Universiti Teknologi MARA
Cawangan Perlis
Jurnal Intelek

Volume 20 Issue 2 (August) 2025

Evaluating Machine Learning Algorithms for Sentiment Analysis: A Comparative Study to Support Data-Driven Decision Making

Nur Hafiza Mohamad Daud[1], Nor Hayati Shafii[2*], Diana Sirmayunie Md Nasir[3], Nur Fatihah Fauzi[4]

[1,2,3,4]Kolej Pengajian Perkomputeran, Informatik dan Matematik, Universiti Teknologi MARA, Cawangan Perlis, Kampus Arau, 02600 Arau, Perlis, Malaysia

*Authors' Email Address: [1]nrhfza2707@gmail.com ,[*2]norhayatishafii@uitm.edu.my , [3]dianasirmayunie@uitm.edu.my, [4]fatihah@uitm.edu.my*

Received Date: 15 May 2025
Accepted Date: 26 June 2025
Revised Date: 29 July 2025
Published Date: 31 July 2025

*Corresponding Author

ABSTRACT

This research investigates the accuracy and robustness of sentiment analysis models through a comparative analysis of three distinct machine learning algorithms: Bernoulli Naive Bayes, Linear Support Vector Machines, and Logistic Regression. The primary objective is to assess the performance of these models across various domains and datasets in sentiment analysis tasks. The study employs data from the IMDb 500k movie reviews dataset, utilizin12g machine learning techniques for sentiment classification. Specifically, the selected algorithms—Bernoulli Naive Bayes, Linear Support Vector Machines, and Logistic Regression—are employed to train the dataset. Upon evaluating the models, the findings reveal notable differences in accuracy. Both LinearSVM and Bernoulli Naive Bayes achieved the highest accuracy, with each recording 89% when rounded to the nearest hundredth. However, LinearSVM slightly outperforms Bernoulli Naive Bayes in other performance metrics. In contrast, Logistic Regression records the lowest accuracy among the three algorithms. These results highlight the significance of algorithm choice in sentiment analysis tasks, with LinearSVM and Bernoulli Naive Bayes outperforming Logistic Regression. The research contributes valuable insights into the comparative performance of these algorithms, providing guidance for practitioners and researchers in choosing effective models for sentiment analysis across diverse datasets and domains.

Keywords: *accuracy, Bernoulli Naïve Bayes, machine learning, sentiment analysis, Support Vector Machine*

INTRODUCTION

Reviews are evaluations or assessments of something, often providing opinions or critiques. In the context of movies or products, reviews offer insights into the quality, strengths, weaknesses, and overall experience. They can be subjective, reflecting the individual reviewer's preferences, or objective, focusing on specific criteria. Reading reviews helps people make informed decisions by considering others' perspectives before watching a movie or purchasing a product.

E-ISSN 2231-7716
DOI: http://10.24191/ji.v20i2.6492

IMDb, or the Internet Movie Database, founded in 1990, is one of the most intensive and popular online databases that provides a vast collection of information related to films, television series, and video games. It includes details such as cast and crew lists, release dates, trivia, quotes, reviews, and ratings. IMDb is widely used by enthusiasts, industry professionals, and the public as a comprehensive resource for information about movies and TV shows. Users can create accounts to rate and review titles, contributing to a dynamic community of film enthusiasts. In addition, it features a mobile application that enables users to access the platform conveniently and stay informed about the latest releases and updates.

Sentiment analysis is a natural language processing (NLP) technique that involves the use of computational methods to determine the sentiment or emotional tone behind a piece of text. Sentiment analysis detection models are designed to automatically classify the sentiment expressed in textual data, such as reviews, social media posts, customer feedback, and more. The primary goal of sentiment analysis on movie reviews is to extract and understand the sentiments expressed by viewers. By analyzing the sentiment of large amounts of text data, sentiment analysis models enable to determine whether the overall sentiment toward a movie is positive, negative, or neutral. Additionally, it uncovers common themes or aspects of the movie that consistently receive positive or negative feedback. By analyzing audience reactions, it will provide valuable insights to filmmakers, studios, and actors, to help them understand what worked well and what areas might need improvement. Incorporating sentiment analysis into movie marketing helps in creating more targeted and effective campaigns, enhancing the overall promotion and reception of a film.

Overall, Sentiment analysis plays a crucial role in understanding public opinion and audience feedback or reviews. However, achieving accurate and robust sentiment analysis remains a challenge due to the complexity of human language, the presence of noisy and ambiguous text, and the need to handle various types of sentiments. For example, in terms of subjective and ambiguous language, different individuals may interpret and express sentiments differently, making it challenging to create a unified and accurate sentiment analysis model. The nuances of sarcasm, irony, figurative language, or cultural context further complicate sentiment classification.

Another challenge in sentiment analysis is the availability of properly labelled and annotated training data. Human annotation of sentiment labels can be subjective, leading to inconsistencies or biases. Ensuring high-quality, reliable, and representative labelled datasets for training and evaluation poses a challenge. Moreover, sentiment analysis models trained on one domain may not generalize well to other domains. Sentiment expressions, vocabulary, or linguistic patterns can significantly differ across domains, such as social media, product reviews, news articles, or customer feedback. Adapting sentiment analysis models effectively to new domains is also a challenge. Contextual information plays a crucial role in sentiment analysis accuracy. The meaning of a word or phrase can change based on its surrounding context. Models must understand the relationships between words, sentences, or documents to capture sentiment accurately. Incorporating contextual information is challenging, especially in longer texts or complex linguistic structures.

Data sparsity is another problem in sentiment analysis, particularly for fine-grained sentiment analysis tasks. Collecting labelled data for specific sentiment categories (e.g., strongly positive, mildly negative) can be challenging, resulting in imbalanced datasets. Imbalanced data distributions can lead to biased sentiment predictions and lower accuracy for minority classes. Additionally, sentiment analysis often deals with noisy and unstructured text data, such as social media posts or customer reviews with spelling errors, abbreviations, slang, or informal language. These linguistic variations can impact sentiment classification accuracy, requiring robust pre-processing techniques and noise-handling strategies.

Sentiment analysis is feasible on three levels: phrase, document, and aspect. Sentiment analysis at the sentence or phrase level breaks down texts or paragraphs into sentences and identifies the polarity of

each sentence (Nandwani et al., 2021). The emotion is identified from the complete document or record at the document level (Nandwani, P., & Verma, R., 2021; Cui, L., Wang, Y., Wang, J., & Li, H., 2023; Ullah, S., Naeem, H., Jabbar, S., Latif, S., Khalid, S., & Rizwan, M., 2022). Document-level sentiment analysis is required to extract global sentiment from extensive texts including repetitive local patterns and a lot of noise. Taking into consideration the relationship between words and phrases as well as the whole context of semantic information to represent document composition is the most difficult component of document-level sentiment categorization (Liu et al., 2020a). It needs a more in-depth comprehension of the complex internal structure of feelings and dependent words (Li et al., 2020b).

Therefore, this research investigates the accuracy and robustness of sentiment analysis models by evaluating the performance of three specific machine learning algorithms: Bernoulli Naive Bayes (BernoulliNB), Linear Support Vector Machines (LinearSVM), and Logistic Regression (LR). By systematically comparing these algorithms, the study aims to gain insights into their effectiveness and explore strategies for enhancing the accuracy and resilience of sentiment analysis systems.

METHODOLOGY

In this study, a comprehensive analysis is performed on sentiment data from the IMDb movie reviews dataset, Retrieved from the Kaggle platform. The dataset undergoes preprocessing before being split into training and testing sets using the standard 70:30 ratio. Three classification algorithms are employed for sentiment analysis: Bernoulli Naive Bayes, Logistic Regression, and Linear Support Vector Machines (SVM). A comparative evaluation of these models is conducted, followed by hyperparameter tuning to assess its impact on model performance. The overall research workflow is illustrated in Figure 1.

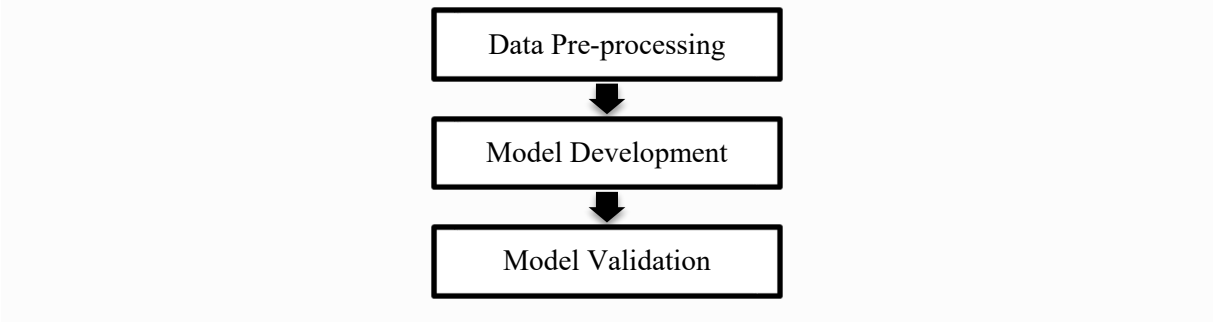

Figure 1: The steps of the machine learning process.

Data Pre-Processing

Preprocessing is a critical stage in the data preparation process for categorization. Data preprocessing is a necessary job for cleaning the data and preparing it for a machine learning model, which improves the accuracy and efficiency of the machine learning model. In this study, we began with data completion and noise reduction. This was followed by data transformation and a dimensionality reduction phase. Finally, the data was validated. Figure 2 illustrates the detailed steps involved in the data preprocessing process.

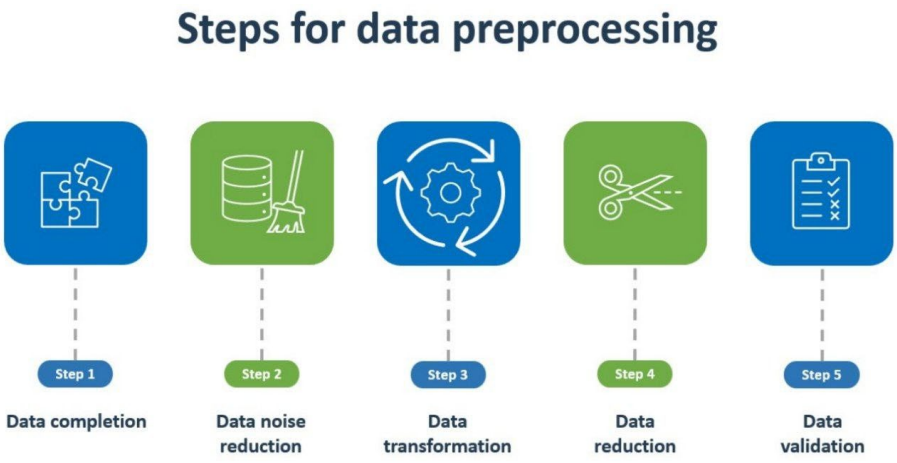

Figure 2: The details steps of data preprocessing.

Feature extraction is a crucial step in the model development pipeline. It involves transforming raw data into numerical representations that machine learning algorithms can understand and learn from. In this project, feature extraction will be carried out using Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) techniques.

The Bag of Words (BoW) model is a fundamental method in natural language processing where text is represented as a collection of its words, ignoring grammar and word order but preserving word frequency. This allows the textual data to be converted into numerical feature vectors suitable for machine learning models. The process involves:

i. Tokenization: Breaking down a text into individual words or tokens.
ii. Counting: Creating a vector that represents the frequency of each word in the document.

Term Frequency-Inverse Document Frequency (TF-IDF) is a method for measuring textual data from a corpus into numerical representation. The number of words that occur in a document or corpus is defined as term frequency. Using the TF-IDF method with classifier models improves the model's effectiveness. This measure helps to identify words that are important within a document but not too common across all documents, thus capturing the uniqueness of words in a specific context.

Term Frequency (TF): It measures how often a term (word) appears in a document. It's calculated as the ratio of the number of times the term appears in the document to the total number of terms in the document. The equation below states the term frequency formula.

$$TF = \frac{\text{Number of times term appears in document}}{\text{Total number of terms in the document}}$$

Inverse Document Frequency (IDF): It measures how important a term is across a collection of documents. It's calculated as the logarithm of the total number of documents divided by the number of documents containing the term. Inverse document frequency is shown in the equation below.

$$IDF = \frac{\text{Total number of documents}}{\text{Number of documents containing the term}}$$

TF-IDF Score: It's the product of TF and IDF. The TF-IDF score increases with the number of times a word appears in a document but is offset by the frequency of the word in the corpus. The equation below states the Term Frequency-Inverse Document Frequency (TF-IDF).

$$TF - IDF = TF \times IDF$$

Model Development

a. Bernoulli Naive Bayes (BernoulliNB)

Naïve Bayes is a class of basic probabilistic classifiers based on the Bayes theorem and naïve independence assumptions. This model family is also known by several other names, including Simple Bayes and Independence Bayes. Because the family of classifiers requires numerous parameters that are proportional to the number of variables in a learning task, it is extremely scalable. The Naive Bayes family of classifiers assumes that each feature contributes independently to the probability of a particular outcome, meaning that no relationships are assumed between the features. Since data can exist in different forms, different classification methods are required for effective analysis. The Naive Bayes family includes several variants; each suited to specific types of data. The Naive Bayes algorithm is formally expressed in Equation 1.

$$\mathrm{P}(X|Y) = \frac{\mathrm{P(Y|X)} * \mathrm{P(X)}}{\mathrm{P(Y)}} \tag{1}$$

Where *P(X|Y)* is the posterior probability of the class *X* given the feature *Y, P(X)* is the class *X*'s prior probability, *P(Y|X)* is the likelihood of the occurrence of the event. This is sometimes referred to as Bayes' theorem. When expressed in simplified language, the underlying concept is represented in Equation 2

$$\text{Posterior probability} = \frac{\text{Likehood} * \text{Class prior probability}}{\text{Feature prior probabilty}} \tag{2}$$

Bernoulli distribution is used for discrete probability calculation. It either calculates success or failure. Here the random variable is either 1 or 0 whose chance of occurring is either denoted by p or (1-p) respectively. The mathematical formula as in Equation 3 below:

$$\mathrm{f(x)} = \begin{cases} p^x * (1-p)^{1-x} & \text{if } x = 0,1 \\ 0\,, \quad otherwise \end{cases} \tag{3}$$

Where if, we put x =1 then the value of f(x) is p and if we put x = 0 then the value of f(x) is 1-p. Here, p denotes the success of an event.

Bernoulli Naive Bayes is a subcategory of the Naive Bayes Algorithm. It is used for the classification of binary features such as 'Yes' or 'No', '1' or '0', 'True' or 'False' etc. Here it is to be noted that the features are independent of one another. Bernoulli Naive Bayes is basically used for spam detection, text classification, Sentiment Analysis, used to determine whether a certain word is present in a document or not. The decision rule of Bernoulli NB is given as follows Equation 4 below:

$$p(x_i|y) = p(i|y)x_i + (1 - p(i|y))(1 - x_i) \tag{4}$$

where,

$p(x_i|y)$: conditional probability of x_i occurring provided y has occurred
i: event
x_i: holds binary value either 0 or 1

b. Linear Support Vector Machines (LinearSVM)

Vapnik and Lerner (1963) defined SVMs as discriminative classifiers based on a separating hyperplane. In other words, given supervised training data, the algorithm produces an ideal hyperplane that categorises fresh cases. This hyperplane is a line in two dimensions that divides a plane into two sections, with each class laying on one side.

Let us consider the case of two classes. Given a training dataset of n points of the form (x_1, y_1)…(x_n , y_n) where y_i has the value of either 1 or –1, indicating the class to which x_i belongs. The goal is to find the "maximum margin hyperplane" that divides the group of points of both classes. Any hyperplane can be represented as the set of points x that satisfy the condition given in Equation 5:

$$w'.x - b = 0 \tag{5}$$

where w is the hyperplane's normal vector and $\frac{b}{\|w\|}$ is the hyperplane's offset from the origin along the normal vector. The minimisation issue for the hard-margin situation is Equation 6:
minimise $\|\boldsymbol{w}\|$ subject to:

$$y_i(w'.x_i - b) \geq 1 \quad \text{for } i = 1, \ldots, n \tag{6}$$

The function we want to reduce in the soft-margin situation is in Equation 7:

$$[\frac{1}{n}\sum_{i=1}^{n} \max\,(0.1 - y_i(w'.x - b))] + \lambda\|\boldsymbol{w}\|^2 \tag{7}$$

where l is the trade-off between increasing the margin size and guaranteeing that each point is on the proper side of the margin. The kernel technique, a method of employing a linear classifier to tackle a non-linear issue, is used to turn SVM into a non-linear classifier. Boser, Guyon, and Vapnik (1992) proposed applying the kernel trick to maximum-margin hyperplanes to create non-linear classifiers. The resultant method is identical to the original, except that each dot-product is substituted by a nonlinear kernel function. This enables the algorithm to find the hyperplane with the greatest margin in a converted feature space. The Linear SVC method is formally expressed in Equation 8.

$$Linear\ Kernel = x^T x' \tag{8}$$

c. Logistic Regression (LR)

Logistic regression is also known as logit regression, maximum-entropy classification (MaxEnt), or the log-linear classifier in the literature. Despite its name, this linear model functions more as a classifier than a regressor. The logistic function is a monotonic function with values ranging from 0 to 1 that shows in Equation 9:

$$f(x) = \frac{L}{1 + e^{-k(x-x_0)}}, f(x) \in [0,1] \tag{9}$$

where x_0 is the sigmoid's midpoint, L is the curve's saturation point, and k is the logistic growth rate or steepness of the curve. The logistic regression objective function maximises the likelihood function. Maximum Likelihood Estimation (MLE) is denoted as follows Equation 10:

$$argmax_\beta: log\left\{\prod_{i=1}^{n} P(y_i|x_i)^{y_i}(1 - P(y_i|x_i))^{(1-y_i)}\right\} \tag{10}$$

where y_i is the output between 0 and 1, $P(y_i \mid x_i)$ is the posterior probability equal to 1/ (1+ $e.f$), and b is the weights/coefficients vector.

Model Validation

Model evaluation refers to the process of assessing how well a trained machine learning model performs on unseen data. Model validation plays a critical role in determining the model's accuracy, generalizability, and reliability prior to deployment. The evaluation process typically involves making predictions, selecting appropriate performance metrics, analyzing results, and optionally applying cross-validation for more robust assessment. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1-score, and ROC-AUC. In a multi-class classification setting, the model categorizes inputs into multiple classes—such as positive, negative, and neutral. Model performance is often described using the following classification outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

The following equations present the standard formulas for key metrics commonly used to evaluate the performance of a classification model in machine learning

$$Accuracy = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{11}$$

$$Accuracy = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{12}$$

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{13}$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

$$FI - Score = \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

Machine Learning Process Using Python Software

```
In [41]: X=dataset['review']
         y=dataset['sentiment']
         X_train=train_reviews
         X_test=test_reviews
         y_train=train_sentiments
         y_test=test_sentiments

         X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, test_size=0.3, shuffle=False)
```

```
In [42]: LogReg_clf_tfidf=Pipeline([('tfidf',TfidfVectorizer(tokenizer=LemmaTokenizer(),ngram_range=(1,2),sublinear_tf=True)),
                                   ('norm',Normalizer()),
                                   ('clf',LogisticRegression(random_state=42,C=1,penalty='l2'))])
```

In [44]:
```
LogReg_clf_bow=Pipeline([('bow',CountVectorizer()),
                         ('classifier',LogisticRegression())
                        ])
LogReg_clf_bow.fit(X_train,y_train)
Y_pred_LogReg_bow=LogReg_clf_bow.predict(X_test)
accuracy_bow_LogReg = accuracy_score(y_test, Y_pred_LogReg_bow)
print(f"Accuracy (BoW): {accuracy_bow_LogReg}")
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred_LogReg_bow))
```

```
Accuracy (BoW): 0.8800536372779082
              precision    recall  f1-score   support

    negative       0.88      0.88      0.88      7424
    positive       0.88      0.88      0.88      7491

    accuracy                           0.88     14915
   macro avg       0.88      0.88      0.88     14915
weighted avg       0.88      0.88      0.88     14915
```

In [45]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [43]:
```
LogReg_clf_tfidf=Pipeline([('tfidf',TfidfVectorizer()),
                           ('classifier',LogisticRegression())
                          ])
LogReg_clf_tfidf.fit(X_train,y_train)
Y_pred_LogReg_tfidf=LogReg_clf_tfidf.predict(X_test)
accuracy_tfidf_LogReg = accuracy_score(y_test,Y_pred_LogReg_tfidf)
print(f"Accuracy (TF-IDF): {accuracy_tfidf_LogReg}")
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred_LogReg_tfidf))
```

```
Accuracy (TF-IDF): 0.88990948709353
              precision    recall  f1-score   support

    negative       0.90      0.88      0.89      7424
    positive       0.88      0.90      0.89      7491

    accuracy                           0.89     14915
   macro avg       0.89      0.89      0.89     14915
weighted avg       0.89      0.89      0.89     14915
```

In [47]:
```
PipeNB_bow=Pipeline([('bow',CountVectorizer()),
                     ('classifier',BernoulliNB())
                    ])
PipeNB_bow.fit(X_train,y_train)
Y_pred_bow=PipeNB_bow.predict(X_test)
accuracy_bow_PipeNB = accuracy_score(y_test, Y_pred_bow)
print(f"Accuracy (BoW): {accuracy_bow_PipeNB}")
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred_bow))
```

```
Accuracy (BoW): 0.8516693483507642
              precision    recall  f1-score   support

    negative       0.84      0.88      0.86      4979
    positive       0.87      0.83      0.85      4965

    accuracy                           0.85      9944
   macro avg       0.85      0.85      0.85      9944
weighted avg       0.85      0.85      0.85      9944
```

In [46]:
```
PipeNB_tfidf=Pipeline([('tfidf',TfidfVectorizer()),
                       ('classifier',BernoulliNB())
                      ])
PipeNB_tfidf.fit(X_train,y_train)
Y_pred_tfidf=PipeNB_tfidf.predict(X_test)
accuracy_tfidf_PipeNB = accuracy_score(y_test,Y_pred_tfidf)
print(f"Accuracy (TF-IDF): {accuracy_tfidf_PipeNB}")
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred_tfidf))
```

```
Accuracy (TF-IDF): 0.8516693483507642
              precision    recall  f1-score   support

    negative       0.84      0.88      0.86      4979
    positive       0.87      0.83      0.85      4965

    accuracy                           0.85      9944
   macro avg       0.85      0.85      0.85      9944
weighted avg       0.85      0.85      0.85      9944
```

In [48]:
```
from sklearn.svm import LinearSVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import make_classification
X, y = make_classification(n_features=4, random_state=0)
clf = make_pipeline(StandardScaler(),
                    LinearSVC(dual="auto", random_state=0, tol=1e-5))
clf.fit(X, y)
print(clf.named_steps['linearsvc'].coef_)
print(clf.named_steps['linearsvc'].intercept_)
print(clf.predict([[0, 0, 0, 0]]))
```

```
[[0.14144338 0.52678408 0.67978708 0.49307555]]
[0.16935932]
[1]
```

In [50]:
```
Pipe_SVM_bow=Pipeline([('bow',TfidfVectorizer()),
                       ('classifier',LinearSVC())
                      ])
Pipe_SVM_bow.fit(X_train,y_train)
Y_pred_Pipe_SVM_bow=Pipe_SVM_bow.predict(X_test)
accuracy_bow_SVM= accuracy_score(y_test, Y_pred_Pipe_SVM_bow)
print(f"Accuracy (BoW): {accuracy_bow_SVM}")
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred_Pipe_SVM_bow))
```

```
Accuracy (BoW): 0.8919951729686243
              precision    recall  f1-score   support

    negative       0.90      0.88      0.89      4979
    positive       0.88      0.90      0.89      4965

    accuracy                           0.89      9944
   macro avg       0.89      0.89      0.89      9944
weighted avg       0.89      0.89      0.89      9944
```

```
In [49]: Pipe_SVM_tfidf=Pipeline([('tfidf',TfidfVectorizer()),
                                  ('classifier',LinearSVC())
                                 ])
Pipe_SVM_tfidf.fit(X_train,y_train)
Y_pred_Pipe_SVM_tfidf=Pipe_SVM_tfidf.predict(X_test)
accuracy_tfidf_SVM = accuracy_score(y_test,Y_pred_Pipe_SVM_tfidf)
print(f"Accuracy (TF-IDF): {accuracy_tfidf_SVM}")
from sklearn.metrics import classification_report
print(classification_report(y_test,Y_pred_Pipe_SVM_tfidf))
```

```
Accuracy (TF-IDF): 0.8919951729686243
              precision    recall  f1-score   support

    negative       0.90      0.88      0.89      4979
    positive       0.88      0.90      0.89      4965

    accuracy                           0.89      9944
   macro avg       0.89      0.89      0.89      9944
weighted avg       0.89      0.89      0.89      9944
```

Figure 3: Accuracy and classification reports of Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF)

FINDINGS AND DISCUSSION

Based on the accuracy values presented in Table 1, two feature extraction methods were evaluated: Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). The Linear Support Vector Machine (SVM) achieved the same accuracy score of 0.89 using both BoW and TF-IDF. Similarly, Logistic Regression also reached an accuracy of 0.89 with TF-IDF, matching the performance of SVM. However, its accuracy with BoW was slightly lower, also at 0.89, indicating consistent performance across both techniques. The lowest accuracy was recorded by the Bernoulli Naïve Bayes classifier, with a score of 0.85 for both BoW and TF-IDF. It's important to note that accuracy provides a general overview of model performance, but it may not fully capture how well the model performs in terms of overall correctness.

Table 1: Machine Learning accuracy of Bow and TF-IDF

Machine Learning Algorithm	Accuracy of Bag-of-Words (BoW)	Accuracy of Term Frequency-Inverse Document Frequency (TF-IDF)
Logistic Regression	0.88	0.89
Linear Support Vector Machine	0.89	0.89
Bernoulli Naïve Bayes	0.85	0.85

Classification reports were generated for each model, including metrics such as precision, recall, F1-score, and support. Table 2 compares the classification performance of the three models. For the positive class, both Logistic Regression and Linear Support Vector Machine (SVM) achieved the highest scores, with a precision of 0.88, recall of 0.90, and an F1-score of 0.89. The Bernoulli Naïve Bayes classifier also performed well, with a precision of 0.87, recall of 0.83, and an F1-score of 0.85. In the negative class, Logistic Regression and Linear SVM again achieved the best results, with a precision of 0.90, recall of 0.88, and F1-score of 0.89. Bernoulli Naïve Bayes followed closely, with a precision of 0.84, recall of 0.88, and an F1-score of 0.86. In terms of support (i.e., the number of actual occurrences of each class), Logistic Regression had 7,491 instances of the positive class and 7,424 instances of the negative class. Both Linear SVM and Bernoulli Naïve Bayes were evaluated on 4,925 positive and 4,979 negative instances.

Table 2: Classification report of three machine learning

	Logistic Regression	Linear Support Vector Machine	Bernoulli Naïve Bayes

	Positive	Negative	Positive	Negative	Positive	Negative
Precision	0.88	0.9	0.88	0.9	0.87	0.84
Recall	0.9	0.88	0.9	0.88	0.83	0.88
F1-score	0.89	0.89	0.89	0.89	0.85	0.86
Support	7491	7424	4965	4979	4965	4979

CONCLUSION AND RECOMMENDATION

In conclusion, the Linear Support Vector Machine (SVM) is the most suitable model for sentiment analysis on this dataset. It demonstrated slightly better overall performance than Logistic Regression, based on both accuracy and the classification report. Additionally, the SVM model outperformed Bernoulli Naïve Bayes across all evaluated metrics, making it the most effective choice among the three classifiers.

It is recommended to use a sufficiently large dataset when developing machine learning models, as a greater volume of data typically enhances model performance. Larger datasets offer more representative information for the model to learn from, which can lead to improved accuracy and better generalization to unseen data. Next, use several machine learning techniques that are appropriate for the dataset. Use supervised machine learning, such as K-nearest neighbour (KNN) or Recurrent Neutral Network (RNN), and compare it to basic machine learning, such as logistic regression. Evaluate the performance using other metrics, such as K-fold, to see how it differs from the standard metrics, which are the classification reports. If feasible, do sentiment analysis in a different language, such as Malay, and use the primary dataset rather than the secondary dataset. Other than that, employ fine-tuning and hyperparameter optimization. Perform experiment with multiple hyperparameters and model architectures to discover the best-performing configuration and leverage approaches like grid search or random search.

ACKNOWLEDGEMENTS

We express our heartfelt gratitude for the invaluable support and resources generously provided by the UiTM Perlis branch, which has played a pivotal role in facilitating the completion of this research. The encouragement, understanding, and unwavering belief in our work from the UiTM Perlis community have served as a profound source of motivation, propelling our research endeavours forward. We are sincerely appreciative of the nurturing environment fostered by the UiTM Perlis branch, which has empowered.

FUNDING

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

AUTHORS' CONTRIBUTION

Nur Hafiza spearheaded data collection and preparation efforts, while Nor Hayati and Diana Sirmayunie took charge of simulation and manuscript composition. Nur Fatihah played pivotal roles in data analysis and interpreting the results.

CONFLICT OF INTEREST DECLARATION

We certify that the article is the Authors' and Co-Authors' original work. The article has not received prior publication and is not under consideration for publication elsewhere. This manuscript has not been submitted for publication, nor has it been published in whole or in part elsewhere. We testify to the fact that all Authors have contributed significantly to the work, validity and legitimacy of the data and its interpretation for submission to Jurnal Intelek.

REFERENCES

Arya, V., Mishra, A. K., & Gonzalez-Briones, A. (2022). Analysis of sentiments on the onset of Covid-19 using Machine Learning Techniques. Advances in Distributed Computing and Artificial Intelligence Journal, 11(1), 45–63. https://doi.org/10.14201/adcaij.27348

A. M. Rahat, A. Kahir, and A. K. M. Masum (2019). Comparison of Naive Bayes and SVM Algorithm based on sentiment analysis using review datasets. 8th International Conference System Modeling and Advancement in Research Trends (SMART), 2019, 266–270. https://doi.org/10.1109/SMART46866.2019.9117512

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory (pp. 144–152). ACM Press. https://doi.org/10.1145/130385.130401

Cui, L., Wang, Y., Wang, J., & Li, H. (2023). Sentiment analysis: A review. Journal of Healthcare Engineering, 2023, Article ID 9816550. https://doi.org/10.1155/2023/9816550

Liu, F., Zheng, J., Zheng, L., & Chen, C. (2020). Combining attention-based bidirectional gated recurrent neural network and two-dimensional convolutional neural network for document-level sentiment classification. Neurocomputing, 371, 39–50. https://doi.org/10.1016/j.neucom.2019.09.012

Li, W., Zhu, L., Shi, Y., Guo, K., & Cambria, E. (2020). User reviews: Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models. Applied Soft Computing, 94, 106435. https://doi.org/10.1016/j.asoc.2020.106435

Nandwani, P., & Verma, R. (2021). A review on sentiment analysis and emotion detection from text. Social Network Analysis and Mining, 11(1), 81. https://doi.org/10.1007/s13278-021-00776-6

Ullah, S., Naeem, H., Jabbar, S., Latif, S., Khalid, S., & Rizwan, M. (2022). A hybrid deep learning model for sentiment analysis of movie reviews. Electronics, 11(24), 4096. https://doi.org/10.3390/electronics11244096

Sharma, K. (n.d.). Cyberbullying Score Classification Using Machine Learning Techniques MSc Research Project Data Analytics.

Sindhu, I., & Shamsi, F. (2023). Prediction of IMDB Movie Score and Movie Success by Using the Facebook. 2023 International Multi-Disciplinary Conference in Emerging Research Trends, IMCERT 2023. https://doi.org/10.1109/IMCERT57083.2023.10075189

Vapnik, V. N., & Lerner, A. Ya. (1963). Pattern recognition using generalized portrait method. Automatika i Telemekhanika, 24(6), 774–780.