Uludağ University Journal of The Faculty of Engineering, Vol. 31, No. 2, 2026 RESEARCH

DOI: 10.17482/uumfd.1779769

OPTIMIZING MACHINE LEARNING CLASSIFIERS FOR HIGH-ACCURACY SENTIMENT DETECTION IN THE TURKISH LANGUAGE

Ahmad BWIDANI [*]
Ali KARAH BASH [**]

Received: 01.10.2025; revised: 04.04.2026; accepted: 12.04.2026

Abstract: Sentiment analysis is widely used to extract opinions from textual data; however, its application to morphologically rich languages such as Turkish remains challenging. This study investigates the optimization of classical machine learning classifiers and ensemble learning strategies for binary Turkish sentiment analysis under a unified experimental framework. Several ML models are trained on a balanced dataset of user reviews, including Linear Support Vector Machine Classifier (LSVMC), Multinomial Naïve Bayes (MNB), and Logistic Regression (LR). Their outputs were further combined using Ensemble Learning (EL) models, namely Majority Voting (MVEL) and Stacking (SEL). Results demonstrate that the SEL Classifier outperforms all examined models, achieving 92.80% accuracy at the cost of increased computational complexity. Among the examined individual models, LSVMC (92%), MNB (92%), and LR (92%) had the best accuracy. While the study does not aim to achieve state-of-the-art performance with deep or transformer-based architectures, the results demonstrate that optimized classical models remain highly effective in Turkish SA, and their accuracy can be further improved with EL mechanisms.

Keywords: Machine Learning, Ensemble Learning, Turkish Language Processing, Turkish Sentiment Analysis

Türkçede Yüksek Doğruluklu Duygu Tespiti için Makine Öğrenmesi Sınıflandırıcılarının Optimizasyonu

Özet: Duygu analizi, metinsel verilerden görüşleri ortaya çıkarmak için yaygın olarak kullanılmaktadır; ancak Türkçe gibi morfolojik açıdan zengin dillerdeki uygulamaları zorluğunu korumaktadır. Bu çalışma, birleşik bir deneysel çerçeve altında, ikili (binary) Türkçe duygu analizi için klasik makine öğrenmesi sınıflandırıcılarının ve topluluk öğrenme (ensemble learning) stratejilerinin optimizasyonunu incelemektedir. Dengeli bir kullanıcı yorumları veri seti üzerinde, doğrusal destek vektör makinesi sınıflandırıcısı (LSVMC), çok terimli Naïve Bayes (MNB) ve lojistik regresyon (LR) dahil olmak üzere çeşitli makine öğrenmesi modelleri eğitilmiştir. Bu modellerin çıktıları, çoğunluk oylaması (MVEL) ve yığınlama (SEL) gibi topluluk öğrenme yöntemleri kullanılarak birleştirilmiştir. Sonuçlar, SEL sınıflandırıcısının artan hesaplama karmaşıklığı pahasına %92,80 doğruluk oranına ulaşarak incelenen tüm modellerden daha iyi performans gösterdiğini ortaya koymaktadır. İncelenen bireysel modeller arasında en yüksek doğruluk oranlarını LSVMC (%92), MNB (%92) ve LR (%92) sergilemiştir. Çalışma, derin öğrenme veya transformatör tabanlı mimariler aracılığıyla literatürdeki en ileri düzey (state-of-the-art) performansı geride bırakmayı hedeflemese de, sonuçlar, optimize edilmiş klasik modellerin Türkçe duygu analizinde halen oldukça başarılı olduğunu ve performanslarının topluluk öğrenme mekanizmalarıyla daha da geliştirilebileceğini kanıtlamaktadır.
Anahtar Kelimeler: Makine Öğrenmesi, Topluluk Öğrenmesi, Türkçe Dil İşleme, Türkçe Duygu Analizi.

[*] Department of Electronic and Computer Engineering, Faculty of Engineering, Hasan Kalyoncu University, 27000 Şehitkamil, Gaziantep, Türkiye.
[**] Department of Electrical and Electronic Engineering, Faculty of Engineering, Hasan Kalyoncu University, 27000 Şehitkamil, Gaziantep, Türkiye.

Correspondence Author: Ahmad BWIDANI (ahmad2135@gmail.com)

NOMENCLATURE

Term	Description	Term	Description
SA	Sentiment Analysis	**LR**	Logistic Regression
LSVMC	Linear Support Vector Machine Classifier	**GBC**	Gradient Boosting Classifier
DTC	Decision Tree Classifier	**ABC**	Adaboost Classifier
TF-IDF	Term Frequency - Inverse Document Frequency	**RF**	Random Forest
KNNC	K-Nearest Neighbors classifier	**SVM**	Support Vector Machine
EL	Ensemble Learning	**DL**	Deep Learning
ML	Machine Learning	**LSTM**	Long Short-Term Memory
MNB	Multinomial Naïve Bayes	**MVEL**	Majority Voting Ensemble Learning
SEL	Stacking Ensemble Learning	**NB**	Naïve Bayes
SGDC	Stochastic Gradient Descent Classifier	**BNB**	Bernoulli Naïve Bayes

1. INTRODUCTION

With the development of internet services, people's use, reliance, and interaction with them in various fields have increased, providing them with the opportunity to express their opinions and share their positive or negative impressions of these services. Users' textual feedback is important to many parties involved in the user experience, such as e-commerce and hotel service companies. SA is a series of methods used to detect and extract subjective information, such as opinions and attitudes, from language (Mäntylä et al, 2018). It commonly concentrates on classifying the text describing the main sense in the dataset, e.g., Negative, Positive, and Neutral. This classification aims to determine the writer/speaker's intent and the subject of their feelings.

Various methodological approaches have been proposed for automated SA, including lexicon-based methods, classical ML models, DL architectures, and, more recently, transformer-based language models. While recent research increasingly emphasizes deep and pretrained models, classical ML approaches remain attractive due to their interpretability, lower computational requirements, and competitive performance on structured and moderate-to-large datasets. However, systematic evaluations of optimized classical classifiers and ensemble learning methods under standardized experimental settings for Turkish sentiment analysis remain relatively limited.

1.1. Literature Review

Kiziltepe et al. (2024) achieved 96% accuracy by training SVMC on a dataset of 244,866 Turkish-language reviews. Moreover, Ucan et al. (2016) proposed an automatic translation approach to generate a sentiment lexicon for Turkish, achieving an accuracy of nearly 80%. They used the hotels-and-movie-reviews dataset, which contains 65000 records. The researchers Erşahin et al. (2019) proposed a hybrid model combining lexicon-based and learning-based approaches to predict the polarity of Turkish text, increasing average accuracy by 7% to nearly 92% on a hotel reviews dataset. Additionally, researchers Gifari et al. (2021) proposed a

stacking EL model to detect the polarity of English movie reviews. They used NB, KNNC, and LR as estimators, with LR as the meta-learner. They obtained an accuracy of 89.40%, increasing the best estimator's accuracy by 0.71%. Rumelli et al. (2019) followed a hybrid model combining lexicon-based SA and an ML model to detect the polarity of Turkish text. They achieved an accuracy of 73%. Demirci et al. (2019) compared ML and deep learning SA models on a Turkish-language Twitter dataset. DL method with an accuracy of (81.86%) superseded the RF with the tf-idf method (81.74%).

Subba et al. (2022) proposed a heterogeneous EL model of different RNN base learners: LSTM, GRU, and bi-GRU alongside an LSTM meta learner. The best resultant accuracy was 92% on the IMDb reviews dataset. Karagoz et al. (2019) proposed an aspect-based SA model of Turkish text. The model aims to differentiate among multiple aspects within a single text rather than providing an overall polarity score for the text. They used two phases: one to extract the aspects, and the other to match the aspects with the sentiment words in the text. They obtained an F1-score of 85% in aspect-sentiment matching on a relatively small, manually annotated dataset of 1000 sentences, which opens another track of SA for future studies.

Armaeni et al. (2024) achieved 75% accuracy when training a decision tree model on Indonesian comments on the YouTube video-sharing platform. Habberrih et al. (2024) trained and evaluated SVM and LR models on Arabic SA, achieving 72% accuracy with SVM. They also reported that using SVM without stop-word removal achieved 1% higher accuracy.

Demircan et al. (2021) compared several ML models in Turkish-language SA. They've trained the models on a dataset of user reviews, based on user scoring, to classify the text polarity as positive, negative, or neutral. They found that the results of the SVM- and RF-based models are consistently the highest. The SVM model correctly predicted 9420 labels out of 10480, yielding 90% accuracy. The RF model correctly predicted 8929 labels, yielding an accuracy of 85%. It is worth noting that their dataset was unbalanced, with about 61% of the records labeled positively, 15% neutrally, and 24% negatively. That imbalance led to their models being biased toward the positive class.

Savci et al. (2023) compared several ML and DL models across English, Arabic, and Turkish. Results show that the SVM and RF models achieve the highest, or very close to the highest, results across all 3 languages. The accuracy of the SVM and RF models was 91% and 86% for Turkish, 91.3% and 79% for Arabic, and 86.7% and 87.5% for English, respectively. Oliko et al. (2025) proposed an MNB Lexicon Pooled Ensemble model with an accuracy of 82.5%, outperforming individual MNB by 0.1%. Wijati et al. (2024) trained a KNN classifier on a new dataset of Indonesian-language application reviews, achieving an accuracy of 82%. Okereke et al. (2024) compared several ML models by training them on a dataset of 50000 English-language movie reviews, and achieved 91% accuracy with both LSVMC and LR. They also achieved a close result with the SGDC, achieving 90% accuracy.

Başarslan et al. (2023) compared NB, SVM, LR, and KNNC models and applied Stacking and Majority Voting EL models on all four models. The models were trained on two datasets: one with 20,000 records and the other with over 480,000 records. They also compared 2 vectorizer algorithms: Word2Vec and TF-IDF. Their results showed that Word2Vec outperformed the TF-IDF in terms of accuracy. Additionally, Majority Voting EL achieved higher accuracy than Stacking EL. The best result was Majority Voting on the large dataset, achieving 88.7% accuracy with the Word2Vec vectorizer.

Iyer (2024) compared several ML and DL models on an English dataset with 1.6 million records. They stated that LR, with an accuracy of 78%, achieved the best overall accuracy and stability among all models in the study, including LSTM and Bi-LSTM. Aydoğan et al. (2023) tested several ML and DL models on the publicly available Turkish Sentiment Analysis Version 1 (TRSAv1) dataset, which contains 150,000 records. The results show that the NB model with N-Gram achieved 82% accuracy, outperforming SVM, LR, DT, RT, and Voting Classifier ML models.

Danyal et al. (2024) compared four NB model types trained on two movie review datasets: the IMDb Movie Reviews Dataset and the Rotten Tomatoes Movie Reviews Dataset. They also compared the TF-IDF vectorizer with the count vectorizer in all experiments. On the IMDB dataset, BNB achieved the highest accuracy of 85.01% with the count vectorizer, whereas MNB achieved 85.77% with TF–IDF. On the Rotten Tomatoes dataset, BNB achieved maximum accuracies of 76.22% with TF–IDF and 76.37% with the count vectorizer. Özmen et al. (2025) manually constructed a domain-specific sentiment dictionary, which was then used to automatically label 875,455 cosmetic product reviews in Turkish. These reviews are used to train several ML models; all the examined models are superseded by their SVM model, which achieves 93% accuracy.

Zorarpaci (2024) trained several models on a Turkish dataset of 3600 records. They aimed to classify the text into 6 different classes based on the aspects. Results showed that the voting ensemble of RF, ADB, Bagging, and MNB achieved the highest accuracy, at 93.8%. Guven (2021) compared several ML models with the BERT DL models on a pre-labeled Twitter dataset with 5 emotion labels. Results showed that the BERT model, with 98.75% accuracy, outperformed the most performant ML model, LR, which achieved 98.4% accuracy.

Collectively, existing studies demonstrate that classical ML models—particularly SVM, LR, and NB—remain competitive baselines for Turkish sentiment classification. Reported performance variations are strongly influenced by dataset characteristics, preprocessing, feature representation strategies, and evaluation methodologies. Ensemble methods often provide incremental gains that depend on dataset scale and model complementarity, while DL approaches, although powerful, do not consistently and efficiently outperform well-optimized classical models, given the computational complexity trade-off. However, cross-study comparisons of classical models remain difficult due to variations in datasets, class balance, preprocessing, feature extraction, and tuning procedures. In many cases, optimization processes are not described in detail, and experimental steps differ substantially. Therefore, a controlled comparative evaluation of systematically optimized classical and ensemble learning models within a unified experimental framework is necessary to better understand their relative effectiveness and computational trade-offs for Turkish sentiment analysis.

1.2. Contributions

- **Competitive accuracy across the literature**

While previous studies on the same dataset have utilized a wrapper-based greedy approach (TSA) (Sağbaş, 2024), BERT-based architectures (Özçift, 2021), pure classical models (Kılınç, 2019), and translation-based models (Yıldırım, 2020), this research represents the first application of ensemble learning using classical models on this specific dataset, to the best of the authors' knowledge. The achieved 92.8% accuracy not only sets a new benchmark for this data but also stands among the top results for classical models in the existing literature. Table 19 provides a comparative overview of these studies alongside this work, organized by accuracy.

- **Optimization of classical models**

Several widely used classical ML classifiers—including SVM, LR, NB, and KNN—were trained within a unified preprocessing and feature extraction environment. This environment ensures a fair comparison and maintains controlled evaluation conditions across all models.

- **Comparative Ensemble Evaluation**

Majority Voting and Stacking-based ensemble learning approaches are implemented and analysed to determine whether ensemble strategies consistently improve performance over individually optimized base models.

- **Controlled experimental environment**

This study applies a standardized pipeline for feature extraction and dataset characteristics, reducing experimental variability and enabling clearer interpretation and reproducibility of results.

- **Evidence of the Validity of Ensemble Models in Morphologically Rich Languages**

The findings provide evidence on the effectiveness and stability of classical and ensemble ML approaches for SA in Turkish, which is a good example of a morphologically rich, agglutinative language, without incurring the costs of higher-complexity models, such as transfer-based models.

2. MATERIALS AND METHODS

This research compares several ML models, namely SVMC, MNB, LR, KNNC, GBC, AdaBoost Classifier, and DTC, by training them on a mid-sized dataset of hotel and film reviews. The dataset has undergone several preprocessing steps, including removing punctuation and numbers, balancing the data, and performing TF-IDF feature extraction. All trained models are fed into heterogeneous Stacking and Majority Voting EL models to achieve the best accuracy. This section presents a step-by-step representation of the experiments conducted on the dataset to derive the findings. Figure 1 summarizes the overall algorithmic workflow followed in this work to obtain all outcomes.

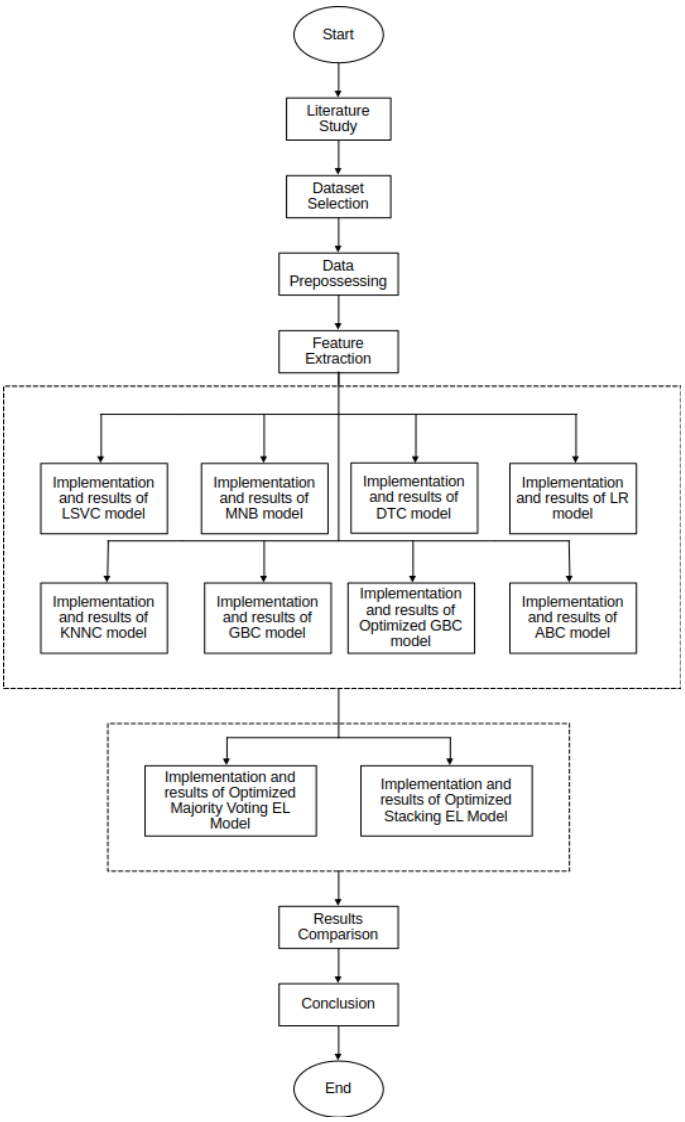

Figure 1:
Research flow

2.1. System Prerequisites

The models are trained on a local Linux machine with Python 3.12, 32GB of RAM, and an Intel Core i7 9700K processor. It's preferable to have high storage speed to accelerate model training. SSD storage with 20 GB of free space was used.

2.2. Dataset

The field of SA draws from a wide range of potential data sources. Given the nature of this research, a precisely annotated dataset—ideally labeled by native Turkish speakers—is essential for ensuring high accuracy during both the training and evaluation phases. The dataset used in this study is well known in Turkish SA literature as HUMIR; it comprises approximately 65,000 records of hotel and product user reviews, with both negative and positive polarity (Ucan et al, 2016). The dataset has two resources of widely used movie and hotel websites. One of which has a 1 to 5 user rating, while the other has a user rating of 0 to 100. Binary labeling was applied based on these scores: reviews with ratings of 1–2 (out of 5) or 0–40 (out of 100) were classified as negative. Conversely, reviews with ratings of 4–5 (out of 5) or 80–100 (out of 100) were classified as positive (Ucan et al, 2016). Table 1 presents a sample of the dataset used in this research.

Table 1. Dataset sample

Clean comment	Category
Fiyat ve kalite arasında uçurum var.	Negative
Rezaletti. Çok pisti.	Negative
Yemeklerde çeşit az ama lezzet iyiydi.	Negative
Tek çeşit ana yemek çıktı. Yarım pansiyon dediler.	Negative
Saydığım nedenlerden dolayı tavsiye etmem tesisi.	Negative
tek kelimeyle harikaaaaa	Positive
film enteresan bence güzeldi	Positive
russell crowe ve yine muhteşem ..	Positive
3-4 kez izlemişimdir hala da izlerim süper bir animasyon 10/10	Positive

2.3. Data Preprocessing

The data preprocessing phase aims to find a more generalized form of data to reduce outliers. This significantly improves the accuracy and generalization of the used model.

2.3.1. Drop empty records

Dropping empty records is an essential step in the data preprocessing. This step removes empty records that interfere with the ML models during training.

2.3.2. Convert category to numerical indicators

This step is also important, as the textual Negative/Positive flags are converted to numerical flags. Thus, the Negative category is represented by -1 and the Positive category by 1. Table 2 displays a sample of the dataset after this preprocessing step.

Table 2. Category renaming example

Clean comment	Category
Fiyat ve kalite arasında uçurum var.	-1
Russell Crowe ve yine muhteşem.	1

2.3.3. Turkish Sentence Normalization

Turkish words involve distinct letters from English words. However, some keyboards may not support Turkish letters, even though Turkish people use them, resulting in Turkish words being written incorrectly and misleading the models during learning and testing. The Zemberek library, written in Python, normalizes Turkish words using various NLP techniques. Table 3 presents a sample of Turkish word normalization.

Table 3. Turkish sentence normalization example

Raw Text	Normalized Text
gitmeyinturk musteriyi gorup sirtini donen bir patron ortalarda ingilizlerle ortalarda dolasan bir menejer iscilerin bile …	gitmeyintürk müşteriyi görüp sırtını dönen bir patron ortalarda ingilizlerle ortalarda dolaşan bir menejer işçilerin bile …

2.3.4. Remove punctuation, numbers, and special characters

Punctuation and special characters might be good examples of potential outliers for every ML model. Numbers are treated differently depending on the dataset or the ML model: quantities, flight numbers, or percentages. Since SA numbers often represent meaningless data, they are removed from the dataset during preprocessing. Table 4 shows a sentence before and after removal.

Table 4. Sample of punctuation, numbers, and special characters removal.

Raw Text	Pre-processed Text
3-4 kez izlemişimdir hala da izlerim süper bir animasyon 10/10	kez izlemişimdir hala da izlerim süper bir animasyon

2.3.5. Remove duplicates

Duplicates do not add to the training and testing stages; thus, they are simply removed from the dataset during preprocessing.

2.3.6. Balance data

Another factor that might affect the model's accuracy is data imbalance. Having data with more negative/positive polarity might lead to a positively/ negatively biased model, leading to class-specific low accuracy. To balance the data, some records with specific polarity are removed from the dataset to ensure equal numbers of negative/ positive records. The process dropped 1173 records, resulting in a balanced dataset of 63827 records. Figure 2 shows the distribution of the data after the balancing process.

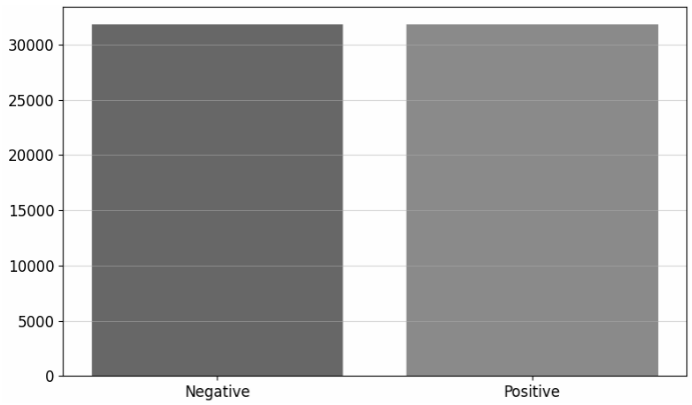

Figure 2:
Balanced data distribution

2.3.7. TF-IDF Vectorization

The TF-IDF technique was used with a mix of 1-grams, 2-grams, and 3-grams. The TF-IDF technique is based on the idea of emphasizing the most significant terms during training while ignoring less significant tokens, such as unimportant stop words, that do not contribute to the text's polarity. N-grams stand for combining more than one token/word in the token evaluation process. In this research, ML models 1, 2, and 3-grams are used across the entire document. Table 5 lists the set of parameters used in TF-IDF feature extraction.

Table 5. TF-IDF parameters

Parameter	Value	Description
stop_words	None	Specifies the norm used in penalization
analyzer	word	The token used in vocabulary building
ngram_range	(1, 3)	1, 2, and 3 grams
max_df	1.0 (100%)	Do not ignore terms using a high threshold
min_df	1	Ignore terms having df < 1
max_features	None	All features are used
norm	l2	Specifies the norm used
use_idf	true	enable inverse document frequency
sublinear_tf	false	apply sublinear tf scaling

The idea of using TF-IDF with n-grams is to capture the context of a word, in case important contextual cues are lost during other preprocessing steps. Table 6 shows examples of 1-, 2-, and 3-gram feature extraction for 2 records.

Table 6. 1, 2, and 3-grams feature extraction example

Input Data	Result Features (comma-separated)
Tek çeşit ana yemek çıktı. Yarım pansiyon dediler.	ana,ana yemek,ana yemek çıktı,dediler,pansiyon,pansiyon dediler,tek,tek çeşit,tek çeşit ana,yarım,yarım pansiyon,yarım pansiyon dediler,yemek,yemek çıktı,yemek çıktı yarım,çeşit,çeşit ana,çeşit ana yemek,çıktı,çıktı yarım,çıktı yarım pansiyon,ama,ama lezzet,ama lezzet iyiydi,az,az ama,az ama lezzet,iyiydi,lezzet,lezzet iyiydi,yemeklerde,yemeklerde çeşit,yemeklerde çeşit az,çeşit az,çeşit az ama
Yemeklerde çeşit az ama lezzet iyiydi.	

It is worth noting that there's no step to remove stop words depending on stop-word dictionaries in this research. Stop-word removal resulted in a 1% decrease in accuracy, as noted for Arabic-language models (Habberrih, 2024; Karah Bash et al., 2024). On the other hand, TF-IDF vectorization could compensate for stop-word removal more efficiently, as it considers stop words in context and through n-gram tokenization, then assigns them relevant weights, mostly the lowest, depending on their importance. Table 7 shows a sample of term weights for some records and how the weights of stop words diminish compared to normal words in the TF-IDF index.

Table 7. TF-IDF weights example

Doc	ama	aynı	az	izlemişimdir	müthiş
1	0.04	0.10	0.07	0	0
2	0.09	0	0	0	0
3	0	0	0.16	0.54	0
4	0.04	0.06	0	0	0.21
5	0.03	0.09	0	0	0
6	0	0	0.04	0	0
7	0.04	0.05	0	0	0
8	0.17	0	0.05	0	0
9	0.08	0	0.06	0	0
10	0	0	0	0	0.46

2.3.8. Split into training and testing sets

To evaluate the ML models, the dataset is split into 75% for training and 25% for testing. All the above preprocessing steps are applied independently to both the training and testing sets to prevent data leakage. The only exception was the filtering of short or long records; this step was omitted during the evaluation phase to ensure a fair and unbiased assessment of the models' performance on the original test distribution.

2.3.9. Drop long/short records

Both long and short text records may exhibit ambiguous sentiment polarity, posing challenges for accurate classification. Moreover, excessively long records can result in high-dimensional feature spaces, which increases the risk of overfitting during model training. On the other hand, there might be a good number of these long/short records. Therefore, this step is only necessary for the training set, so it's not applied to the testing set to preserve the trustworthiness of the evaluation metrics. A specific criterion is applied to remove the short and long by removing long records that have more than 10 times the average length of the corpus, and short records that have less than the average corpus length divided by 10. Applying this criterion resulted in dropping 1647 records from 47685, about 3% of the entire training dataset.

2.4. Training Models

To facilitate a robust comparison, several models were trained on the same dataset using a 75% training and 25% testing split. Optimization of the classical models was performed via the Grid Search methodology. These optimizations were conducted independently of the final evaluation to maintain a controlled experimental environment and prevent data leakage. Furthermore, certain base models were utilized as components within more complex ensemble architectures to achieve optimal performance. This research employs five single-learner models (LinearSVC, MNB, DTC, LR, and KNN) and four ensemble models (AdaBoost, GB, Majority Voting, and Stacking).

AdaBoost and GB are categorized as homogeneous ensemble models (typically based on DT estimators), whereas Majority Voting and Stacking are heterogeneous ensemble models that combine different types of base estimators. This section details all models and their corresponding training parameters, as illustrated in Figure 3.

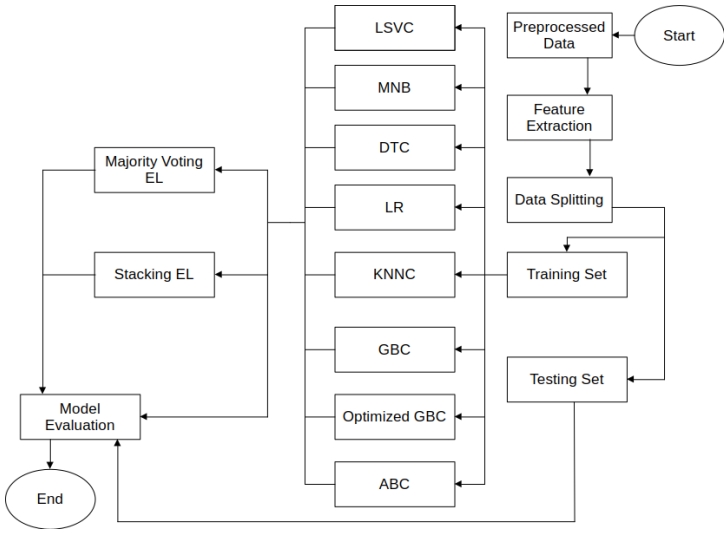

Figure 3:
Training and testing workflow of the ML models

2.4.1. Linear Support Vector Machine Classification (LSVMC)

The SVM is a supervised algorithm, a simple yet effective method primarily used in text classification. The training data labeled with the created vector model are classified. A boundary line called the hyperplane separates the groups. The goal is to find the hyperplane that creates the greatest separation between the classes (Savci et al, 2023). This research uses the LSVMC implementation in the scikit-learn library for Python, a widely used ML library. The LSVMC model was trained using optimized parameters. The most important parameters are listed in Table 8.

Table 8. LSVMC parameters

Parameter	Value	Description
penalty	L2	Specifies the norm used in penalization
loss	hinge	Specifies the loss function
tol	0.0004	Tolerance for stopping criteria.
C	10.0	Regularization parameter
max_iter	1000	The maximum number of iterations to be run.

2.4.2. Multinomial Naïve Bayes (MNB)

The MNB is the application of Bayes' theorem to predict which category a sample belongs to, with the strong assumption that each feature is independent of the others. This algorithm is a probabilistic classifier that computes the probability of each category using Bayes' theorem and outputs the category with the highest probability for a given sample. The scikit-learn implementation of the MNB model is utilized. The MNB model is trained using optimized parameters. Table 9 lists the parameters and their descriptions.

Table 9. MNB parameters

Parameter	Value	Description
alpha	0.1	All classes are considered (reduces fitting)
fit_prior	false	Class priors are estimated from training data

2.4.3. Decision Tree Classifier (DTC)

The DTC is trained using the Scikit-learn Python library. Table 10 presents the optimized model parameters along with their descriptions.

Table 10. DTC parameters

Parameter	Value	Description
criterion	gini	The split gain/loss algorithm.
splitter	best	Choose the best split among all options. (not among random list)

max_depth	1,000,000	Maximum depth limit for splits
min_samples_split	2	The minimum subtree size to proceed with the split
min_samples_leaf	2	Absolute value of the total sample size
min_weight_fraction_leaf	0.001	Ratio of the sum of the total weights
max_features	None	Splits of all feature sizes are considered
max_leaf_nodes	None	No maximum leaves (tree width) is specified
min_impurity_decrease	0	All splits will be considered regardless of the gain/loss
ccp_alpha	0	No pruning
monotonic_cst	None	No feature-specific monotonic constraints
criterion	gini	The split gain/loss algorithm.
splitter	best	Choose the best split among all options. (not among random list)

2.4.4. Logistic Regression (LR)

The LR is a statistical method used to analyze a data set with one or more independent variables that determine the result. The result is measured by a binary variable (Başarslan, 2023). This research utilizes the LR model from the Scikit-Learn library. The model is trained using the optimized parameter settings detailed in Table 11.

Table 11. LR parameters

Parameter	Value	Description
penalty	l2	Specify the norm of the penalty.
dual	false	Specify whether the solver equation is regularized
tol	0.0004	Tolerance for stopping criteria
C	10.0	Regularization parameter
fit_intercept	true	Add a constant to the decision function for uncentered data
intercept_scaling	1	Scale of the intercept value. (not used as the solver is not liblinear)
class_weight	None	All classes are supposed to have equal weight
solver	lbfgs	Optimizer algorithm
max_iter	100	The maximum number of iterations for the solver to converge

2.4.5. K-Nearest Neighbors classifier (KNNC)

The KNN method is a common pattern recognition technique often chosen for categorizing data groups due to its very simple approach. This algorithm classifies unlabeled data samples using the majority label of their nearest neighbors in the training dataset. The probability of merging documents into a single unit increases when documents are similar, but decreases when they are different (Wijati, 2024). In this study, the KNNC implementation in the Scikit-Learn library is used. Table 12 presents the optimized training parameters and their descriptions.

Table 12. KNNC parameters

Parameter	Value	Description
n_neighbors	200	The number of k-neighbors used in the training process.
algorithm	auto (ball_tree)	The implementation chooses the algorithm here depending on the data and metric method
leaf_size	30	The leaf size for ball_tree or kd_tree algorithms
p	2	The power of the distance metric
metric	Minkowski	The distance metric used
weights	uniform	All points in each neighborhood are weighted equally.

2.4.6. Gradient Boosting Classifier (GBC)

This research includes two homogeneous boosting EL algorithms with a homogeneous base learner set (estimators): GBC and ADB. The GBC model is trained with the optimized parameters shown in Table 13.

Table 13. GBC parameters

Parameter	Value	Description
loss	log_loss	The loss function
learning_rate	0.2	The contribution of each tree
n_estimators	200	The number of boosting stages
subsample	1.0	The number of samples used for fitting the individual base learners
criterion	friedman_mse	The function to measure the quality of the split. mse/friedman_mse. (friedman_mse) mean square error with improvement score by Friedman)
min_samples_split	2	Minimum number of samples required to split a node
min_samples_leaf	1	Minimum number of samples to be in a leaf node
min_weight_fraction_leaf	0	The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node.
min_impurity_decrease	0	All splits will be considered regardless of the gain/loss
max_depth	3	Max depth of each estimator
init	Dummy estimator	An estimator computing initial predictions
max_features	(None)	The number of features to consider when looking for best split
max_leaf_nodes	None (unlimited)	The maximum number of leaf nodes
n_iter_no_change	None (no early stopping)	Number of allowed iterations without improvement of results

ccp_alpha	0 (no pruning)	The complexity parameter used in Minimal cost complexity pruning (ccp_alpha)

2.4.7. Gradient Boosting Classifier - Optimized version

The optimized version of the GBC in the XGBoost Python library is used to increase the number of estimators and the tree depth, while keeping the model efficient. It's trained with the same parameters as described in Table 13, except that n_estimators are set to 400, max_depth set to 6 and learning_rate set to 0.1 to obtain better results. The mentioned parameters are presented in Table 14.

Table 14. Optimized GBC parameters

Parameter	Value	Description
n_estimators	400	The number of boosting stages
max_depth	6	Max depth of each estimator
learning_rate	0.1	The contribution of each tree

2.4.8. Adaboost Classifier (ABC)

The ABC model from the Scikit-learn library is employed and trained using the optimized parameter configuration provided in Table 15.

Table 15. ABC parameters

Parameter	Value	Description
estimator	decision tree estimator (None)	Specify the base estimator from which the boosted ensemble is built.
n_estimators	100	Specify the maximum number of base estimators at which the boosting is terminated
learning_rate	1.0	A weight value specifying the contribution of each estimator at each boosting iteration
algorithm	SAMME	use the SAMME discrete boosting algorithm

2.4.9. Majority Voting Ensemble Classifier

This research includes two heterogeneous EL models that are based on heterogeneous estimators. The first one is the Majority Voting EL model, which chooses the class with the majority vote among all learners. The second one is Stacking EL, which will be discussed in the next section. The Majority Voting ensemble was implemented using the voting classifier from the Scikit-Learn library, with the parameters shown in Table 16. The estimator's parameter included the set of learners examined in this study: LR, MNB, LSVMC, DTC, KNNC, GBC, Optimized GBC, and ABC. Each of these learners was incorporated with the same parameter settings used during their training.

Table 16. Majority voting ensemble classifier parameters

Parameter	Value	Description
estimators	A list of the previous models	The list of estimators used for voting.
voting	Hard	Use the count of class labels instead of averaging the predicted probabilities of each model
weights	None	All estimators are equally weighted

2.4.10. Stacking Ensemble Classifier

The stacking EL classifier is based on heterogeneous estimators, just like the majority voting model. The key difference between them is that the stacking model uses a meta-model to learn from the results of the estimator models. The Stacking Classifier from the Scikit-Learn library was used and trained with the parameters presented in Table 17. Like the majority voting model, the estimator's parameter was set to include the learners evaluated in this research: LR, MNB, LSVMC, DTC, KNNC, GBC, Optimized GBC, and ABC. Each estimator was trained with the same parameters used when trained individually.

Table 17. Stacking ensemble classifier parameters

Parameter	Value	Description
Estimators	A list of the previous models	Base estimators which will be stacked together
final_estimator	Logistic Regression	A classifier that is used to combine base estimators
Cv	5-fold cross validation	Determines the cross-validation splitting strategy

3. RESULTS AND DISCUSSION

3.1. Results

It’s noted that the stacking model achieves the best result, with an accuracy of 92.80%. The full list of the examined models' result metrics is given in Table 18, sorted by best accuracy. Among the models evaluated in this study, MNB, LSVMC, and LR achieved the highest performance. MNB achieved strong results with low training time, benefiting from low computational requirements. The Majority Vote Ensemble Classifier produced accuracy comparable to MNB; however, the marginal performance gains did not sufficiently compensate for the increased computational complexity. The K-Nearest Neighbors classifier achieved moderate accuracy. In particular, the optimized Gradient Boosting Classifier outperformed its unoptimized variant and AdaBoost, showing a 1% improvement in accuracy. Conversely, the Decision Tree Classifier model yielded less competitive results. In conclusion, the Stacking Ensemble Classifier achieved the highest accuracy among all the models examined, but its performance advantage over MNB was minimal and was accompanied by significantly longer training times.

Table 18. Proposed models' metrics sorted by best accuracy

Model	Fit time	ACC	F1	PRS	REC
Stacking EL	14:25:31	0.93	0.93	0.93	0.93
MNB	0:00:01	0.92	0.92	0.92	0.92
LSVMC	0:00:17	0.92	0.92	0.92	0.92
LR	0:00:06	0.92	0.92	0.92	0.92
Majority Voting EL	0:32:36	0.91	0.91	0.91	0.91
KNN	0:00:01	0.88	0.88	0.89	0.88
Opt. GBC	0:06:53	0.88	0.88	0.88	0.88
GBC	0:16:39	0.87	0.87	0.87	0.87
ABC	0:05:05	0.87	0.87	0.87	0.87
DTC	0:01:16	0.80	0.80	0.80	0.80

ACC: Accuracy, F1: F1-Score, PRS: Precision, REC: Recall

This research's results show that the MNB model has the best cost-effective performance. Its relatively short training time and high accuracy make it the most reliable model for mid-sized datasets. Results in the literature also show that this model is highly performant, even though it's been superseded by more complicated models like EL, LLMs, and DL models. Table 19 compares the examined SEL model with other results in the literature.

3.2. Comparison with the literature studies

Results indicate that the model examined by this study performs competitively with state-of-the-art approaches identified in the literature. Notably, within the specific scope of polarity classification. The voting EL model proposed by Zorarpaci (2024) achieved better results in aspect analysis, with an accuracy of nearly 94%. The SVM model proposed by Kiziltepe et al. (2024) achieved the best polarity classification accuracy of 96%. Guven (2021) achieved the highest accuracy in multi-class sentiment analysis, reaching 99%.

Comparing the results of studies on the same dataset, labeled as Humir in Table 19, shows that the examined model's accuracy is relatively close to that of the state-of-the-art model used by Özçift et al. (2021), which requires high computational complexity. Sağbaş (2024) used a two-stage greedy approach with MNB across multiple datasets, including HUMIR, achieving better performance than in our study. However, Stacking Ensemble learning reduces potential data leakage during the training phase. On the other hand, the classical models' performance in this study is slightly better than Kılınç's (2019). Their model achieved 87.02% accuracy using SVM.

Table 19. Models' metrics in literature and this study sorted by best accuracy

Author	Model	Classes	Language	Data topics	# of records	Accuracy
Guven (2021)	BERT	5 (emotions)	Turkish	social	4,500	98.75%
Özçift et al. (2021)	ELECTRA-Tr, BERT-Tr	2 (polarity)	Turkish	HUMIR + reviews	~100,000	98.38%
Kiziltepe et al. (2024)	SVM	2 (polarity)	Turkish	e-commerce	244,866	96.31%

Sağbaş (2024)	MNB (TSA Wrapper)	2 (polarity)	Turkish	user reviews + HUMIR	18,000	95.57%
Zorarpaci (2024)	Voting EL	6 (aspects)	Turkish	news	3,600	93.80%
Özmen et al. (2025)	SVM	3 (polarity)	Turkish	cosmetics	875,455	93.00%
This Study	Stacking EL	2 (polarity)	Turkish	hotels + movie (HUMIR)	65,000	92.80%
Subba et al. (2022)	Stacking EL	2 (polarity)	English	movie	50,000	92.00%
Erşahin et al. (2019)	SVM & Lexicon	3 (polarity)	Turkish	hotels	11,600	91.96%
Savci et al. (2023)	SVM	3 (polarity)	Ar, En, Tr	e-commerce	300,000	91.30%
Okereke et al. (2024)	LSVMC	3 (polarity)	English	movie	50,000	91.20%
Demircan et al. (2021)	SVM	3 (polarity)	Turkish	e-commerce	31,757	89.88%
Gifari et al. (2021)	Stacking EL	2 (polarity)	English	movie	50,000	89.40%
Başarslan et al. (2023)	Majority Voting EL	3 (polarity)	English	movie	480	88.70%
Kılınç (2019)	Naïve Bayes, SVM	2 (polarity)	Turkish	hotels + movie (HUMIR)	65,000	87.02%
Danyal et al. (2024)	Bernoulli Naïve Bayes	2 (polarity)	English	movie	50,000	85.77%
Aydoğan et al. (2023)	B-LSTM with GloVe	3 (polarity)	Turkish	e-commerce	150,000	83.61%
Oliko et al. (2025)	MNB & Lexicon	3 (polarity)	English	e-commerce	37,724	82.50%
Demirci et al. (2019)	Neural Network	3 (polarity)	Turkish	politics	3,000	81.86%
Wijati et al. (2024)	KNN	3 (polarity)	Indonesian	software app	3,000	81.60%
Ucan et al. (2016)	lexicon based	3 (polarity)	Turkish	hotels + movie (Humir)	65,000	80.68%
Iyer (2024)	LR	3 (polarity)	English	social	1,600,000	77.81%
Armaeni et al. (2024)	DTC	3 (polarity)	Indonesian	social	7,000	74.71%
Rumelli et al. (2019)	KNN & Lexicon	3 (polarity)	Turkish	e-commerce	272,218	73.80%
Habberrih et al. (2024)	SVM	2 (polarity)	Arabic	poems	22,762	71.63%

It's noted that research papers in the literature report accuracies ranging from 80% to 90%, and only a few models in low-resource languages reach the 90s. Wijati et al. (2024) achieved

the best Indonesian result in the literature reviewed in this research, with an accuracy of 82%. Note that not all high-accuracy research papers in the English and Indonesian languages are mentioned in this research.

It's also noted that the text topic and language play a major role in determining the best model for sentiment analysis. As a result, a high-performing model might not achieve the same accuracy across different text topics, which requires taking into account the data's language and topic when comparing models.

Although product, movie, or hotel reviews often have a limited number of features, they can pose a significant challenge in avoiding informal language or speech. The nature of informal texts requires more complex preprocessing steps to retain valuable features while removing unimportant ones. On the other hand, news or social texts have a large feature space, while they may include more formal speech. In this study, several preprocessing techniques were tried, including stemming, stop-word removal, and word vectorization. The proposed preprocessing steps achieved the best accuracy in this study's circumstances.

3.3. Limitations

While this study compares several types of models, including LLMs, in the literature, it is intentionally concentrated on classical and ensemble learning models in practical experiments. These models were prioritized to explore low-to-mid computational complexity solutions. Secondly, the examined models are trained and evaluated exclusively on hotel and movie reviews, which may lead to undetected domain-specific overfitting. As a result, the model's performance might not generalize as effectively to other text genres.

4. CONCLUSION

This research tested various ML models and evaluated them on a precisely labeled dataset of over 65,000 records. The Stacking EL Classifier achieved 92.80% accuracy, but had an extremely high fit time, whilst the MNB, LSVMC, and LR had the best time-performance with a short fit time, achieving an accuracy of 92%. On the other hand, the Majority Voting EL model achieved 91% accuracy, similar to the MNB, but was considered time-inefficient, as it did not provide any improvement over the MNB. KNNC, DTC, and boosting models, including GBC and AdaBoost models, did not perform well compared to other models.

In summary, the Stacking EL Classifier and MNB models are considered the most performing classical models in Turkish sentiment analysis in this research. Future work would consider training models on datasets that include other types of text to mitigate the risk of domain sensitivity.

CONFLICT OF INTEREST

Authors approve that to the best of their knowledge, there is no conflict of interest or common interest with an institution/organization or a person that may affect the review process of the paper.

AUTHOR CONTRIBUTION

Ahmad BWIDANI and Ali KARAH BASH contributed to the conceptual design of the study. Ahmad BWIDANI managed the research process and carried out data collection. Both authors contributed to data analysis, interpretation of the results, manuscript preparation, and critical revision of the intellectual content. Both authors read and approved the final version of the manuscript and accept full responsibility for the study.

REFERENCES

1. Armaeni, P. P., Wiguna, I. K. A. G., & Parwita, W. G. S. (2024). Sentiment analysis of YouTube comments on the closure of TikTok Shop using Naïve Bayes and Decision Tree method comparison. *Jurnal Galaksi*, *1*(2), 70-80. doi:10.70103/galaksi.v1i2.15
2. Aydoğan, M., & Kocaman, V. (2023). TRSAv1: A new benchmark dataset for classifying user reviews on Turkish e-commerce websites. *Journal of Information Science*, *49*(6), 1711-1725. doi:10.1177/01655515221074328
3. Başarslan, M. S., & Kayaalp, F. (2023). Sentiment analysis with ensemble and machine learning methods in multi-domain datasets. *Turkish Journal of Engineering*, *7*(2), 141-148. doi:10.31127/tuje.1079698
4. Danyal, M. M., Khan, S. S., Khan, M., Ullah, S., Ghaffar, M. B., & Khan, W. (2024). Sentiment analysis of movie reviews based on NB approaches using TF–IDF and count vectorizer. *Social network analysis and mining*, *14*(1), 87. doi:10.1007/s13278-024-01250-9
5. Demircan, M., Seller, A., Abut, F., & Akay, M. F. (2021). Developing Turkish sentiment analysis models using machine learning and e-commerce data. *International Journal of Cognitive Computing in Engineering*, *2*, 202-207. doi:10.1016/j.ijcce.2021.11.003
6. Demirci, G. M., Keskin, Ş. R., & Doğan, G. (2019, December). Sentiment analysis in Turkish with deep learning. In *2019 IEEE international conference on big data (big data)* (pp. 2215-2221). IEEE. doi:10.1109/BigData47090.2019.9006066
7. Erşahin, B., Aktaş, Ö., Kilinç, D., & Erşahin, M. (2019). A hybrid sentiment analysis method for Turkish. *Turkish Journal of Electrical Engineering and Computer Sciences*, *27*(3), 1780-1793. doi:10.3906/elk-1808-189
8. Gifari, M. K., Lhaksmana, K. M., & Dwifebri, P. M. (2021, October). Sentiment analysis on movie review using ensemble stacking model. In *2021 International conference advancement in data science, e-learning and information systems (ICADEIS)* (pp. 1-5). IEEE. doi:10.1109/ICADEIS52521.2021.9702088
9. Guven, Z. A. (2021, September). Comparison of BERT models and machine learning methods for sentiment analysis on Turkish tweets. In *2021 6th international conference on computer science and engineering (UBMK)* (pp. 98-101). IEEE. doi:10.1109/UBMK52708.2021.9559014
10. Habberrih, A., & Ali Abuzaraida, M. (2024, July). Sentiment analysis of Libyan dialect using machine learning with stemming and stop-words removal. In *5TH INTERNATIONAL CONFERENCE ON COMMUNICATION ENGINEERING AND COMPUTER SCIENCE (CIC-COCOS'24)* (pp. 259-264). Cihan University-Erbil. doi:10.24086/cocos2024/paper.1171
11. Iyer, V. (2024). A Comparative Analysis of Sentiment Classification Models for Improved Performance Optimization. *TechRxiv*. doi:10.36227/techrxiv.171073040.03879551/v1
12. Karagoz, P., Kama, B., Ozturk, M., Toroslu, I. H., & Canturk, D. (2019). A framework for aspect based sentiment analysis on turkish informal texts. *Journal of Intelligent Information Systems*, *53*(3), 431-451. doi:10.1007/s10844-019-00565-w
13. Karah Bash, A. A., & Ercelebi, E. (2024). Advancing Sentiment Analysis during the Era of Data-Driven Exploration via the Implementation of Machine Learning Principles, *Balkan Journal of Electrical and Computer Engineering, (2024), 12(1), 1-9*. doi:10.17694/bajece.1340321
14. Kılınç, D. (2019). A spark-based big data analysis framework for real-time sentiment prediction on streaming data. *Software: Practice and Experience*, *49*(9), 1352-1364. doi:10.1002/spe.2724
15. Kiziltepe, R. S., Ezin, E., & Karakus, M. (2024, October). A comparative study of ml models on large-scale turkish sentiment datasets. In *2024 Innovations in Intelligent Systems*

and Applications Conference (ASYU) (pp. 1-6). IEEE. doi:10.1109/ASYU62119.2024.10757130

16. Mäntylä, M. V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Computer science review*, *27*, 16-32. doi:10.1016/j.cosrev.2017.10.002

17. Okereke, G. E., Udanor, C. N., Echezona, S. C., Asogwa, C. N., & Uzoh, I. (2024). Sentiment Analysis-An optimized Weighted Horizontal Ensemble approach. *International Journal of Advanced Trends in Computer Science and Engineering*, *13*(2), 72-79. doi:10.30534/ijatcse/2024/061322024

18. Oliko, G. V., Otieno, C., & Muhambe, T. M. (2025). An Ensemble Model Based on Multinomial Naïve Bayes and Lexicon for Sentiment Classification of Product Reviews. *International Journal of Computer Trends and Technology*, *73*(3), 1-15. doi:10.14445/22312803/IJCTT-V73I3P101

19. Özçift, A., Akarsu, K., Yumuk, F., & Söylemez, C. (2021). Advancing natural language processing (NLP) applications of morphologically rich languages with bidirectional encoder representations from transformers (BERT): an empirical case study for Turkish. *Automatika: časopis za automatiku, mjerenje, elektroniku, računarstvo i komunikacije, 62(2)*, 226-238. doi:10.1080/00051144.2021.1922150

20. Özmen, C. G., & Gündüz, S. (2025). Comparison of machine learning models for sentiment analysis of big Turkish web-based data. *Applied Sciences*, *15*(5), 2297. doi:10.3390/app15052297

21. Rumelli, M., Akkuş, D., Kart, Ö., & Isik, Z. (2019, October). Sentiment analysis in Turkish text with machine learning algorithms. In *2019 Innovations in intelligent systems and applications conference (ASYU)* (pp. 1-5). IEEE. doi:10.1109/ASYU48272.2019.8946436

22. Sağbaş, E. A. (2024). A novel two-stage wrapper feature selection approach based on greedy search for text sentiment classification. *Neurocomputing*, *590*, 127729. doi:10.1016/j.neucom.2024.127729

23. Savci, P., & Das, B. (2023). Prediction of the customers' interests using sentiment analysis in e-commerce data for comparison of Arabic, English, and Turkish languages. *Journal of King Saud University-Computer and Information Sciences*, *35*(3), 227-237. doi:10.1016/j.jksuci.2023.02.017

24. Subba, B., & Kumari, S. (2022). A heterogeneous stacking ensemble based sentiment analysis framework using multiple word embeddings. *Computational Intelligence*, *38*(2), 530-559. doi:10.1111/coin.12478

25. Ucan, A., Naderalvojoud, B., Sezer, E. A., & Sever, H. (2016, November). SentiWordNet for new language: Automatic translation approach. In *2016 12th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)* (pp. 308-315). IEEE. doi:10.1109/SITIS.2016.57

26. Wijati, D., Atika, P. D., Setiawati, S., & Rasim, R. (2024). Sentiment analysis of application reviews using the K-Nearest Neighbors (KNN) algorithm. *PIKSEL: Penelitian Ilmu Komputer Sistem Embedded and Logic*, *12*(1), 209-218. doi:10.33558/piksel.v12i1.9490

27. Yıldırım, M., Okay, F. Y., & Özdemir, S. (2020, December). Sentiment analysis for Turkish unstructured data by machine translation. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 4811-4817). IEEE. doi:10.1109/BigData50022.2020.9377784

28. Zorarpaci, E. (2024, September). A Comparative Study of Ensemble Learning Methods for Turkish Text Categorization. In *2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP)* (pp. 1-6). IEEE. doi:10.1109/IDAP64064.2024.10711008