NÖHÜ Müh. Bilim. Derg. / NOHU J. Eng. Sci., 2025; 14(4), 1447-1461
Niğde Ömer Halisdemir Üniversitesi Mühendislik Bilimleri Dergisi
Niğde Ömer Halisdemir University Journal of Engineering Sciences
Araştırma makalesi / Research article
www.dergipark.org.tr/tr/pub/ngumuh / www.dergipark.org.tr/en/pub/ngumuh

The effect of text representation and model selection on classification performance: A comprehensive comparison of TF-IDF, Bow and Transformer-based methods on the Covid19-FNIR dataset

Metin temsili ve model seçiminin sınıflandırma performansına etkisi: Covid19-FNIR veri seti üzerinde TF-IDF, BoW ve Transformatör tabanlı yöntemlerin kapsamlı bir karşılaştırması

Muhammet Sinan Başaraslan[1,*], **Fatih Bal**[2]

[1] *Istanbul Medeniyet University, Computer Engineering Department, 34700, Istanbul Türkiye*
[2] *Kırklareli University, Software Engineering Department, 39010, Kırklareli, Türkiye*

Abstract

This study evaluates the performance of various machine learning (ML) models on a dataset split into 80% training and 20% testing using Term Frequency-Inverse Document Frequency (TF-IDF) and Bag of Words (BoW) text vectorization. Transformer-based models like DistilBERT, RoBERTa, and alBERT were integrated with classical ML algorithms and ensemble methods such as Stacking, Hard Voting, and Soft Voting. Stacking achieved the highest performance with both methods—92.62% Accuracy (Acc) and 92.51% F1-score (F1) with TF-IDF, and 92.29% Acc and 92.41% F1 with BoW. Hard Voting with BoW yielded the highest Recall (95.23%). Classical models like Logistic Regression (LR) and Support Vector Machine (SVM) performed better with BoW, reaching 90.98% and 90.51% Acc, respectively. Overall, TF-IDF produced balanced outcomes, while BoW offered higher Recall and Precision in specific cases. These results highlight the significance of both model and text representation choices in achieving optimal classification performance.

Keywords: Fake news, ML, Text Representation, Pre-trained

Öz

Bu çalışmada, Terim Frekansı-Ters Doküman Frekansı (TF-IDF) ve Bag of Words (BoW) metin vektörleştirmesi kullanılarak %80 eğitim ve %20 teste ayrılmış bir veri kümesi üzerinde çeşitli makine öğrenimi (ML) modellerinin performansı değerlendirilmiştir. DistilBERT, RoBERTa ve alBERT gibi dönüştürücü tabanlı modeller, klasik makine öğrenimi algoritmaları ve Stacking, Hard Voting ve Soft Voting gibi topluluk yöntemleriyle entegre edilmiştir. Yığınlama her iki yöntemle de en yüksek performansı elde etmiştir- TF-IDF ile %92.62 Doğruluk ve %92.51 F1, BoW ile %92.29 Doğruluk ve %92.41 F1. BoW ile Hard Voting en yüksek geri çağırmayı (%95,23) vermiştir. Lojistik Regresyon ve DVM gibi klasik modeller BoW ile daha iyi performans göstererek sırasıyla %90.98 ve %90.51 Doğruluğa ulaşmıştır. Genel olarak, TF-IDF dengeli sonuçlar üretirken, BoW belirli durumlarda daha yüksek geri çağırma ve kesinlik sunmuştur. Bu sonuçlar, optimum sınıflandırma performansına ulaşmada hem model hem de metin temsili seçimlerinin önemini vurgulamaktadır.

Anahtar kelimeler: Sahte haber, ML, Metin Gösterimi, Önceden eğitilmiş

1 Introduction

The COVID-19 pandemic has not only led to a major global health crisis but has also triggered an unprecedented surge in the spread of misinformation on online platforms. As social media has become the primary source of real-time information, the line between verified facts and misleading claims has become increasingly blurred. The rapid spread of fake news about treatments, vaccines and infection rates has led to widespread confusion, public panic and, in some cases, harmful behavior. There is therefore an urgent need to develop robust methods to detect and prevent the spread of such misinformation.

In recent years, the field of Natural Language Processing (NLP) has witnessed a major transformation with the emergence of deep learning (DL) and transducer-based language models. Traditional approaches to fake news detection often rely on statistical representations of text such as TF-IDF and BoW, which, while effective to some extent, often fail to capture the semantic and contextual nuances of language. On the other hand, transducer-based models such as BERT, RoBERTa, aLBERT, and DistilBERT have shown remarkable success in a wide range of NLP tasks due to their ability to learn deep contextual representations from large-scale corpora.

In this work, we aim to evaluate the effectiveness of both traditional ML methods and modern transducer-based architectures for the task of COVID-19 fake news detection. For this purpose, we use the CoVID19-FNIR dataset, a carefully curated corpus containing fact-checked real and fake news related to the pandemic [1]. The dataset includes posts collected from verified Twitter accounts and sources

* Sorumlu yazar / Corresponding author, e-posta / e-mail: muhammet.basarslan@medeniyet.edu.tr (M. S. Başarslan)
Geliş / Received: 07.05.2025 Kabul / Accepted: 03.09.2025 Yayımlanma / Published: 15.10.2025
doi: 10.28948/ngumuh.1694988

such as Poynter, covering various regions including India, the USA, and Europe between February and June 2020.

We compare the performance of models using both TF-IDF/BoW representations and contextual embeddings derived from transformative models. Our goal is to analyze how different text representation techniques affect classification performance and identify the most effective model setup for detecting misinformation related to COVID-19. This research not only contributes to the growing literature on fake news detection but also provides practical insights for building reliable misinformation monitoring systems during global crises.

Section 2 presents the related work and Section 3 presents the technical background including data source, text representation, classification models, pre-trained models. Section 4 presents the experimental setup and results. Section 5 presents the results and discussion.

2 Related works

Several recent studies have utilized the CoVID-19 Fake News and Infodemic Research (CoVID19-FNIR) dataset to detect misinformation during the pandemic. These works vary in terms of the learning models and text representation techniques employed.

Sikosana et al. [2] evaluated both traditional ML algorithms such as Naive Bayes (NB), SVM, and Random Forest (RF), and DL models including Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), and a hybrid CNN+LSTM. In addition, they experimented with transformer-based pre-trained language models such as DistilBERT and RoBERTa. Their best results were achieved with a CNN+LSTM model using Word2Vec embeddings, yielding 99.21% Acc and 99.17% F1.

Vinay et al. [3] focused on ML approaches including NB, SVM, Decision Trees (DT), RF, and LR, and employed TF-IDF as a representation method. Their RF model achieved the highest performance with 99.14% Acc and 99.14% F1-score.

Qadess and Hannan [4] reported moderate results by comparing NB, Gradient Boosting, SVM, DT, RF, and LR models. Their highest F1-score was 91.0%.

Bozuyla and Özçift [5] applied both DL architectures (LSTM, Bi-LSTM) and transformer-based models (BERT, RoBERTa, BerTURK) adapted for Turkish texts. They obtained their best result using BerTURK, with 98.5% aCC and 98.4% F1-score, demonstrating the effectiveness of domain-specific transformers.

These studies confirm that the CoVID19-FNIR dataset enables high-Acc classification using a variety of approaches. However, most previous research has focused either on a single model category or has not systematically compared multiple representation methods across ML and DL architectures.

3 Material methods

This section describes the technical background, including data source, text representation, classification models, pre-trained models.

3.1 Data source

The CoVID19-FNIR dataset is a curated collection of news content related to the COVID-19 pandemic, specifically designed for fake news detection tasks. It includes fact-checked fake news obtained from the Poynter Institute and authentic news sourced from the verified Twitter accounts of credible news organizations. The dataset comprises samples gathered from various regions including India, the United States, and parts of Europe, covering online social media activity between February and June 2020. To ensure data quality and consistency, preprocessing steps such as the removal of special characters and irrelevant content have been applied. The overall class distribution within the dataset is illustrated in Figure 1.

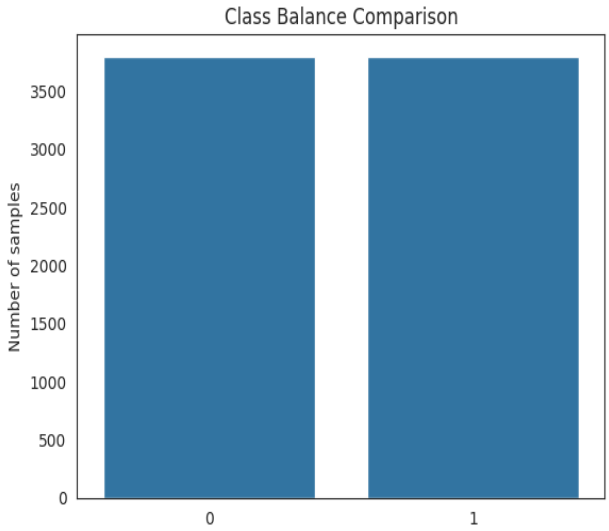

Figure 1. Class distribution of the dataset

According to the class distribution given in Figure 1, the dataset is balanced. Figure 2 shows the word cloud of the dataset

Although the dataset used in the study is open source and pre-processed, it has been reprocessed. In case these transactions are listed.

- HTML tags have been removed,
- special expressions such as URL and email have been removed,
- numbers have been removed,
- extra spaces between words have been removed,
- punctuation has been removed,
- it has been broken down into tokens,
- all text data has been converted to lower case so that words are in a single format,
- Stopwords were removed.
- Each word was lemmatized to its root form to reduce inflectional forms and improve semantic consistency.

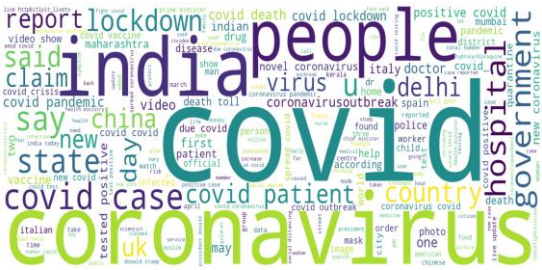

Figure 2. Dataset word cloud.

The wordcloud in Figure 2 reflects the prominent topics in public discourse during the COVID-19 pandemic, particularly centered in India. Various dimensions are prominently represented, such as the spread of the pandemic ("coronavirus", "covid", "case"), its social impact ("lockdown", "people"), its impact on the healthcare system ("hospital", "patient"), and government responses ("government", "state").

3.2 Text representation

Text representation methods, also known as language models, are probability distributions over groups of words that attempt to make sense of text in a context. Various statistical and probabilistic techniques are used to determine the probability of a given sequence of words occurring in a sentence. Language models are used in various NLP applications such as text classification, machine translation, etc.

In order to process textual data, the data must first be made understandable to machines. Text representation methods are used to do this. Text representation methods are a language modelling technique. In this technique, words are digitized into vectors and represented in vector space. Another name for this technique is text vectorization.

Text representation methods are classified as follows.

- Frequency-based methods,
- Prediction-based methods,
- Transformer methods.

In this study, BoW and TF-IDF are used as predictive methods and RoBERTa, DilstilBERT, alBERT is used as pre-trained transformative methods.

3.2.1 Frequency based methods

The methods used in this study, which are based on the occurrence of words in sentences, are TF-IDF and BOW. This section describes TF-IDF and BOW.

3.2.1.1 Term Frequency-Inverse Document Frequency

TF-IDF works by determining the relative frequency of words in each document and the inverse ratio of these words in the dataset. It is a statistically calculated weighting factor that indicates the importance of a term in a document [6]. TF is the work of Luhn [7] and IDF is the work of Jones [8]. TF is a technique used to determine the importance of a word within a specific document. It is calculated by dividing the frequency of a term by the total number of terms in the same document. This normalization helps account for document length and emphasizes terms that appear more frequently relative to the document size. The formula representing the TF calculation is given in Equation (1).

$$TF(i,j) = \frac{Number\ of\ terms\ i\ in\ document\ j}{Number\ of\ times\ term\ i\ appers\ in\ document\ j} \quad (1)$$

IDF is obtained by dividing the total number of documents by the number of documents in which the word occurs. As the value obtained as a result of this process approaches zero, it is understood that the word occurs in many places. If it is close to one, it is understood that it occurs less frequently. This indicates the IDF value, which is the importance of the word in the document. The IDF is shown in Equation (2) [9]. The TF-IDF score is calculated as given in Equation (3).

$$IDF(i) = log\left(\frac{Total\ \ number\ of\ doceument}{Number\ of\ documents\ containing\ term\ i}\right) \quad (2)$$

$$TF - IDF(i,j) = TF(i,j)x\ IDF(i) \quad (3)$$

In this study, version 1.0.2 of the Python sci-kit learn [10] library was used for the TF-IDF method.

Table 1. Examples from the dataset

Original data	After preprocessing
Breaking News! Visit https://covid-info.com for more info. Stay safe! 👂 Email us at info@covid.org	break news visit info stay safe email
COVID-19 vaccines are 95% effective!!! #pandemic #vaccine2021 😉	covid vaccine effective pandemic vaccine

3.2.1.2 Bag of Words

BoW is a commonly used approach in natural language processing that represents text based on word frequency, disregarding the order or grammar of the words in the document. Instead of considering the sequence of terms, BoW focuses solely on how often each word appears. This technique transforms documents into fixed-length feature vectors by counting the occurrences of known words across the dataset. These frequency-based vectors are then utilized by ML algorithms during the model training process [11,12].

3.2.2 Transformer text representation methods

Waswasi et al [13] developed a new text representation method called Transformer based on Attention network. In Transformers, the training time is reduced as the text representation process is performed in parallel.

Transformer architecture is a closed system consisting of six Encoders and Decoders.

The six Encoders and six Decoders used in this architecture are identical. The only difference is the word embedding at the bottom, which converts the sentences received by the Encoder into word vectors. There is also a parameter called position-embedding that converts the position of the words in the sentences into numerical code. This vector is the same length as the word embedding vector.

In this way, the position information of the words collected with word embedding vectors is known [13].

3.3 Classification models

In this section of the study, information on the models whose performance was evaluated for classification is presented.

3.3.1 Logistic regression

LR is a statistical model that examines the relationship between one or more independent variables and the independent variable [14]. The LR model transforms linear combinations of independent variables into a value between 0 and 1 by predicting the probability of an event using a logistic function [15]. The main purpose of the LR is to enable interpretation of the effect of variables on the outcome, with the coefficients reflecting the effect of a one-unit increase in the independent variables on the log-likelihood of the dependent variable [16].

3.3.2 Naive bayes

NB is a model that performs efficient and probabilistic classification using Bayes' Theorem [17]. The term “naive” means that the model assumes conditional independence between attributes and that each attribute contributes independently to the class probability [18]. This independence assumption allows NB to calculate probabilities efficiently, allowing for effective classification.

3.3.3 Decision tree

The DT model, which works by subdividing the dataset according to feature conditions, is based on a tree structure where each internal node represents an attribute test, each branch represents a test result, and each leaf node represents a class label or prediction [19]. This tree structure simplifies the decision-making process and enables effective monitoring and interpretation of results.

3.3.4 K-nearest neighbor

KNN is a simple and efficient model used in regression and classification models. Based on instance-based learning, this nonparametric algorithm classifies a data point according to the majority class of its KNN in the feature space; it is flexible and adaptive to different types of data as it makes no assumptions about the data distribution [20]. The most common class among these neighbors is selected as the predicted class for the new instance in classification tasks [21].

3.3.5 Support vector machine

SVM, developed by Vladimir Vapnik and colleagues, performs classification by creating an optimal hyperplane that separates data points of different classes in a high-dimensional space [22]. The main goal of SVM is to maximize the margin between the support vectors, which are the closest data points of each class, so that the model can perform strong generalization on new and unseen data [23].

SVM, which works on the principle of finding the optimal hyperplane separating classes with the largest margin, provides a robust and effective solution to overfitting in high-dimensional data where the number of dimensions exceeds the number of samples [24].

3.3.6 Adaptive boosting

Adaptive Boosting (AdaBoost) is a tree-based model that aims to create a strong classification model by combining weak and inaccurate prediction trees [25]. AdaBoost is basically a sequential modeling approach in which each subsequent weak classifier is trained on examples that were misclassified in previous steps [26]. AdaBoost starts by initially giving equal weight to all instances; after each weak classifier, it increases the weights of the misclassified instances, allowing subsequent classifiers to focus on these instances, gradually increasing the overall Acc of the model [27].

3.3.7 Stacking

It is a technique that combines multiple models to realize the classification result more efficiently. By combining the outputs of different base learners through a meta-model, this method utilizes the strengths of each algorithm to create a superior prediction model; the stacking approach can be applied with homogeneous or heterogeneous learners and generally provides high Acc [28]. The basis of stacking is a multi-layer structure consisting of a first layer with multiple base classifiers and a meta-model that learns the outputs of these models and makes the final prediction; algorithms such as DT, SVM and neural networks can be used in the base layer, while the second layer combines these predictions to form a powerful learning model [29].

3.3.8 Hard and Soft Voting

Voting classifiers are ensemble learning methods that aim to improve classification performance by combining the predictions of multiple models. There are two basic types of voting: Hard Voting and Soft Voting. Hard Voting, or majority voting, is an ensemble method where each model votes for only one class label and the class with the most votes is the final prediction [30]. The Hard Voting approach shows improved classification performance, especially when predictions from several classifiers that complement each other's strengths are combined [31]. In contrast to Hard Voting, Soft Voting considers not only the class labels of the models but also the probability distributions they provide for each class. These probabilities are averaged and the class with the highest average probability is selected as the final prediction [32]. This method provides a more precise combination of predictions by considering the confidence levels of different classifiers, which usually results in higher overall Acc.

3.3.9 Extreme gradient boosting

Extreme Gradient Boosting (XGBoost) is an optimized model of classical gradient boosting techniques, offering computational efficiency, performance, additional regularization methods and innovative tree learning strategies in predictive modeling [33].

XGBoost is a gradient boosting-based method that minimizes a loss function with a gradient descent algorithm,

forming an ensemble of DT where each tree corrects the errors of previous trees, resulting in high prediction Acc [34].

3.4 Pre-trained models

A pre-trained model is an AI model that has been trained on large data sets pre-task and has learned the general features of the language. It is then retrained and used for a specific task.

Pre-trained models are AI models that have been pre-trained on large datasets. Since these models have learned the general structure and statistical properties of the language, they can show high performance with much less data by fine-tuning for a specific task. Thus, both the training time is shortened, and the generalization ability of the model is increased [35]. In this study, comparative performance analysis was performed on text classification tasks using the pre-trained language models RoBERTa, DistilBERT and alBERT.

3.4.1 RoBERTa

RoBERTa (A Robustly Optimized BERT Pretraining Approach) is a version of the BERT architecture proposed by Liu et al. that further optimizes the training process and is trained on large datasets [36].

3.4.2 DistilBERT

DistilBERT is a smaller and faster version of BERT developed by Sanh et al. using the knowledge distillation technique to reduce model size and computational cost [37]

3.4.3 alBERT

aLBERT (A Lite BERT) is an architecture proposed by Lan et al. that aims to reduce the size of the model through strategies such as parameter sharing and embedding factorization [38]. These models are characterized by high Acc and representativeness in NLP tasks.

4 Experimental setup and results

In this study, experiments were conducted to detect fake news using both traditional and transformer-based NLP approaches. For traditional models, text data was vectorized using TF-IDF and BoW methods, which were then provided as input to classical ML algorithms such as NB, LR, DT, SVM, and KNN.

In parallel, contextual embeddings were derived from pre-trained transformer models — specifically DistilBERT, RoBERTa, and aLBERT — to capture deeper semantic information. Each sentence was tokenized using the Hugging Face tokenizer, and the [CLS] token embedding was extracted from the final hidden state of the transformer model. These fixed-size embeddings were not fine-tuned but used as static feature vectors. These vectors were then passed as input to classifiers including LR, SVM, KNN, DT, AdaBoost, and XGBoost, which were trained on top of the extracted embeddings. This pipeline allowed evaluation of traditional classifiers using transformer-derived features.

All experiments were conducted using the CoVID19-FNIR dataset with 80% training and 20% testing split. Preprocessing steps such as removal of special characters, lowercasing, and tokenization were applied prior to model input.

Hyperparameter tuning was performed using GridSearchCV for classifiers such as LR, XGBoost, DT, KNN, and SVM, and the configuration details are provided in Table 2.

Table 2. Parameter optimization table.

	Parameter	Values
LR	C	0.1, 1, **10**, 100
	penalty	l1, l2
XGB	n_estimator	50, **100**, 200
	max_depth	3, 4, **5**
DT	max_depth	5, **10**, 20
KNN	n_neighbor	1 to 20; **7**
SVM	C	0.1, 1,2,**3,**4,5,6,7,8,910

* Selected parameter is in bold

After the TF-IDF process, the training and test data were separated into 80% and 20% respectively, and the results of the confusion matrix are presented in Table 3. The Acc, F1, Precision, Recall, and Specificity metrics are presented in Table 4. The confusion matrix results for the same representation method using 5-fold cross-validation separation are given in Table 5, alongside the metrics obtained from this matrix in Table 6.

Table 3. Confusion matrix results of models with TF-IDF after 80%-20%.

Model	Predicted Values	Real Values	
		0	1
LR	0	676	67
	1	88	687
NB	0	661	121
	1	103	633
DT	0	656	70
	1	108	684
KNN	0	709	470
	1	55	284
SVM	0	696	90
	1	68	664
AdaBoost	0	546	54
	1	218	700
Stacking	0	693	44
	1	71	710
Hard Voting	0	698	63
	1	66	691
Soft Voting	0	673	36
	1	91	718
XGBoost	0	666	42
	1	98	712

According to the results of the confusion matrix presented in Table 3, the Stacking model with TF-IDF representation was the most successful. This model stood out

due to its high Acc and balanced TP and TN values. Soft Voting yielded the highest recall, while the LR and SVM models also demonstrated stable performance.

In contrast, the KNN and AdaBoost models exhibited inferior performance due to their elevated error rates. Overall, ensemble methods delivered stronger results when combined with the TF-IDF representation.

Table 4. Performance results of models with TF-IDF after 80%-20%.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.8972	0.8986	0.8865	0.9111	0.8845
NB	0.8523	0.8490	0.8590	0.8393	0.8657
DT	0.8825	0.8848	0.8636	0.9071	0.8585
KNN	0.6540	0.5194	0.8378	0.8376	**0.9279**
SVM	0.8959	0.8932	0.9066	0.8801	0.9110
AdaBoost	0.8205	0.8375	0.7627	0.9284	0.7141
Stacking	0.9262	**0.9251**	0.9091	0.9417	0.9071
Hard Voting	0.9163	0.9146	**0.9128**	0.9164	0.9136
Soft Voting	0.9084	0.9107	0.8791	0.9443	0.8715
XGBoost	**0.9165**	0.9189	0.8875	**0.9517**	0.8809

Table 5. Confusion matrix results of models with TF-IDF after 5-fold.

Model	Predicted Values	Real Values	
		0	1
LR	0	887	88
	1	868	159
NB	0	861	92
	1	931	617
DT	0	914	118
	1	717	71
KNN	0	910	58
	1	916	83
SVM	0	883	47
	1	874	55
AdaBoost	0	887	88
	1	868	159
Stacking	0	861	92
	1	931	617
Hard Voting	0	914	118
	1	717	71
Soft Voting	0	910	58
	1	916	83
XGBoost	0	883	47
	1	874	55

According to Table 4, the Stacking model achieved the highest Acc and F1 using the TF-IDF representation (92.62% Acc, 92.51% F1). This was followed by the XGBoost, Hard Voting and Soft Voting models. Notably, the XGBoost model achieved the highest recall value of 95.17%, making it the most effective at discriminating between positive classes. Conversely, the KNN model showed the lowest performance in all metrics. These findings suggest that ensemble methods with TF-IDF offer stronger classification performance.

According to the results in Table 5, the most successful models in the confusion matrices obtained using the BoW vectorization method with an 80:20 training-test split are Stacking and Hard Voting. The Stacking model demonstrated a balanced and robust performance with 617 TP and only 92 FN. The Hard Voting model was similarly successful in distinguishing positive classes, with 617 TP and 118 FN. The Soft Voting model showed a more moderate performance, with 583 TP and 47 FN, but its false positive values were higher compared to Stacking and Hard Voting. On the other hand, the KNN model showed the weakest performance with 142 FP and 103 FN, revealing its sensitivity to the data structure. Similarly, the AdaBoost model also failed to distinguish positive classes and showed a remarkably low performance, particularly with 218 FN.

Overall, the results demonstrate the effectiveness of the TF-IDF method in identifying positive classes within certain models, particularly when ensemble methods are employed.

Table 6. Performance results of models with TF-IDF after 5-fold.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.8976	0.8984	0.9111	0.8861	0.9097
NB	0.8525	0.8497	0.8394	0.8602	0.8452
DT	0.8826	0.8847	0.9071	0.8635	0.9035
KNN	0.6543	0.5199	0.3768	0.8382	0.6014
SVM	0.8961	0.8939	0.8808	0.9074	0.8857
AdaBoost	0.8209	0.8374	0.9283	0.7627	0.9099
Stacking	**0.9242**	**0.9251**	**0.9414**	**0.9093**	0.9401
Hard Voting	0.9147	0.9143	0.9162	0.9125	0.9169
Soft Voting	0.9166	0.919	0.9525	0.8878	**0.9495**
XGBoost	0.9076	0.9103	0.9444	0.8786	0.9408

According to the results in the Table 6, the Stacking method showed the best performance among the models trained with TF-IDF. This model outperformed all other methods in terms of Acc (0.9242), F1 (0.9251), and precision (0.9414).

The Soft Voting model achieved the highest success in terms of specificity (0.9495), excelling in distinguishing negative classes. Among the single models, XGBoost attracted attention with its high precision (0.9444), but its sensitivity remained relatively low (0.8786). LR and SVM showed balanced and strong performance, while Naive Bayes and Decision Tree yielded more average results. The

weakest model was KNN, with Acc (0.6543) and F1 (0.5199) values.

Overall, ensemble methods were found to be much more successful than individual models, with the Stacking model in particular delivering the most balanced and highest results across all metrics.

After BoW, the training and test data were split in an 80-20 ratio, and the confusion matrix results are presented in Table 7. The Acc, F1, Precision, Recall, and Specificity metrics are presented in Table 8.

The confusion matrix results for the same representation method with 5-fold cross-validation are presented in Table 9, and the results of the metrics obtained from this matrix are presented in Table 10.

Table 7. Confusion matrix results of models with BoW after 80%-20%.

Model	Predicted Values	Real Values	
		0	1
LR	0	675	49
	1	89	705
NB	0	646	76
	1	118	678
DT	0	655	67
	1	109	687
KNN	0	661	142
	1	103	612
SVM	0	679	59
	1	85	695
AdaBoost	0	546	54
	1	218	700
Stacking	0	689	42
	1	75	712
Hard Voting	0	672	36
	1	92	718
Soft Voting	0	695	56
	1	69	698
XGBoost	0	668	40
	1	96	714

According to the results in Table 7, Stacking, Hard Voting, and XGBoost emerged as the most successful models in the confusion matrices obtained using the BoW vectorization method with an 80:20 training-test split. Stacking demonstrated balanced and strong performance, achieving 712 true positives (TP) and only 75 false negatives (FN), while also keeping false positives low.

Hard Voting achieved the highest recall, with 718 TP and just 36 FN, highlighting its strong capability to distinguish positive classes. Similarly, XGBoost also performed impressively, with 714 TP and 40 FN, confirming its robustness. Soft Voting was competitive as well, achieving 698 TP and 56 FN, but performed slightly behind Hard Voting and XGBoost.

Conversely, the KNN model showed weak performance, with 142 false positives (FP) and 103 false negatives (FN), indicating its sensitivity to the data structure. Likewise, the AdaBoost model underperformed in identifying positive classes, producing a very high number of false negatives (218) despite achieving a reasonable number of true positives.

Table 8. Performance results of models with BoW after %80-%20.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.9098	0.9109	0.8879	0.9350	0.8836
NB	0.8722	0.8748	0.8518	0.8992	0.8455
DT	0.8841	0.8865	0.8631	0.9111	0.8573
KNN	0.8386	0.8332	0.8559	0.8117	0.8652
SVM	0.9051	0.9061	0.891	0.9218	0.8887
AdaBoost	0.8208	0.8373	0.7625	0.9284	0.7147
Stacking	0.9229	0.9241	0.9047	0.9443	0.9018
Hard Voting	0.9157	0.9182	0.8864	0.9523	0.8796
Soft Voting	**0.9177**	**0.9178**	**0.91**	0.9257	**0.9097**
XGBoost	0.9104	0.913	0.8815	**0.9469**	0.8743

Table 9. Confusion matrix results of models with BoW after after 5-Fold Cross validation

Model	Predicted Values	Real Values	
		0	1
LR	0	886	64
	1	848	100
NB	0	860	88
	1	868	186
DT	0	891	77
	1	717	71
KNN	0	904	55
	1	882	47
SVM	0	912	74
	1	877	52
AdaBoost	0	886	64
	1	848	100
Stacking	0	860	88
	1	868	186
Hard Voting	0	891	77
	1	717	71
Soft Voting	0	904	55
	1	882	47
XGBoost	0	912	74
	1	877	52

According to the performance results in Table 8, the Stacking model produced the best results with the BoW representation, achieving 92.29% Acc and a 92.41% F1 score. Hard Voting performed best at detecting positive classes, achieving the highest recall of 95.23%. Soft Voting and XGBoost also performed well, achieving balance and high performance.

Classical models such as LR and SVM produced strong results, achieving over 90% Acc. By contrast, the AdaBoost and KNN models performed relatively poorly, particularly in terms of specificity and precision. Overall, ensemble models with BoW vectorization provided the most consistent, high-quality performance.

Table 9 shows the classification trends of the models after 5-fold cross-validation using the BoW representation. Notably, the Stacking and NB models achieved relatively high TP values (186 for both), indicating stronger performance in identifying positive classes and thus higher recall. Conversely, models such as KNN, Soft Voting, and XGBoost were more successful at predicting the negative class correctly, as reflected in their lower numbers of false positives. The SVM model demonstrated balanced prediction ability, achieving 912 TN and 52 FN, which indicates stable and reliable performance across both classes.

Overall, these results confirm that the choice of model and the representation method significantly affect classification performance in the 5-fold cross-validation setting, with ensemble and probabilistic methods excelling in recall, while distance-based models like KNN performed better in controlling false positives.

Table 10. Performance results of models with BoW after 5-Fold Cross validation

Model	Acc	F1	Precision	Recall	Specificity
LR	0.9091	0.9109	0.9353	0.8877	0.9326
NB	0.8721	0.8747	0.899	0.8517	0.8945
DT	0.8841	0.8865	0.9111	0.8632	0.9072
KNN	0.8389	0.8334	0.8119	0.8561	0.8235
SVM	0.9051	0.9061	0.9221	0.8906	0.9205
AdaBoost	0.8209	0.8374	0.9283	0.7627	0.9099
Stacking	**0.9232**	**0.9243**	0.9444	0.905	0.9426
Hard Voting	0.9157	0.9181	**0.9525**	0.8862	**0.9494**
Soft Voting	0.9172	0.9174	0.9253	**0.9096**	0.9249
XGBoost	0.9106	0.9133	0.9474	0.8815	0.944

When Table 9 and Table 10 are evaluated together, the Stacking model demonstrated the best overall performance after 5-fold cross-validation with BoW representation. This model stood out with its high true positive values and low false negative rate, and also outperformed all other models with the highest Acc and F1 score. Hard Voting stood out with the highest precision and specificity values, making it the most successful method in reducing false positives.

The Soft Voting model also showed balanced performance, yielding results closest to Stacking with 91.72% Acc and strong Recall. XGBoost showed strong performance with high Precision and Specificity, but lagged behind Stacking and Soft Voting due to its relatively lower recall. Among the classic models, LR and SVM achieved strong results with Acc and F1 scores above 90%, while DT and NB showed moderate performance. In contrast, KNN achieved lower success, and AdaBoost was insufficient in distinguishing positive classes, particularly due to its low recall value. Overall, the results clearly show that ensemble methods with BoW representation offer more balanced and superior performance compared to classical models.

After the DistilBERT training, the data was split into training and test sets at a ratio of 80:20, and the results of the confusion matrix are presented in Table 11. The Acc, F1, Precision, Recall, and Specificity metrics are presented in Table 12.

The confusion matrix results for the same representation method using 5-fold cross-validation are given in Table 13, and the metrics obtained from this matrix are given in Table 14.

Table 11. Confusion matrix results of models with DistilBERT after 80%-20%.

Model	Predicted Values	Real Values	
		0	1
LR	0	1070	79
	1	110	1018
NB	0	981	183
	1	199	914
DT	0	879	238
	1	301	859
KNN	0	926	90
	1	254	1007
SVM	0	1013	62
	1	167	1035
AdaBoost	0	1021	186
	1	159	911
Stacking	0	1074	78
	1	106	1019
Hard Voting	0	1040	57
	1	140	1040
Soft Voting	0	1045	94
	1	135	1003
XGBoost	0	1047	81
	1	133	1016

According to the results in Table 11, the models trained using the DistilBERT representation achieved high classification performance with a training-to-test separation of 80:20. Notably, the Stacking model achieved high correct prediction rates for both classes. Ensemble methods such as Hard Voting and Soft Voting also produced balanced results and achieved high positive class recognition rates. XGBoost and SVM models also demonstrated high Acc and consistent

performance. In contrast, NB and DT models performed less well, particularly in negative class predictions.

Overall, models integrated with DistilBERT produced more stable and robust results than classical representations, and ensemble methods successfully balanced the two classes using this representation method.

Table 12. Performance results of models with DistilBert after 80%-20%.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.9170	**0.9188**	**0.9068**	0.9312	**0.9105**
NB	0.8322	0.8271	0.8212	0.8332	0.8314
DT	0.7633	0.7612	0.7405	0.7830	0.7449
KNN	0.8489	0.8541	0.7986	0.9180	0.7847
SVM	0.8994	0.9004	0.8611	0.9435	0.8585
AdaBoost	0.8485	0.8408	0.8514	0.8304	0.8653
Stacking	**0.9192**	0.9172	0.9058	0.9289	0.9102
Hard Voting	0.9135	0.9135	0.8814	**0.9480**	0.8814
Soft Voting	0.8994	0.8975	0.8814	0.9143	0.8856
XGBoost	0.9060	0.9047	0.8842	0.9262	0.8873

According to the performance results of the models based on the DistilBERT representation in Table 12, which have an 80%-20% training test ratio.The Stacking model was the most successful overall, achieving the highest Acc (Acc = 91.92%) and a very high F1 (91.72%). The LR and Hard Voting models also performed well, achieving high Acc and balanced F1, precision and recall values. SVM was particularly successful in correctly predicting positive classes, achieving a high recall of 94.35%, while Hard Voting improved upon this metric, reaching 94.80%. This demonstrates the effectiveness of these models in reducing false negatives. Conversely, some models, such as NB and DT, had lower Acc and balance metrics, indicating that these models are less adaptive to transformer-based vector representations.Overall, DistilBERT provided more consistent and higher performance than traditional text representation methods for many models, particularly when used with ensemble methods and DL integrations.

Table 13 results show that the most successful models with DistilBERT representation are SVM, XGBoost, and Soft Voting. These models stand out with high true positive values and low false negative rates. In contrast, some methods such as DT and Hard Voting have been weaker in capturing positive classes, while NB and Stacking have shown inconsistent performance.

Overall, ensemble methods and robust classical models (especially SVM and XGBoost) provide more reliable performance with DistilBERT.

Table 13. Confusion matrix results of models with DistilBERT after 5-Fold cross validation

Model	Predicted Values	Real Values	
		0	1
LR	0	1404	104
	1	1288	240
NB	0	1154	312
	1	1215	118
DT	0	1330	81
	1	1340	244
KNN	0	1410	102
	1	1365	75
SVM	0	1372	123
	1	1374	106
AdaBoost	0	1404	104
	1	1288	240
Stacking	0	1154	312
	1	1215	118
Hard Voting	0	1330	81
	1	1340	244
Soft Voting	0	1410	102
	1	1365	75
XGBoost	0	1372	123
	1	74	938

Table 14. Performance results of models with DistilBert after 5-Fold cross validation.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.917	0.9151	**0.9278**	0.9027	0.931
NB	0.8324	0.8273	0.8333	0.8214	0.8429
DT	0.7634	0.7612	0.7832	0.7405	0.7872
KNN	0.8491	0.8543	0.9181	0.7988	0.9115
SVM	0.8996	0.9005	0.9437	0.8611	0.9426
AdaBoost	0.8484	0.8408	0.8306	0.8512	0.846
Stacking	**0.9193**	**0.9173**	0.9291	**0.9058**	0.9325
Hard Voting	0.9133	0.9133	0.9479	0.8812	**0.9479**
Soft Voting	0.8996	0.8977	0.9145	0.8814	0.9177
XGBoost	0.906	0.9047	0.9264	0.884	0.9284

When Table 13 and Table 14 are evaluated together, the Stacking, Hard Voting, Soft Voting, and XGBoost models achieved the most successful results after 5-fold cross-validation with the DistilBERT representation. The Stacking

model stood out with its high number of true positives and low false negative rate and also led in overall performance with the highest Acc and F1.

Hard Voting was the method that best reduced false positives, particularly with the highest Precision and specificity values, while Soft Voting and XGBoost also attracted attention with their balanced and strong results. Among classical models, LR showed strong performance with 91.7% Acc and the highest precision value, while SVM also demonstrated consistent success. In contrast, DT, NB, KNN, and AdaBoost models performed poorly with lower Acc and F1 values. Overall, the results reveal that ensemble methods stand out under the DistilBERT representation and offer the most consistent performance.

After RoBERTa training, the data was split into training and test sets at a ratio of 80:20, and the results of the confusion matrix are presented in Table 15. The Acc, F1, Precision, Recall, and Specificity metrics are presented in Table 16. The confusion matrix results for the same representation method using 5-fold cross-validation are given in Table 17, and the metrics obtained from this matrix are given in Table 18.

Table 15. Confusion matrix results of models with RoBERTa after %80-%20.

Model	Predicted Values	Real Values	
		0	1
LR	0	1009	96
	1	171	1001
NB	0	954	212
	1	226	885
DT	0	909	237
	1	271	860
KNN	0	1050	125
	1	130	972
SVM	0	936	142
	1	244	955
AdaBoost	0	999	172
	1	181	925
Stacking	0	1027	90
	1	153	1007
Hard Voting	0	977	113
	1	203	984
Soft Voting	0	984	142
	1	196	955
XGBoost	0	1052	69
	1	128	1028

Table 15 shows the classification performance of the models trained using the RoBERTa representation, with 80% of the data used for training and 20% for testing. XGBoost is particularly notable as the most successful model, achieving a high number of correct predictions in both the negative (1052) and positive (1,028) classes. The Stacking and LR models also produced balanced and robust results with high Acc. Conversely, the NB and DT models performed poorly in recognising the negative class, resulting in more misclassifications. Ensemble models such as Hard Voting and Soft Voting performed strongly in distinguishing positive classes. Overall, models working with the RoBERTa representation demonstrated high levels of Acc, Recall and stability, and more advanced methods were more successful with this representation.

Table 16 shows the performance results for the models trained using the RoBERTa representation on the test set with an 80:20 training-testing split. The results show the Acc, F1, Precision, Recall, and Specificity metrics. The most successful model was XGBoost, achieving 91.22% Acc and a 91.26% F1. This was followed by Stacking, which also demonstrated high levels of Acc and Rec. While the KNN and NB models produced balanced results, others such as the DT and SVM models performed relatively poorly.

The ensemble models Hard Voting and Soft Voting produced good results, particularly in terms of precision; however, their overall Acc was lower than that of XGBoost and Stacking. These results demonstrate that models based on the RoBERTa representation offer a robust foundation for text classification and that achieving high-performance hinges on selecting the right model.

Table 16. Performance results of models with RoBERTa after %80-%20.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.8825	0.8559	0.8537	0.9125	0.8821
NB	0.8827	0.8832	0.855	0.9131	0.854
DT	0.7770	0.7703	0.7605	0.7839	0.7720
KNN	0.8875	0.8898	0.8821	0.8861	0.8841
SVM	0.8302	0.7932	0.7965	0.8705	0.8317
AdaBoost	0.8448	0.8466	0.8366	0.8431	0.8398
Stacking	0.8914	0.8922	0.8680	0.9180	0.8704
Hard Voting	0.8633	0.8616	0.8291	0.8970	0.8279
Soft Voting	0.8547	0.8496	0.8297	0.8705	0.8347
XGBoost	**0.9122**	**0.9126**	**0.8892**	**0.9371**	**0.8915**

According to the results in Table 17, after 5-fold cross-validation with RoBERTa representation, the most successful models were KNN and Soft Voting. Both models demonstrated strong performance in distinguishing positive classes, with 148 true positives and 118 false negatives.

LR and AdaBoost produced similar results, showing balanced performance with 278 true positives and 126 false negatives. DT and Hard Voting remained at an intermediate level with 226 true positives and 186 false negatives, while NB and Stacking models showed poor performance with 164 true positives and 311 false negatives due to high error rates. On the other hand, SVM and XGBoost were the models with the lowest performance in capturing the positive class, with only 91 true positives and 186 false negatives.

Overall, Soft Voting and KNN stood out under the RoBERTa representation, while some models were found to be inadequate, particularly in identifying positive classes.

Table 17. Confusion matrix results of models with RoBERTa after 5-fold cross validation.

model	Predicted Values	Real Values	
		0	1
LR	0	1324	126
	1	1252	278
NB	0	1193	311
	1	1378	164
DT	0	1228	186
	1	1311	226
KNN	0	1348	118
	1	1282	148
SVM	0	1292	186
	1	1381	91
AdaBoost	0	1324	126
	1	1252	278
Stacking	0	1193	311
	1	1378	164
Hard Voting	0	1228	186
	1	1311	226
Soft Voting	0	1348	118
	1	1282	148
XGBoost	0	1292	186
	1	1381	91

Table 18. Performance results of models with RoBERTa after 5-fold cross validation.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.8829	0.8825	0.9125	0.8544	0.9131
NB	0.8076	0.8017	0.8069	0.7964	0.8183
DT	0.7768	0.772	0.784	0.7603	0.7932
KNN	0.8879	0.884	0.8861	0.8818	0.8936
SVM	0.8306	0.832	0.8707	0.7966	0.8685
AdaBoost	0.8448	0.8396	0.8431	0.8361	0.853
Stacking	0.8933	0.8923	0.9181	0.868	0.9195
Hard Voting	0.8614	0.8619	0.8972	0.8293	0.8965
Soft Voting	0.8517	0.8498	0.8707	0.8298	0.8742
XGBoost	**0.9133**	**0.9124**	**0.9368**	**0.8893**	**0.9382**

According to the results in Table 18, XGBoost is the most successful model after 5-fold cross-validation with RoBERTa representation. XGBoost achieved the highest Acc, F1, and Precision values, and also strongly distinguished both positive and negative classes with Recall and Specificity results. Stacking also performed strongly, producing balanced results with an Acc of 0.8933 and an F1 score of 0.8923. The KNN and SVM models showed moderate success, with KNN being one of the best classical models, achieving 88.79% Acc and an F1 score of 0.884. LR also offered balanced performance with 0.8829 Acc and 0.8825 F1 scores. In contrast, DT and NB showed lower performance, while AdaBoost remained at an intermediate level.

Overall, the results in Table 18 show that XGBoost delivered the strongest performance under the RoBERTa representation, followed by Stacking and KNN, while LR and SVM stood out among the classical methods.

After the alBERT training and test data were separated by 80% and 20%, respectively, the results of the confusion matrix are presented in Table 19.

The Acc, F1, Precision, Recall, and Specificity metrics are presented in Table 20. The results of the confusion matrix for the 5-fold cross-validation separation using the same representation method are given in Table 21, alongside the metrics obtained from this matrix in Table 22.

Table 19. Confusion matrix results of models with alBERT after %80-%20.

Model	Predicted Values	Real Values	
		0	1
LR	0	1052	132
	1	128	965
NB	0	824	356
	1	356	741
DT	0	899	301
	1	281	796
KNN	0	939	227
	1	241	870
SVM	0	1001	102
	1	179	985
AdaBoost	0	932	203
	1	248	894
Stacking	0	1049	110
	1	131	987
Hard Voting	0	1042	95
	1	138	1002
Soft Voting	0	1054	125
	1	126	972
XGBoost	0	1036	121
	1	144	976

Table 19 shows the classification performance of the models trained with the aLBERT representation, after the data was separated into 80% for training and 20% for testing. Notably, the Stacking, Hard Voting and XGBoost models achieved high correct prediction rates in both the negative and positive classes. For instance, the Stacking model achieved a balanced outcome by correctly predicting 1049 negative and 987 positive examples. Similarly, the Soft Voting model achieved high success in predicting the positive class (TP = 972). Conversely, the NB and DT models demonstrated lower Acc, particularly in the negative classes, suggesting that they are less well adapted to the aLBERT representation.

Overall, the transformer-based alBERT representation, when combined with ensemble and DL-based models, produced robust and balanced classification results.

Table 20. Performance results of models with aLBERT after %80-%20.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.8856	0.8812	0.8830	0.8795	0.8915
NB	0.6873	0.6755	0.6755	0.6755	0.6983
DT	0.7444	0.7323	0.7391	0.7256	0.7619
KNN	0.7945	0.7880	0.7831	0.7931	0.7958
SVM	0.8766	0.8763	0.8475	0.907	0.8483
AdaBoost	0.8019	0.7986	0.7828	0.8149	0.7898
Stacking	0.8942	0.8912	0.8828	0.8997	0.8890
Hard Voting	0.8977	**0.8958**	0.8789	**0.9134**	0.8831
Soft Voting	**0.8898**	0.8805	0.8714	0.8897	0.8780
XGBoost	0.8836	0.8856	**0.8852**	0.8861	**0.8932**

As shown in Table 20, the results demonstrate the classification performance of the models trained using the aLBERT representation with an 80:20 training-testing separation. The Hard Voting model achieves the highest Acc (89.77%) and F1 (89.58%). This is closely followed by the Stacking and Soft Voting models, which also achieved a very balanced, high performance. Stacking stands out, achieving 89.42% Acc and 89.12% F1, while XGBoost demonstrates balanced performance in terms of both Precision (88.52%) and Sensitivity (88.61%). Conversely, the NB and DT models showed lower performance than the other models.

These results demonstrate the effectiveness and reliability of ensemble methods and advanced ML models combined with the aLBERT representation for text classification tasks.

Table 21 shows the confusion matrices obtained in the 5-fold cross-validation process using the aLBERT representation. These results are important for evaluating general trends and the success of the models' classification.

Notably, the KNN, Soft Voting and XGBoost models demonstrate high rates of correct prediction in both positive and negative classes. These models have a low misclassification rate and are well balanced. For instance, the SVM model correctly classified 1,054 negative samples and made 121 correct predictions in the positive class.

Conversely, models such as NB, LR and AdaBoost produced more incorrect predictions, particularly in the positive class, resulting in weaker performance compared to the other models. The same is true of the Stacking model: the low number of correct predictions in the positive class limits the model's overall performance.

Overall, this table shows that, with the aLBERT representation, the KNN, SVM, Soft Voting and XGBoost models are more stable and perform better in cross-validation.

Table 21. Confusion matrix results of models with alBERT after 5-fold cross validation.

Model	Predicted Values	Real Values	
		0	1
LR	0	1052	132
	1	824	356
NB	0	899	301
	1	939	227
DT	0	1001	102
	1	932	203
KNN	0	1049	110
	1	1042	95
SVM	0	1054	125
	1	1036	121
AdaBoost	0	1052	132
	1	824	356
Stacking	0	899	301
	1	939	227
Hard Voting	0	1001	102
	1	932	203
Soft Voting	0	1049	110
	1	1042	95
XGBoost	0	1054	125
	1	1036	121

Table 22. Performance results of models with aLBERT after 5-fold cross validation.

Model	Acc	F1	Precision	Recall	Specificity
LR	0.8858	0.8813	0.8797	0.8829	0.8885
NB	0.6873	0.6755	0.6755	0.6755	0.6983
DT	0.7444	0.7323	0.7256	0.7391	0.7492
KNN	0.7945	0.788	0.7931	0.7831	0.8053
SVM	0.876	0.8752	0.9062	0.8462	0.9075
AdaBoost	0.8019	0.7986	0.8149	0.7828	0.8211
Stacking	0.8942	0.8912	0.8997	0.8828	0.9051
Hard Voting	**0.8977**	**0.8958**	**0.9134**	0.8789	**0.9164**
Soft Voting	0.8898	0.8856	0.8861	**0.8852**	0.894
XGBoost	0.8836	0.8805	0.8897	0.8714	0.8954

Table 22 shows the performance metrics of each model at the end of the 5-fold cross-validation process using the aLBERT representation.

The Hard Voting model was the most successful, achieving 89.77% Acc, 89.58% F1, 91.34% precision and 87.89% recall. The Stacking and Soft Voting models showed similarly balanced and high performance, with Stacking achieving success in negative classes with 90.51% specificity.

The SVM model effectively classified the positive classes with 90.62% precision, whereas the other classical models (NB and DT) had lower success rates.

These findings demonstrate that models based on the aLBERT representation offer more stable and superior performance, particularly when combined with ensemble methods. They also emphasise the importance of robust evaluation methods, such as accurate text representation, cross-validation and model selection.

The study comparatively evaluates the various ML and DL models and text representation methods (TF-IDF, BoW, DistilBERT, RoBERTa and aLBERT) used. Transformer-based representations, particularly DistilBERT, RoBERTa, and aLBERT, generally produced higher Acc, F1 and Rec values than classical methods such as TF-IDF and BoW. Across all representations, ensemble models such as Stacking, Hard Voting and Soft Voting demonstrate the strongest and most consistent performance. Notably, Stacking achieves the highest success rates with both classical and transformative representations. Conversely, simpler models such as NB and DT performed poorly, with generally low precision and recall values. The results demonstrate that the choice of representation method directly impacts model performance, and that ensemble methods, particularly when combined with robust text representations, can significantly enhance classification performance.

5 Conclusion and discussion

This study assessed the classification performance of various ML and DL models on the CoVID19-FNIR dataset for the detection of false information related to the SARS-CoV-2 pandemic. Two widely used text vectorization techniques, TF-IDF and BoW, were employed. The dataset was split into 80% for training and 20% for testing.

In addition to classical ML algorithms such as LR and SVM, transformer-based models (DistilBERT, RoBERTa, aLBERT) and ensemble approaches (Stacking, Hard Voting, Soft Voting) were integrated. The results revealed that both the choice of model and text representation method significantly influence classification performance. Among all approaches, the Stacking ensemble model delivered the highest performance with both vectorization techniques — achieving 92.62% Acc and 92.51% F1 with TF-IDF, and 92.29% Acc and 92.41% F1 with BoW. Notably, the Hard Voting model, combined with BoW, achieved an impressive recall of 95.23%, highlighting its strength in correctly identifying positive samples.

Furthermore, LR and SVM showed improved performance with BoW compared to TF-IDF, reaching 90.98% and 90.51% Acc, respectively. While TF-IDF yielded more balanced performance across metrics, BoW stood out with higher precision and recall in several models — suggesting that word frequency-based representations may be more compatible with traditional and ensemble models.

Overall, the findings emphasize that optimal performance in fake news detection depends on the careful selection of both the model and the feature representation.

Future work will focus on evaluating domain-specific pre-trained BERT variants, exploring multilingual datasets for broader applicability, and testing model robustness in open-world scenarios. In addition, the explainability of model decisions will be investigated through explainable artificial intelligence techniques to enhance model transparency and user trust.

Conflict of interest

The authors declare, to the best of their knowledge, that there are no conflicts of interest or affiliations with any individual, institution, or organization that could influence the impartial evaluation of this manuscript.

Similarity rate(iThenticate): %18

References

[1] J. A. Saenz, S. R. Kalathur Gopal and D. Shukla, Covid-19 fake news infodemic research dataset (CoVID19-FNIR Dataset), IEEE Dataport, 2021. https://dx.doi.org/10.21227/b5bt-5244

[2] M. Sikosana, O. Ajao and S. Maudsley-Barton, A comparative study of hybrid models in health misinformation text classification. OASIS ’24: 4th Int. Workshop on Open Challenges in Online Social Networks, pp. 18–25. Poznań, Poland, 9-13 October 2024. https://doi.org/10.1145/3677117.3685007

[3] R. Vinay, B. Premjith, D. Shukla, and K. P. Soman, Feature engineering and selection for the identification of fake news in social media, 2nd Int. Conf. on Signal and Data Processing, Bhopal, India, 10-11 June 2022. https://doi.org/10.1007/978-981-99-1410-4_24.

[4] M. Qadees and A. Hannan, Cross comparison of COVID-19 fake news detection machine learning models, 17th Int. Conf. on Open Source Systems and Technologies, Lahore, Pakistan, pp. 1–7, 20–21 December2023.https://doi.org/10.1109/ICOSST60641.2023.10414227

[5] M. Bozuyla and A. Özçift, Developing a fake news identification model with advanced deep language transformers for Turkish COVID-19 misinformation data, Turkish Journal of Electrical Engineering and Computer Sciences, 30, 3, 908–926, 2022, https://doi.org/10.55730/1300-0632.3818.

[6] S. N. Başa and M. S. Basarslan, Sentiment analysis using machine learning techniques on IMDB dataset, 7th Int. Symp. on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Ankara, Turkey, pp. 1–5, 26-28 October, 2023, https://doi.org/10.1109/ISMSIT58785.2023.10304923

[7] H. P. Luhn, A statistical approach to mechanized encoding and searching of literary information, IBM Journal of Research and Development, 1, 4, 309–317, October 1957, https://doi.org/10.1147/rd.14.0309.

[8] M. B. Çaki and M. Sinan Başarslan, Classification of fake news using machine learning and deep learning, Journal of Artificial Intelligence and Data Science, 4, 1, 22–32, 2024, https://dergipark.org.tr/pub/jaida

[9] R. Sjögren, K. Stridh, T. Skotare, and J. Trygg, Multivariate patent analysis—Using chemometrics to

analyze collections of chemical and pharmaceutical patents, Journal of Chemometrics, 34, 1, 2020, https://doi.org/10.1002/cem.3041.
[10] D. Cournapeau, Scikit-Learn, https://scikit-learn.org/stable/about.html, Accessed 1 March 2003
[11] M. Tezgider, B. Yildiz, and G. Aydin, Improving word representation by tuning Word2Vec parameters with deep learning model, 2018 Int. Conf. on Artificial Intelligence and Data Processing (IDAP 2018), Malatya, Turkey, pp. 1–7, 28–30 September 2018, https://doi.org/10.1109/IDAP.2018.8620919
[12] A. Onan, Mining opinions from instructor evaluation reviews: A deep learning approach, Computer Applications in Engineering Education, 28, 1, 117–138, 2020, https://doi.org/10.1002/cae.22179.
[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, Advances in Neural Information Processing Systems 30 (NeurIPS 2017), Long Beach, California, USA, pp. 5999–6010, 4–9 December 2017.
[14] D. M. A. S. Elkahwagy, C. J. Kiriacos, and M. Mansour, Logistic regression and other statistical tools in diagnostic biomarker studies, Clinical and Translational Oncology, 26, 9, 2172–2180, 2024, https://doi.org/10.1007/s12094-024-03413-8.
[15] H. Mu and H. Nie, Research on the evaluation and enhancement strategies of college students' health human capital in 'Healthy Hunan' under the background of big data, Applied Mathematics and Nonlinear Sciences, 9 (1), 2024, https://doi.org/10.2478/amns-2024-0400.
[16] W. Mao et al., Power transformers fault diagnosis using graph neural networks based on dissolved gas data, Journal of Physics: Conference Series, 2387, 1, 012029, November, 2022, https://doi.org/10.1088/1742-6596/2387/1/012029.
[17] Ö. Bezek Güre, Classification of liver disorders Diagnosis using Naïve Bayes method, Bitlis Eren Üniversitesi Fen Bilimleri Dergisi, 13(1), 153–160, 2024, https://doi.org/10.17798/bitlisfen.1361016.
[18] F. F. Hasibuan, M. H. Dar, and G. J. Yanris, Implementation of the Naïve Bayes method to determine the Level of Consumer Satisfaction, SinkrOn, 8 (2), 1000–1011, 2023, https://doi.org/10.33395/sinkron.v8i2.12349.
[19] H. A. Abdulqader and A. M. Abdulazeez, Review on Decision Tree Algorithm in Healthcare Applications, Indonesian Journal of Computer Science, vol. 13, no. 3, Jun. 2024, https://doi.org/10.33022/ijcs.v13i3.4026.
[20] R. Rahim and A. S. Ahmar, Cross-Validation and Validation Set Methods for Choosing K in KNN Algorithm for Healthcare Case Study, JINAV: Journal of Information and Visualization, 3(1), 57–61, 2022, https://doi.org/10.35877/454RI.jinav1557.
[21] F. Aldi, I. Nozomi, and S. Soeheri, Comparison of Drug Type Classification Performance Using KNN Algorithm, SinkrOn, 7(3), 1028–1034, 2022, https://doi.org/10.33395/sinkron.v7i3.11487.
[22] C. Cortes and V. Vapnik, Support-vector networks, Machine Learning, 20, (3) 273–297, 1995, https://doi.org/10.1007/BF00994018.
[23] T. S. Eswar and V. Karthick, Realtime visual object recognition using support vector machine comparing with K-Nearest Neighbor algorithm for improving accuracy, Journal of Pharmaceutical Negative Results, 13(SO4),2022,https://doi.org/10.47750/pnr.2022.13.S04.097.
[24] J. Cai, M. Wang, and Y. Wu, Research on pedestrian crossing decision models and predictions based on machine learning, Sensors, 24 (1), 258, 2024, https://doi.org/10.3390/s24010258.
[25] M. A. M. Mohammed and F. Türk, A Research: investigation of financial applications with blockchain technology, Hittite Journal of Science and Engineering, 11 (1), 33–40, 2024, https://doi.org/10.17350/HJSE19030000329.
[26] Y. Chen, S. Chen, Y. Yang, and S. Lu, Comparison of decision tree and ensemble algorithms, Applied and Computational Engineering, 55 (1), 241–248, 2024, https://doi.org/10.54254/2755-2721/55/20241535.
[27] M. Riansyah, S. Suwilo, and M. Zarlis, Improved accuracy in data mining decision tree classification using adaptive boosting, SinkrOn, 8 (2), 617–622, 2023, https://doi.org/10.33395/sinkron.v8i2.12055.
[28] A. AlMohimeed, H. Saleh, S. Mostafa, R. M. A. Saad, and A. S. Talaat, Cervical cancer diagnosis using stacked ensemble model and optimized feature selection: an explainable artificial intelligence approach, Computers, 12 (10), 200, 2023, https://doi.org/10.3390/computers12100200.
[29] S. Imangaliyev, J. Schlötterer, F. Meyer, and C. Seifert, Diagnosis of inflammatory bowel disease and colorectal cancer through multi-view stacked generalization applied on gut microbiome data, Diagnostics, 12 (10), 2514, 2022. https://doi.org/10.3390/diagnostics12102514.
[30] M. Hasanah, R. A. Putri, M. A. R. Putra, and T. Ahmad, Analysis of Weight-Based Voting Classifier for Intrusion Detection System, International Journal of Intelligent Engineering and Systems, 17 (2), 190–200. 2024, https://doi.org/10.22266/ijies2024.0430.17.
[31] B. Fieri and D. Suhartono, Offensive language detection using soft voting ensemble model, Mendel, 29 (1), 1–6, 2023. https://doi.org/10.13164/mendel.2023.1.001.
[32] O. Octavian, A. Badruzzaman, Muhammand Yusuf Ridho, and B. D. Trisedya, Enhancing Weighted Averaging for CNN Model Ensemble in Plant Diseases Image Classification, Jurnal Resti, 8 (2), 272–279, 2024, https://doi.org/10.29207/resti.v8i2.5669.
[33] B. Hasan, Zubair, S. A. Shaikh, A. Khaliq, and G. Nadeem, Data-Driven decision-making: accurate customer churn prediction with Cat-Boost, The Asian Bulletin of Big Data Management, 4 (02), 2024. https://doi.org/10.62019/abbdm.v4i02.175.
[34] T. Suresh, T. A. Assegie, S. Ganesan, R. L. Tulasi, R. Mothukuri, and A. O. Salau, Explainable extreme

boosting model for breast cancer diagnosis, International Journal of Electrical and Computer Engineering, 13(5), 5764, 2023. https://doi.org/10.11591/ijece.v13i5.pp5764-5769.

[35] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186, Minnesota, USA, 2-7 June 2019.

[36] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, RoBERTa: A robustly optimized BERT pretraining approach, arXiv preprint arXiv:1907.11692, July 26, 2019. https://doi.org/10.48550/arXiv.1907.11692

[37] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108, October 2, 2019. https://doi.org/10.48550/arXiv.1910.01108

[38] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, ALBERT: A Lite BERT for self-supervised learning of language representations, arXiv preprint arXiv:1909.11942, September 26, 2019. https://doi.org/10.48550/arXiv.1909.11942