stopping criterion, such as a maximum depth or a minimum number of samples per leaf. A forest of decision trees (e.g., 100 trees) is then trained, with each tree independently built on a distinct bootstrap sample to ensure randomness and diversity across the forest.

For each new review, every tree in the forest generates its prediction. The final classification is determined through majority voting, where the class receiving the highest number of votes (e.g., positive, neutral, or negative) is selected. Notably, since each tree is trained on a bootstrap sample, approximately one-third of the data—known as out-of-bag (OOB) data—is left out during training. This OOB data can then be used to estimate the model's performance, providing an unbiased error assessment.

3.3 Multiclass Sentiment Classification Model-based Deep Learning

This study selected Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) due to their complementary strengths in handling sentiment analysis [22-23]. CNNs excel at capturing local patterns and key phrases within text, allowing them to quickly and efficiently identify sentiment indicators without relying on sequential data, making them computationally efficient for shorter texts. In contrast, LSTMs are good at handling data that comes in a sequence, allowing them to understand how words relate to each other over longer distances in sentences, which is important for picking up subtle changes in sentiment and the order of words in more detailed or longer reviews. Together, CNNs and LSTMs provide a balanced approach by focusing on both local features and sequential context, making them well-suited for comprehensive text sentiment analysis across domains.

CNN [22-23] - The CNN model begins with the input layer, which serves as the initial entry point for the preprocessed data. This layer accepts padded sequences of integers that represent words in product reviews. By setting the maximum sequence length to 100, the input shape is defined as 100, ensuring uniformity across all input sequences, which is critical for the subsequent layers.

Next, the model progresses to the embedding layer, where the integer-encoded words are transformed into

dense vector representations using Word2Vec. This transformation is essential as it captures the semantic relationships between words, allowing the model to understand context more effectively. In this study, we use pre-trained Word2Vec embeddings to build an embedding matrix that connects each word index to its matching vector in a space that usually has 300 dimensions. The result of this layer is a 2D tensor, where each word in the input sequence is shown by its own embedding vector, adding important meaning to the input data.

Following the embedding layer, the model incorporates convolutional layers that apply convolutional operations to the embedded input data. These layers are designed to extract features from the text by capturing local patterns, such as n-grams—phrases or sequences of words. The number of filters in these layers determines how many distinct features the model will learn; typical configurations include 32 filters for the first layer, 64 for the second layer, and 128 for a potential third layer. The kernel size is set to 3, which defines the size of the sliding window used during the convolution operation. Each convolution operation generates an output according to the following formula:

$$output(i) = \sum_{j=0}^{k=1} input(i+j) \cdot filter(j) \tag{12}$$

where k is the kernel size and i indicates the position in the output. After the convolutional operations, the ReLU (Rectified Linear Unit) activation function is applied, introducing non-linearity into the model, as represented by

$$ReLU(x) = \max(o, x) \tag{13}$$

To further down-sample the feature maps generated by the convolutional layers, a pooling layer is added. This layer reduces the spatial dimensions of the data while retaining the most essential features, thereby minimizing computational requirements. Max pooling is employed, selecting the maximum value from each window, and is typically set to a pooling size of 2. The pooling operation can be expressed as

$$output = \max(input[i : i + pool\_size]) \tag{14}$$

where i is the starting index of the pooling operation. This step effectively condenses the feature maps while preserving critical information, thereby enhancing the model's efficiency.

To address the risk of overfitting, a dropout layer is introduced. During training, this layer randomly sets a specified percentage of neurons (commonly 50%) to zero, encouraging the model to learn robust features that do not rely on any specific neuron and thereby improving its generalization capabilities.

The output flattens and enters a fully connected (dense) layer after the pooling and dropout layers. This layer connects every neuron to all neurons in the previous layer, enabling the model to learn complex representations. The number of units in this layer is typically set to 128, with the ReLU activation function applied to maintain non-linearity. The output from this layer is calculated using the following equation:

$$output = activation(weights \cdot input + bias)$$

This layer integrates the features learned from the previous layers and prepares them for high-level abstraction.

Finally, the model concludes with the output layer, which is responsible for generating the final class predictions for the sentiment classification task. The number of units in this layer corresponds to the three sentiment classes: positive, neutral, and negative. We utilize a SoftMax activation function to ensure that the output represents a probability distribution across these classes:

$$softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{15}$$

where C is the total number of classes and z_i is the output score for class i. This SoftMax layer converts the output scores into probabilities, providing clear predictions for each sentiment class.

After establishing the model architecture, we compile it using a suitable loss function for multiclass classification, specifically categorical cross-entropy.

$$Loss = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{16}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for class i. An optimizer, such as Adam, is selected, typically with a learning rate set to 0.001, to effectively minimize the loss function.

During the training phase, the model is fitted to the training data using a specified batch size of 32 and a set number of epochs, typically starting at 20. Throughout this process, validation performance is closely monitored to prevent overfitting and ensure that the model generalizes well to new data.

LSTM [22-23] –This study develops a multiclass sentiment classification model using LSTM networks, which are well-suited for processing sequential data such as textual reviews. The following sections outline the steps involved in building and fine-tuning the model, accompanied by examples and relevant equations. Before training the LSTM model, the dataset undergoes preprocessing to prepare the text data for

analysis. First, text cleaning is performed to remove irrelevant elements from the reviews, such as special characters and excessive whitespace. After cleaning, the reviews are tokenized, or split into individual words. For example, the sentence "*I love this product!*" would be tokenized into ["*I*", "*love*", "*this*", "*product*"]. Because LSTMs require input sequences of uniform length, the tokenized sequences are then padded to a maximum length of 100 tokens.

$$Padded: [0, 0, 0, \ldots, 0, \text{“}I\text{”}, \text{“}love\text{”}, \text{“}this\text{”}, \text{“}product\text{”}]$$

The next step is to apply word embeddings to transform tokens into dense vector representations, using models such as Word2Vec. This layer converts each token into a corresponding embedding vector that captures its semantic meaning. For example, the word "*product*" might be represented as a 300-dimensional vector. If "*product*" is represented as $[0.1, 0.2, \ldots, 0.3]$ in the embedding space, this vector reflects its contextual meaning and relationships with other words.

Next, the LSTM model architecture, which consists of five layers, is constructed and applied.

Input Layer: This layer receives the padded sequences of embeddings, with an input shape defined as 300, where 300 represents the dimensionality of each embedding vector.

LSTM Layer: This layer processes the input sequences and learns to capture temporal dependencies. We may choose to use one or more LSTM layers, with a common configuration being 100 units in the LSTM layer. The output of the LSTM cell can be represented as follows:

$$h_t = f(W_h h_{t-1} + W_x x_t + b \tag{17}$$

where h_i is the hidden state at time t, W_h is the weight matrix for the hidden state, W_x is the weight matrix for the input, x_t is the input at time t, and b is the bias.

Dropout Layer: To prevent overfitting, it applies dropout after the LSTM layer, commonly set at a rate of 0.2.

Fully Connected Layer: To learn complex representations, we add a fully connected (dense) layer with 128 units and ReLU activation after flattening the LSTM output.

Output Layer: The output layer consists of three units (one for each sentiment class: positive, neutral, and negative). We use a SoftMax activation function to output the probabilities for each class.

$$P(y = k|X) = \frac{e^{z_k}}{\sum_{j=1}^{C} e^{z_k}} \tag{18}$$

where z_k is the output score for class k and C is the total number of classes.

The next step is to compile the model with a suitable loss function and the Adam optimizer. For multiclass classification in this study, we calculate categorical cross-entropy as follows:

$$Loss = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{19}$$

where y_i is the true label and $\hat{y}_i$ is the predicted probability for class i. The Adam optimizer is configured with a learning rate of 0.001.

For hyperparameter tuning, the batch size is set to 32, and the number of epochs is 20. Early stopping, based on validation loss or accuracy, may be used to halt training when the model stops improving.

3.4 Multiclass Sentiment Classification Model-based Transformer Learning

Bidirectional Encoder Representations from Transformers (BERT) [25-26] - This study applies the BERT-based model, which effectively leverages the complexities of product reviews across different categories for multiclass sentiment classification. The following steps detail the development process for the multiclass sentiment classification model.

Data Preparation: Firstly, the dataset must be preprocessed to remove irrelevant information, such as special characters and excessive whitespace.

Tokenization: Next, BERT's tokenizer is required to tokenize the reviews, splitting the text into subwords or tokens. BERT utilizes a WordPiece tokenizer, which can break down rare words into known subwords. For example, BERT may tokenize "*unhappiness*" into ["*un*", "*happiness*"]. The tokenization process also adds special tokens to the input sequences, such as [CLS] for the start and [SEP] for separating sentences when necessary. Since BERT has a maximum sequence length (typically 512), you may need to truncate or pad your sequences to meet this requirement.

Creating Attention Masks: After tokenization, generate attention masks to indicate which tokens are actual words and which are padding. The attention mask is a binary array where '1' denotes a valid token, and '0' indicates padding. This feature allows the model to focus only on the meaningful parts of the input data.

Embedding Layer: When feeding the tokenized input into BERT, the model transforms these tokens into dense vector representations through its embedding layer. The generated embeddings capture the contextual relationships between words. For example, the word "*bank*" in the context of "*river bank*" will have a different embedding than "*bank*" in the context of "*money bank.*"

BERT Model: After passing through the embedding layer, the output embeddings are processed by the BERT model, which consists of multiple trans-

former layers. These layers implement attention mechanisms, allowing the model to weigh the importance of different tokens in relation to one another. For instance, in the sentence "*The mobile was not great*," BERT understands that "*not*" negates "*great*," which impacts the sentiment classification.

Pooling Layer: After processing through the BERT layers, the output corresponding to the [CLS] token is typically extracted, as it is designed to capture the overall representation of the input sequence. This pooled output will serve as the input for the subsequent classification layer.

Fully Connected (Dense) Layer: A fully connected (dense) layer comes next to the pooling layer. This layer connects the pooled output to the classification outputs. The number of units in this layer corresponds to the number of sentiment classes—in this case, three: positive, neutral, and negative. This layer applies a ReLU activation function to introduce non-linearity.

Output Layer: The final layer is the output layer, which generates predictions for the sentiment classes. This layer uses a softmax activation function to transform the outputs into probabilities that add up to one for all classes. For example, if the raw output scores are [2.0, 1.0, 0.1], the softmax function will convert these scores into probabilities that indicate the likelihood of each sentiment for the given review.

Model Compilation: After defining the model architecture, compile it using an appropriate loss function for multiclass classification, such as categorical cross-entropy. Formula (20) provides the categorical cross-entropy loss.

During the training phase, it is necessary to fit the model to the training data using a specified batch size (commonly set to 32) and a predetermined number of epochs (usually starting at 20). Afterward, the model's validation performance should be monitored throughout the training process to prevent overfitting and ensure that it generalizes well to unseen data.

3.5 Model Fine-Tuning and Optimization Procedures

To improve the six machine learning models—logistic regression, SVM with a linear kernel, random forest, CNN, LSTM, and BERT—this study used special methods designed for each model's unique features. For the traditional models (Logistic Regression, SVM, and Random Forest), we used grid search to find the best settings. For traditional machine learning models (logistic regression, SVM, and random forest), we performed a grid search to identify optimal hyperparameters. For instance, we tuned the regularization parameter C for SVM, the number of estimators and maximum depth for Random Forest, and the penalty type (L1 or L2) for logistic regression. For deep learning models (CNN and LSTM), hyperparameter tuning involved adjusting the number of layers, the number of units per layer, learning rates, and dropout rates. We incorporated early stopping to prevent overfitting by monitoring the validation loss during training. We used the Adam optimizer with an initial learning rate of 0.001 and a batch size of 32. The number of epochs was empirically set to 20 based on validation performance. For the transformer-based model (BERT), we used pretrained weights from the Hugging Face Transformers library. Fine-tuning was conducted by adjusting learning rates within a range of 2e-5 to 5e-5, using batch sizes of 16 or 32, and limiting the number of epochs to 3–5 to avoid overfitting. We monitored the validation loss during training to guide early stopping decisions. Additionally, to ensure consistent and fair evaluation across all models, we used stratified 10-fold cross-validation, which maintains class distribution (positive, neutral, negative) in each fold. This strategy also supports reproducibility and reduces performance variability. These specific methods for fine-tuning and optimizing the model greatly helped make the model more stable, less likely to overfit, and better at working across different areas in sentiment classification tasks.

4. EXPERIMETAL RESULTS

4.1 Evaluation Metrics

In this study, the following evaluation metrics are utilized: accuracy [27-28], F1 score [27-28], and Area Under the Curve (AUC) [23].

Accuracy (ACC) [27-28]: Accuracy is the proportion of correctly predicted instances (both true positives and true negatives) out of the total number of predictions. While accuracy is often used when classes are balanced, it can be misleading in cases of class imbalance. High accuracy (above 90%) is favorable if the data is balanced, indicating that the model is making correct predictions most of the time. However, in imbalanced datasets, high accuracy may be deceptive, as it could indicate that the model performs well on the majority class but poorly on the minority class.

F1 [27-28]: The F1 score is the harmonic mean of precision and recall, providing a single measure of a model's accuracy when both metrics are important. A high F1 score (e.g., above 0.7) indicates a good balance between precision and recall, meaning the model effectively identifies positive instances while minimizing false positives. Conversely, a low F1 score (e.g., below 0.5) suggests that the model struggles with experimental results for each model based on 10-fold cross-validation precision, recall, or both, which is problematic in tasks where false negatives or false positives are costly.

AUC [29]: AUC is the area under the ROC curve, which illustrates the balance between the true positive rate (sensitivity) and the false positive rate. It evaluates how well a model distinguishes between pos-

itive and negative classes. An AUC of 1.0 indicates a perfect model that can accurately distinguish between positive and negative instances in all cases, while an AUC of 0.5 reflects random guessing, meaning the model has no ability to differentiate between the classes. When the AUC is less than 0.5, it suggests the model performs worse than random guessing, indicating poor predictive performance.

4.2 Evaluation of Models' Performances

Table 2 presents the performance of each multiclass text classification model, measured by accuracy, F1 score, and AUC.

Consider the experimental results of the machine learning models, including LR, SVM with a linear kernel, and RF, as presented in Table 2. Focusing on the results of the LR models, the accuracy, which reflects the overall correctness of the model, remains relatively consistent across all rounds, fluctuating between 0.765 and 0.774. While the accuracy is decent, it does not necessarily account for imbalanced classes, where certain classes might be predicted more accurately than others. The F1 score considers both precision and recall, weighted by the number of instances per class. With values between 0.74 and 0.75, the result suggests that the model maintains a good balance between predicting positive, neutral, and negative reviews. The F1 score is particularly important for multiclass sentiment analysis because it balances false positives and false negatives, indicating that the model does not overly favor one class over another. The AUC values, ranging from 0.77 to 0.78, indicate that the model has a fairly strong ability to distinguish between the different classes. Higher AUC values suggest that the model is better at ranking the correct class higher than incorrect ones. The consistency in the AUC scores shows that logistic regression can separate the sentiment categories well across the rounds. Nonetheless, the fourth model appears to be regarded as the optimal logistic regression model.

Consider the experimental results of the SVM utilizing a linear kernel, as presented in Table 2. The accuracy, which shows how many predictions were correct, is between 0.775 and 0.783, suggesting that the SVM model with a linear kernel makes fairly accurate predictions for all sentiment types (positive, neutral, Negative).

However, relying solely on accuracy can be misleading, especially with multiclass or imbalanced datasets, which is why additional metrics are critical for comprehensive evaluation. The F1 score, ranging from 0.75 to 0.76, balances both precision and recall while accounting for the number of instances in each class. The result suggests that the model effectively predicts all three sentiment classes without disproportionately favoring or neglecting any single class. Achieving this balance is particularly important in sentiment classification, where it is cru-

Table 2: *The Experimental Results of Each Model based on 10-fold cross Validation.*

Models	Rounds	ACC	F1	AUC
Logistic Regression	1	0.765	0.740	0.770
	2	0.771	0.750	0.780
	3	0.768	0.740	0.770
	4	**0.774**	**0.750**	**0.780**
	5	0.769	0.740	0.770
	6	0.773	0.750	0.780
	7	0.766	0.740	0.770
	8	0.772	0.750	0.780
	9	0.770	0.740	0.770
	10	0.771	0.750	0.780
SVM with Linear kernel	1	0.775	0.750	0.780
	2	0.780	0.760	0.790
	3	0.778	0.750	0.780
	4	0.783	0.760	0.790
	5	0.779	0.750	0.780
	6	**0.782**	**0.760**	**0.790**
	7	0.776	0.750	0.780
	8	0.781	0.760	0.790
	9	0.777	0.750	0.780
	10	0.780	0.760	0.790
Random Forest	1	0.823	0.810	0.840
	2	0.827	0.820	0.850
	3	0.831	0.820	0.850
	4	0.823	0.820	0.850
	5	0.830	0.820	0.850
	6	**0.832**	**0.830**	**0.860**
	7	0.826	0.810	0.840
	8	0.829	0.820	0.850
	9	0.831	0.820	0.850
	10	0.830	0.820	0.850
CNN	1	0.835	0.840	0.855
	2	0.840	0.845	0.860
	3	0.837	0.842	0.858
	4	0.843	0.848	0.862
	5	0.838	0.843	0.856
	6	**0.844**	**0.849**	**0.865**
	7	0.836	0.841	0.857
	8	0.839	0.844	0.861
	9	0.837	0.842	0.855
	10	0.842	0.847	0.863
LSTM	1	0.830	0.835	0.850
	2	0.835	0.840	0.855
	3	0.832	0.837	0.853
	4	0.838	0.843	0.857
	5	0.831	0.836	0.852
	6	**0.836**	**0.841**	**0.858**
	7	0.830	0.835	0.854
	8	0.834	0.839	0.856
	9	0.832	0.837	0.853
	10	0.835	0.840	0.855
BERT	1	0.845	0.850	0.865
	2	0.850	0.855	0.870
	3	0.848	0.853	0.868
	4	0.853	0.858	0.872
	5	0.847	0.852	0.866
	6	**0.854**	**0.859**	**0.873**
	7	0.846	0.851	0.867
	8	0.850	0.855	0.870
	9	0.849	0.854	0.868
	10	0.852	0.857	0.871

Note: The bold text highlights the best-performing model for each algorithm.

cial for the model to perform well across all classes rather than skewing toward a more frequent class. The AUC values, ranging from 0.78 to 0.79, further illustrate the model's strong ability to differentiate between correct and incorrect classifications. Rather than random guessing, the SVM model establishes a clear decision boundary that effectively separates the sentiment classes. The consistently high AUC values underscore the model's capacity to handle imbalanced data and accurately distinguish between positive, neutral, and negative sentiments. Nonetheless, the fourth model appears to be regarded as the optimal SVM model utilizing a linear kernel.

Consider the experimental results of the RF model presented in Table 2. The accuracy of the Random Forest model is consistently high, ranging from 0.827 to 0.832, indicating that it correctly predicts the majority of reviews across all three sentiment classes (positive, neutral, and negative). This consistent performance across different folds and rounds is a strong testament to the model's robustness. The F1 score, ranging from 0.81 to 0.83, highlights a well-maintained balance between precision and recall across all classes. This is particularly significant in multiclass sentiment classification, as it reflects the model's ability to handle each class effectively, accounting for both false positives and false negatives. The consistently high F1 score throughout all rounds demonstrates the model's ability to generalize well without favoring any single class disproportionately. Additionally, the AUC values, ranging from 0.84 to 0.86, indicate that the Random Forest model effectively distinguishes between sentiment classes. High AUC scores suggest that the model is adept at ranking the correct class higher than incorrect ones, even when there is some overlap between classes. This feature makes random forest a highly suitable choice for sentiment classification tasks where precise class differentiation is essential. Nonetheless, the sixth model is regarded as the optimal random forest model.

The RF model is the most reliable and efficient option for this study due to its superior accuracy, balanced performance, capacity to differentiate across classes, and robustness to overfitting. Proper management of high-dimensional text data, along with suitable feature selection methods, enhances its appropriateness for this multiclass sentiment classification problem. These combined strengths make Random Forest the most reliable model for obtaining consistent and interpreted outcomes in this study. Therefore, the combination of TF-IDF and PCA is highly beneficial for this experiment. TF-IDF ensures that the most relevant terms are emphasized in the classification process, while PCA reduces dimensionality and enhances the model's efficiency. Together, these methods help the Random Forest model perform well in classifying different sentiments, resulting in high accuracy, F1 score, and AUC values. Without these preprocessing steps, the model's performance would likely degrade due to irrelevant features and computational inefficiencies.

With a focus on CNN, LSTM, and BERT, the findings are as follows. The CNN model consistently demonstrated strong performance, achieving accuracy scores between 0.835 and 0.844, F1 scores from 0.840 to 0.849, and AUC values ranging from 0.855 to 0.865. CNNs are particularly effective at capturing local patterns in data, which is essential for text classification tasks, especially when utilizing word embeddings like Word2Vec. The CNN structure, which has convolutional and pooling layers, effectively pulls out important features from the text, allowing the model to work well with positive, neutral, and negative sentiment categories.

In comparison, the LSTM model exhibited slightly lower performance, with accuracy scores ranging from 0.830 to 0.838, F1 scores between 0.835 and 0.843, and AUC values from 0.850 to 0.858. LSTMs excel at capturing long-range dependencies within sequential data, making them well-suited for sentiment analysis. However, their design focuses more on the order of data rather than the arrangement of features, which might make them less effective than CNNs at picking up nearby details. These results indicate that while LSTMs can understand the timing of reviews well, they might struggle in situations where quick, nearby context is important for correctly identifying sentiment.

On the other hand, the BERT model outperformed both CNN and LSTM, achieving accuracy scores between 0.845 and 0.854, F1 scores from 0.850 to 0.859, and AUC values ranging from 0.865 to 0.873. BERT's transformer-based architecture uses attention mechanisms to capture contextual relationships between words in a sentence. This capability allows BERT to interpret nuanced sentiments in product reviews, making it particularly effective for complex and varied text inputs across different categories. Furthermore, BERT's pre-training on a large and diverse corpus enhances its adaptability to new domains and tasks, giving it a significant performance advantage over the other models.

When comparing feature extraction techniques, CNNs leverage convolutional operations to effectively capture local features, making them particularly beneficial for shorter text inputs like reviews. In contrast, LSTMs are designed to manage long dependencies and sequential information, which is advantageous but may not always yield the best results for classifying fixed-length texts. Meanwhile, BERT's attention mechanism considers the entire context of a sentence, leading to superior performance in sentiment classification, especially in cases where word meanings shift depending on context.

In terms of generalization and domain adaptation, both CNN and LSTM models require careful

hyperparameter tuning to perform well across different domains. BERT, however, benefits from its pre-training, which provides a strong foundation for adapting to various tasks with minimal additional training. The performance results from this study indicate that BERT's architecture is inherently more effective at handling domain adaptation, as reflected by its consistently higher scores in accuracy, F1, and AUC metrics.

While BERT models typically require more computational resources and longer training times due to their complexity, CNNs offer faster training times and a simpler architecture, making them appealing for applications with limited computational capacity. Thus, the choice of model should account for the specific requirements of the application, such as the need for computational efficiency, the importance of contextual understanding, and the extent of domain adaptation required.

In summary, this comparison shows that BERT outperformed all other models, reaching the best accuracy (average 0.85) and AUC (about 0.87) because of its attention mechanism and pre-training, which help it better adapt to different topics and understand context. CNN and LSTM also performed well, with CNN being particularly good at identifying specific text patterns, achieving an accuracy close to 0.84, while LSTM did similarly by effectively recognizing sequences. CNN and LSTM also performed well, with CNN being particularly good at identifying specific text patterns, reaching an accuracy close to 0.84, while LSTM did similarly by effectively understanding the order of words. Random Forest (RF) delivered reliable results, with an accuracy around 0.83, showing effectiveness with feature-rich text but less adaptability across domains. SVM with a linear kernel did better than logistic regression (LR), achieving an average accuracy of about 0.78, which shows it works well with data that can be separated in a straight line but doesn't adapt well to different areas. Overall, BERT's architecture positions it as the most suitable model for complex, cross-domain sentiment analysis, while CNN and LSTM are strong contenders for shorter, domain-specific text classifications.

4.3 Evaluation of Models' Performances with Other Datasets

This section evaluates the optimal models of each algorithm using several product review datasets pertaining to books, beauty & personal care, and automotive. The results are presented in Table 3.

The evaluation results from the Books, Beauty & Personal Care, and Automotive datasets show clear differences in how well the models—Logistic Regression (LR), Support Vector Machine (SVM) with a linear kernel, Random Forest (RF), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and BERT—perform, pointing out the strengths and weaknesses of each model in adapting to different areas for multiclass sentiment classification.

Table 3: *The Experimental Results of Each Model with Other Dataset.*

Models	Datasets	Average ACC	Average F1	Average AUC
Logistic Regression	Books	0.60	0.58	0.65
	Beauty & Personal Care	0.58	0.55	0.62
	Automotive	0.62	0.60	0.65
SVM with linear kernel	Books	0.63	0.60	0.68
	Beauty & Personal Care	0.60	0.58	0.65
	Automotive	0.65	0.62	0.68
Random Forest	Books	0.70	0.68	0.75
	Beauty & Personal Care	0.67	0.63	0.70
	Automotive	0.72	0.68	0.75
CNN	Books	0.75	0.74	0.78
	Beauty & Personal Care	0.72	0.70	0.75
	Automotive	0.76	0.74	0.79
LSTM	Books	0.72	0.71	0.74
	Beauty & Personal Care	0.70	0.68	0.72
	Automotive	0.73	0.72	0.75
BERT	Books	0.83	0.81	0.85
	Beauty & Personal Care	0.80	0.79	0.83
	Automotive	0.83	0.82	0.86

LR demonstrates relatively low performance, with limited accuracy, F1, and AUC scores across all datasets. This outcome reflects its simplistic linear approach, which is inadequate for capturing the complex patterns and dependencies inherent in textual data. Consequently, LR struggles with the nuances of sentiment that vary by domain, rendering it less effective for datasets characterized by subjective expressions, such as Beauty & Personal Care.

The SVM with a linear kernel demonstrates slightly improved performance over logistic regression, benefiting from its margin-maximizing capabilities, which handle linearly separable data more effectively. However, similar to logistic regression, SVM's adaptability is limited by its linear kernel, restricting its ability to generalize across the non-linear and context-dependent features inherent in sentiment data, particularly when applied to different product categories.

RF demonstrates a notable improvement, achieving higher accuracy, F1, and AUC scores across all datasets. Its ensemble approach enables it to capture more complex patterns, particularly in feature-rich datasets such as automotive, where domain-specific terminology and vocabulary are often more predictable. However, Random Forest cannot recognize the order of words in text, making it less suitable for understanding detailed feelings based on context compared to deep learning models.

CNN does a great job with all datasets because it can identify local patterns and relationships, leading to high accuracy and AUC scores, especially in the

Automotive dataset where these patterns are clear. CNN's convolutional layers effectively extract spatial features, making it well-suited for shorter or semi-structured texts. However, CNN's architecture may not fully capture long-range dependencies, which limits its adaptability in datasets such as Beauty & Personal Care, where context and nuanced sentiment play a crucial role.

The performance of LSTM is comparable to that of CNN, albeit with a slight decrease in scores. LSTM's strength lies in processing sequential information, enabling it to capture dependencies over longer spans within sentences—an asset in sentiment analysis where word order is significant. However, LSTM may not perform as well as CNN because the datasets are fixed-length, which doesn't take full advantage of LSTM's ability to handle sequences. Furthermore, its reliance on sequential processing can slow training and reduce adaptability for short, domain-specific phrases.

BERT stands out as the top-performing model, achieving the highest accuracy, F1, and AUC scores across all domains. Its transformer-based architecture, which incorporates attention mechanisms, enables it to capture complex contextual relationships between words within each sentence. This capability allows BERT to adapt effectively to domain-specific expressions through pre-training on large corpora and fine-tuning for specific sentiment contexts. The consistently high performance across all datasets underscores BERT's exceptional generalization capacity and its ability to accurately discern sentiment across diverse product categories.

In conclusion, BERT's performance highlights its capability to manage complex, cross-domain sentiment classification with a high degree of adaptability and contextual understanding. Conversely, the non-linear and context-rich characteristics of sentiment data constrain simpler models like logistic regression and SVM. Random Forest, CNN, and LSTM demonstrate moderate performance, each with unique strengths, yet they lack BERT's capacity to generalize effectively across diverse domains.

5. CONCLUSION

This study closely compares different machine learning models for adapting to different categories in multiclass sentiment classification, looking at how well logistic regression, support vector machine with a linear kernel, random forest, convolutional neural network, long short-term memory, and BERT perform on various datasets for different product types. The findings reveal that BERT consistently outperforms other models, achieving the highest scores in accuracy, F1, and AUC. This superior performance is largely attributed to BERT's transformer-based architecture, which incorporates attention mechanisms that enable it to capture nuanced, context-rich relationships between words. Such capabilities make BERT highly adaptable across domains, as it effectively generalizes and transfers knowledge from one domain to another, even when significant linguistic and contextual differences exist.

CNN and LSTM both perform well, with CNN being great at identifying specific patterns using its convolutional layers and doing well with datasets that have semi-structured or short texts. LSTM does well too, using its ability to process sequences to understand long-range connections, which is helpful when the order of words affects sentiment. LSTM works well too, using its ability to process information in order to understand connections between words that are far apart, which is helpful when the order of words really matters for understanding feelings. However, traditional models like logistic regression and SVM are not very adaptable because they depend on simple relationships and do not consider the context, highlighting the difficulties these models encounter in classifying feelings across different areas. Random Forest offers reliable yet moderate performance, particularly when working with feature-rich data, but it lacks the flexibility required for complex cross-domain tasks.

The significance of this study lies in its contribution to the understanding of how machine learning and deep learning models adapt to domain-specific sentiment analysis tasks. By carefully comparing these models in different areas, this research shows how BERT's design is better for dealing with complicated sentiment data that depends on context and emphasizes the need to choose models that fit specific application needs. These findings can help businesses and researchers pick the right algorithms by considering things like how fast they run, how flexible they are, and how well they understand language details, which will improve the accuracy of sentiment analysis in real-life situations.

This study gives useful information about choosing models and adapting them for sentiment classification, but more research is needed to look into hybrid or ensemble models that can mix the best parts of different algorithms for better results. Also, looking into other transformer-based models like GPT or T5 could give us more understanding of how to adapt models for different sentiment tasks. Future research might look into methods like adversarial training or transfer learning to improve how well models work in different and complicated areas. Finally, looking into lighter transformer models, like DistilBERT or ALBERT, could be a good way to create effective sentiment analysis systems that work well even in places with less computing power. By following these approaches, future research can build on this study's results, improving the area of domain-adaptive sentiment classification and allowing for more flexible and scalable solutions for various uses.

Moreover, future research could explore the inte-

gration of BERT with other deep learning architectures, such as CNN and LSTM, to develop hybrid models. Prior studies suggest that combining BERT's contextual embeddings with CNN's local pattern extraction or LSTM's sequential modeling capabilities can lead to improved sentiment classification performance. This study used the basic BERT model to keep things clear and make fair comparisons between different models, but looking into these combined methods might provide better results, especially in areas with complicated language or subtle feelings.

ACKNOWLEDGEMENT

This work was financially supported by Mahasarakham University.

AUTHOR CONTRIBUTIONS

Conceptualization, T. Lawan and J. Polpinij; methodology, T. Lawan and J. Polpinij; implementation, C. Chan, D. Singhakam, W. Watthusin and B. Luaphol; analysis and validation, J. Polpinij and B. Luaphol; Data collection and preparation, T. Uttha, A. Chotthanom and K. Namee; writing-original draft, T. Lawan and J. Polpinij; writing-review and editing, J. Polpinij. All authors have read and agreed to the published version of the manuscript.

References

[1] J. Polpinij and A.K. Ghose, "An ontology-based sentiment classification methodology for online consumer reviews," *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pp. 518-524, 2008.

[2] F. Batista and R. Batista, "Sentiment analysis and topic classification based on binary maximum entropy classifiers," *Procesamiento de Lenguaje Natural*, pp. 77-84, 2013.

[3] M. Bouazizi and T. Ohtsuki, "Sentiment analysis: From binary to multi-class classification: A pattern-based approach for multi-class sentiment analysis in Twitter," *Proceedings of the 2016 IEEE International Conference on Communications (ICC)*, pp. 1-6, 2016.

[4] E.S. Alamoudi and N.S. Alamoudi, "Sentiment classification and aspect-based sentiment analysis on yelp reviews using deep learning and word embeddings," *Journal of Decision Systems*, vol. 30, pp. 259-281, 2021.

[5] M. Taboada, J. Brooke, M. Tofiloski, K. Voli and M. Sted, "Lexicon-Based Methods for Sentiment Analysis," *Association for Computational Linguistics*, vol. 37, no. 2, pp. 267-307, 2011.

[6] K. Namee, J. Polpinij and B. Luaphol, "A Hybrid Approach for Aspect-based Sentiment Analysis: A Case Study of Hotel Reviews," *Current Applied Science and Technology.*, vol. 23, no. 2, 2023.

[7] J. Polpinij, N. Srikanjanapert and P. Sopon, "Word2Vec Approach for Sentiment Classification Relating to Hotel Reviews," *International Conference on Computing and Information Technology*, pp. 308-316, 2017.

[8] W. Medhat, A. Hassan and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Engineering Journal*, vol. 5, no. 4, pp. 1093-1113, 2014.

[9] M. Tsytsarau and T. Palpanas, "Survey on mining subjective data on the web," *Data Min Knowl Discovery*, vol. 24, pp. 478-514, 2012.

[10] L.C. Yu, J.L. Wu, P.C. Chang and H.S. Chu, "Using a contextual entropy model to expand emotion words and their intensity for the sentiment classification of stock market news," *Knowledge-Based Systems*, vol. 41, pp. 89-97, 2013.

[11] S. Mindrops and V. Kumar, "Multi-Class Sentiment Classification using Machine Learning and Deep Learning Techniques," *International Journal of Computer Sciences and Engineering*, vol. 8, no. 11, pp. 14-20, 2020.

[12] M. Attia, Y. Samih, A. Elkahky and L. Kallmeyer, "Multilingual Multi-class Sentiment Classification Using Convolutional Neural Networks," *The International Conference on Language Resources and Evaluation*, 2018.

[13] R.K. Das, M. Islam, M.M. Hasan, S. Razia, M. Hassan and S.A. Khushbu, "Sentiment analysis in multilingual context: Comparative analysis of machine learning and hybrid deep learning models," *Heliyon*, vol. 9, no. 9, 2023.

[14] H. Kim and Y. S. Jeong, "Sentiment Classification Using Convolutional Neural Networks," *Applied Sciences*, vol. 9, no. 11, 2347, 2019.

[15] [G. Zhou and X. Huang, "Modeling and Mining Domain Shared Knowledge for Sentiment Analysis," *ACM Transactions on Information Systems (TOIS)*, vol. 36, no. 2, pp. 1-36, 2017.

[16] B.P. Majumder and K. Mrini, "Exploring Domain Adaptability for Sentiment Classification Models," Accessed on 15 May 2024, Available at `https://www.semanticscholar.org/paper/Exploring-Domain-Adaptability-for-Sentiment-Models-Majumder-Mrini/9d68f940aa5ef1b65bc11a85a33a3509ae551d9f`

[17] A.W. Pradana and M. Hayaty, "The Effect of Stemming and Removal of Stopwords on the Accuracy of Sentiment Analysis on Indonesian-language Texts," *Kinetik*, vol. 4, no. 4, 2019.

[18] F. Song, Z. Guo and D. Mei, "Feature Selection Using Principal Component Analysis, Joint Conference on Lexical and Computational Semantics," *International Conference on System*

Science, Engineering Design and Manufacturing Informatization, 2010.

[19] M. Maalouf, "Logistic regression in data analysis: an overview," *International Journal of Data Analysis Techniques and Strategies*, vol.3, no.3, 2011.

[20] B. Zou, "Multiple Classification Using Logistic Regression Model," *Internet of Vehicles*, pp. 238–243, 2017.

[21] J. Polpinij, K. Namee and B. Luaphol, "Bug reports identification using multiclassification method," *Science, Enigneering, and Health Studies*, vol. 16, pp. 1-8, 2022.

[22] J. Polpinij and B. Luaphol, "Comparing of Multi-class Text Classification Methods for Automatic Ratings of Consumer Reviews," Multidisciplinary Trends in Artificial Intelligence, pp. 164-175, 2021.

[23] A. Garg, S. Vats, G. Jaiswal and A. Sharma, "Analytical Approach for Sentiment Analysis of Movie Reviews Using CNN and LSTM," Artificial Intelligence and Speech Technology, pp. 99–115, 2022.

[24] S. L. Ramaswamy and C. Jayakumar, "Review on positional significance of LSTM and CNN in the multilayer deep neural architecture for efficient sentiment classification," Journal of Intelligent & Fuzzy Systems, vol. 45, no. 4, pp. 6077-6105, 2023.

[25] A. Areshey and H. Mathkour, "Transfer Learning for Sentiment Classification Using Bidirectional Encoder Representations from Transformers (BERT) Model," Sensors, vol. 23, no. 11, 2023.

[26] A. Hamza, K.B. Majeed, M. Rashad and A. Jaffar, "An Integrated Approach for Amazon Electronic Products Reviews by Using Sentiment Analysis," Bulletin of business and economics, vol. 13, no. 2, pp. 142-153, 2024.

[27] R. Caruana and A. Niculescu-Mizil, "Data mining in metric space: an empirical analysis of supervised learning performance criteria," Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 69 – 78, 2004.

[28] L. Lavazza and S. Morasca, "Common Problems With the Usage of F-Measure and Accuracy Metrics in Medical Research," in *IEEE Access*, vol. 11, pp. 51515-51526, 2023.

[29] C. Ling, J. Huang and H. Zhang, "AUC: A Better Measure than Accuracy in Comparing Learning Algorithms," Canadian Conference on AI, 2003.

Thawatwong Lawan received his Master of Engineering degree in Computer Engineering from Khon Kaen University, Thailand. He is currently a lecturer at the Department of Computer Science, Faculty of Informatics, Mahasarakham University. His research interests include computer networks and data analytics.

Jantima Polpinij received her Ph.D. from the University of Wollongong, Australia. She works as an Associate Professor with the Department of Computer Science, Faculty of Informatics, Mahasarakham University. Her research interests include machine learning, data analytics, natural language processing and text mining.

Chanpong Chan is an undergraduate student in the Department of Computer Science, Faculty of Informatics, Mahasarakham University, Thailand. His areas of interest include data analytics, machine learning, natural language processing, and text mining.

Dolsun Singhakamis an undergraduate student in the Department of Computer Science, Faculty of Informatics, Mahasarakham University, Thailand. His areas of interest include machine learning, natural language processing, and text mining.

Theeraya Uttha received her Master of Science degree in Remote Sensing and Geographic Information Systems from Khon Kaen University, Thailand. She is currently a lecturer at the Department of Geo-Informatics, Faculty of Informatics, Mahasarakham University. Her expertise lies in data analytics.

Anirut Chotthanom received his Master of Science degree in Science Education from King Mongkut's Institute of Technology Ladkrabang, Thailand. He is currently a lecturer at the Department of Information Technology, Faculty of Informatics, Mahasarakham University. His expertise is in data analytics.

Bancha Luaphol received his Ph.D. in Computer Science from the Faculty of Informatics, Mahasarakham University, Thailand. He is currently an Assistant Professor at the Faculty of Administrative Science, Kalasin University. His research interests include machine learning, data analytics, natural language processing, and text mining.

Woraphot Watthusin holds a Bachelor of Science degree in Information Technology from the Faculty of Informatics, Mahasarakham University, Thailand. He is currently serving as a Computer Technical Officer at the Office of Registration and Processing, Mahasarakham University. His area of expertise is data analytics.

Khanista Namee received her Ph.D. from the United Kingdom. She is currently an Assistant Professor in the Department of Information Technology, Faculty of Industrial Technology and Management, King Mongkut's University of Technology North Bangkok (KMUTNB). Her areas of expertise include the Internet of Things (IoT) and data analytics.