Fake Review Detection

Shivank Singh Thakur, Rahul Reddy Gangapuram, Prudhvi Sai Raj Dasari, Jathin Shettigar Nagabhushan
[1]Computer Science Department, [2]College of Graduate Studies, [1,2]San Jose State University
{*shivanksingh.thakur, rahulreddy.gangapuram, prudhvisairaj.dasari, jathin.shettigarnagabhushan*}*@sjsu.edu*

***Abstract*—With the rise of online platforms, fake reviews have become a pervasive issue, compromising consumer trust and distorting the integrity of digital marketplaces. These deceptive reviews often manipulate user perceptions by either exaggerating or downplaying the quality of products and services. Detecting fake reviews is crucial for ensuring transparency, but distinguishing between genuine and fraudulent content is a complex task. This paper explores various features and techniques for identifying fake reviews, focusing on linguistic and behavioral indicators, including average word and sentence lengths, verb and adjective usage, sentence structure, content diversity, and the presence of typos. By analyzing these features—such as the number of passive voice constructions and typo ratios—we aim to provide a comprehensive understanding of current strategies and highlight challenges in improving detection accuracy. Ultimately, this study offers insights into future research directions to enhance the effectiveness and scalability of fake review detection methods.**

***Index Terms*—Fake Review, Linguistic indicators, Behavioral indicators**

I. Introduction

Online reviews play a crucial role in shaping consumer decisions, but the rise of fake reviews—intended to artificially boost or damage a product's reputation—has compromised the integrity of digital marketplaces. Detecting these fraudulent reviews is challenging, as fraudsters use sophisticated techniques to mimic genuine content. Researchers have focused on analyzing various linguistic features to distinguish fake reviews, such as average word and sentence lengths, verb and adjective usage, passive voice, content diversity, and typo frequency. By examining these features, this paper explores current methods for fake review detection and discusses the challenges in improving their accuracy and scalability.
[1]

A. Key contributions

- Rahul and Jathin worked on the feature extraction and processing clean reviews.
- Prudhvi and Shivank worked on model training and testing.
- We leveraged Logistic Regression, Random Forest, XGBoost and SVM models to classify the reviews as fake or not. We extracted 10 features AWL: Average word length, ASL: Average sentence length, NWO: Number of words, NVB: Number of verbs, NAJ: Number of adjectives, NPV: Number of passive voice, NST: Number of sentences, CDV: Content diversity, NTP: Number of typos, TPR: Typo ratio, and then performed review cleaning.

[1]https://www.overleaf.com/read/fbbrcnmqkmhb#f08e3b

B. Organization of the Paper

The paper is organized as follows: Section I introduces the paper, and Section II describes the dataset. Section III explains the preprocessing techniques performed on the dataset. Section IV presents the feature extraction techniques. Section V discusses the various classification models trained. Section V presents the results and evaluation. Section VI concludes the paper with the conclusion and future work.

II. Dataset Description

The dataset consists of restaurant reviews, each accompanied by various features aimed at aiding in fake review detection and sentiment analysis. The columns in the dataset are the name of the restaurant, the review and the label. The Real/Fake column labels the review as either real (1) or fake (0), and the Positive/Negative column represents the sentiment of the review, where 1 denotes positive and 0 denotes negative sentiment. The dataset is evenly distributed with same number of fake and real reviews. Several linguistic features are extracted from the review, such as AWL (Average Word Length), ASL (Average Sentence Length), and NOW (Number of Words), which provide insights into the structure of the review. Additionally, the NVB (Number of Verbs), NAJ (Number of Adjectives), and NPV (Number of Passive Voice) columns quantify specific linguistic elements, while NST (Number of Sentences) captures the review's length. CDV (Content Diversity) measures the diversity of vocabulary by calculating the ratio of unique words to the total number of tokens, excluding punctuation and stop words. Finally, NTP (Number of Typos) and TPR (Typo Ratio) assess the presence of spelling errors, which can be indicative of fake reviews. Together, these features enable analysis of review authenticity, sentiment, and linguistic patterns.

III. Preprocessing

1) **Handling Missing Values and Imbalance**: There are no missing, NaN or null values in the dataset. The dataset is balanced as well.
2) **Extracting features**: The features AWL: Average word length, ASL: Average sentence length, NWO: Number of words, NVB: Number of verbs, NAJ: Number of adjectives, NPV: Number of passive voice, NST: Number of sentences, CDV: Content diversity, NTP: Number of typos, TPR: Typo ratio are extracted from the review

column. All features except CDV are extracted before cleaning the review.

3) **Cleaning reviews**: The reviews are converted into lower text, removing numbers, punctuations, tokenizing the words and finally removing stop words.
4) **Data Splitting**: Each dataset was split into 60% training, 20% validation and 20% testing sets. The validation dataset is used for tuning the hyperparameters of the classification models. The dataset is shuffled and stratified as part of splitting.
5) **Feature Selection**: We used Recursive Feature Elimination, Cross-Validated (RFECV) with Random Forest model as estimator for feature selection. It selects the best subset of features for the supplied estimator by removing 0 to N features (where N is the number of features) using recursive feature elimination, then selecting the best subset based on the cross-validation score of the model. F1 score was used as the metric for comparison between models.

IV. Feature Extraction

A. Tokenization (Words and Sentences)

- **Technique:** Tokenization splits the text into words and sentences.
- **Method:** The review is split into words using Python's `split()` method. Sentence segmentation is performed using `sent_tokenize()` from the `nltk` library.
- **Library:** `nltk` (Natural Language Toolkit) for sentence tokenization.

B. Basic Statistics (Number of Words, Sentences, Average Word Length, Average Sentence Length)

- **Technique:** Basic statistical measures are calculated by counting the number of words and sentences, then computing averages for word length and sentence length.
- **Method:**
  - The number of words is simply the length of the tokenized list. The punctuations are removed and converted to lower case.
  - The number of sentences is the length of the list generated by sentence tokenization.
  - Average word length is the total character count of all words divided by the number of words.
  - Average sentence length is the number of words divided by the number of sentences.
- **Library:** Standard Python functions and `nltk` for sentence tokenization.

C. Part-of-Speech (POS) Tagging (Number of Verbs, Adjectives)

- **Technique:** POS tagging classifies each word into its grammatical category (e.g., verb, adjective).
- **Method:** `nltk.pos_tag()` is used to tag words with their respective POS labels. The number of verbs and adjectives is then counted based on POS tags (e.g., 'VB' for verbs, 'JJ' for adjectives).
- **Library:** `nltk` for POS tagging.

D. Content Diversity

- **Technique:** Content diversity is a measure of how varied the vocabulary is in a review.
- **Method:** After tokenizing the review into words and removing stop words and punctuation (if implemented in preprocessing), the ratio of unique words to total words is calculated.
- **Library:** This is calculated using standard Python functions, with additional stop word filtering potentially relying on libraries like `nltk` or `spaCy`.

E. Passive Voice Detection

- **Technique:** Identifying passive voice constructs in a sentence.
- **Method:** By looking for certain POS tags such as 'VBN' (past participle) or 'VBD' (past tense verbs), passive constructions can be detected. These verbs often signal passive voice usage.
- **Library:** `nltk` for POS tagging.

F. Spell Checking for Typos

- **Technique:** Detecting spelling errors in the review by comparing each word to its most likely correct form.
- **Method:** `TextBlob` is used to process the review text and identify typos. TextBlob can detect and correct misspelled words, it uses Peter Norvig's spelling corrector algorithm. It can identify and label words with their grammatical roles in a sentence. Each word is compared against its corrected form using the `Word().correct()` method from `TextBlob`.
- **Library:** `TextBlob` for spell checking and `Word` for correction.

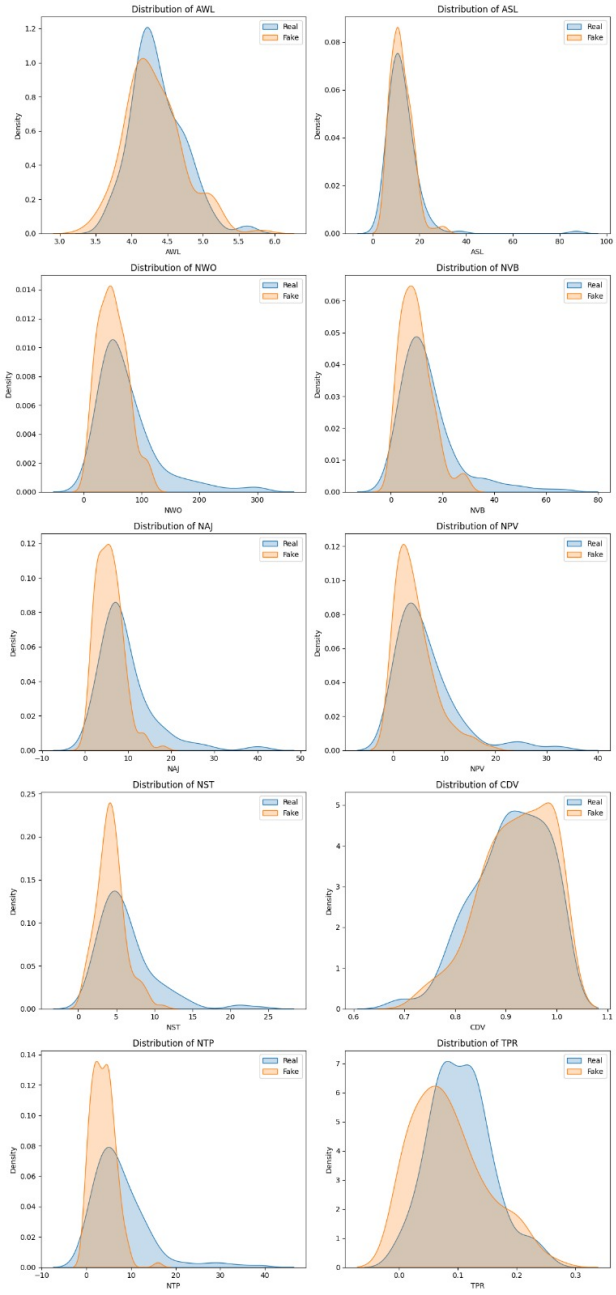
G. Typos Ratio

- **Technique:** Calculating the ratio of words with typos to the total number of words in the review.
- **Method:** The number of typos is determined by comparing each word with its corrected form (using `TextBlob`), and the ratio is calculated by dividing the number of typos by the total number of words.
- **Library:** `TextBlob` for typo checking.

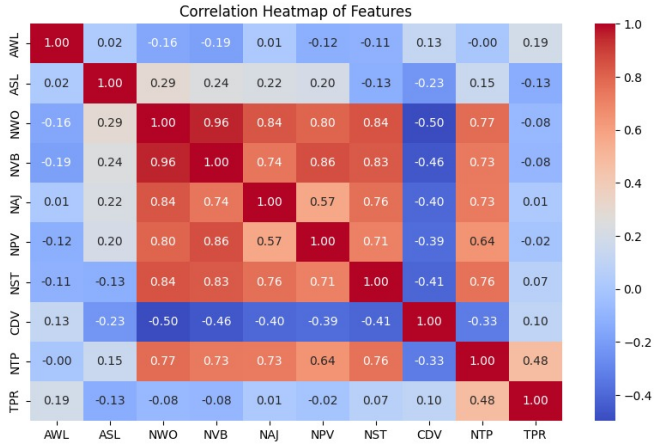
V. Classification Models

We implemented four models:

- **Logistic Regression**: Logistic regression is a simple supervised machine learning algorithm known for its efficiency and low computational cost. It is used for predictive analysis and uses the sigmoid function to estimate the probability of each class.
- **Random Forest:**: Random Forest is a supervised learning algorithm that uses an ensemble of decision trees and bootstrapping on the dataset. It combines the majority vote from multiple trees to make predictions. Random



(a) Distribution of fake vs real



(b) Correlation Matrix

(c) World Cloud for Real Reviews

(d) World Cloud for Fake Reviews

Forests outperform single decision trees as they offer better generalization and are less sensitive to small variations in the training data, thus helping reduce overfitting.

- **eXtreme Gradient Boosting**: eXtreme Gradient Boosting is a powerful supervised ensemble technique that builds multiple decision trees sequentially. Each tree attempts to correct the residual errors from the previous one. The final prediction is the combined result from all the trees, making XGBoost highly effective for complex tasks and improving prediction accuracy.
- **Support Vector Machine**: Support Vector Machine (SVM) classification is a supervised machine learning algorithm used for binary and multiclass classification tasks. It works by finding a hyperplane that best separates data points of different classes in a high-dimensional feature space.

A. Hyperparameter Tuning

To optimize the performance of the models, RandomizedSearchCV was used for hyperparameter tuning. This method allows us to efficiently explore a wide range of hyperparameter combinations without the computational burden of an exhaustive grid search. The tuning process was performed using 5-fold Stratified Cross-Validation (StratifiedKFold) to ensure balanced representation of both classes in each fold.

For each model, training and validation sets were combined to maximize the amount of data used for hyperparameter tuning. This helped improve the model's ability to generalize by exposing it to a broader distribution of samples. The test set remained untouched during tuning to ensure an unbiased evaluation of model performance.

Each model went through 10 iterations of randomized search, selecting hyperparameter combinations randomly from the defined ranges. This approach efficiently explored the parameter space while keeping computational costs reasonable. The performance of each configuration was evaluated using F1-score, ensuring a balance between precision and recall. This hyperparameter tuning process ensured that each model was optimized for detecting fake reviews while maintaining good generalization to unseen data. Below are the hyperparameters used for tuning each model:

1) Logistic Regression:

- C: {0.001, 0.01, 0.1, 1, 10, 100}
- Solver: liblinear, saga
- Class Weight: None, balanced
- Penalty: l1, l2

2) Random Forest:

- Number of Trees (n_estimators): {100, 200, 300}
- Maximum Depth (max_depth): {5, 10, 15, 20}
- Feature Selection (max_features): sqrt, log2, 0.33
- Minimum Samples per Leaf (min_samples_leaf): {2, 4, 8}
- Minimum Samples per Split (min_samples_split): {5, 10, 15}
- Bootstrap Sampling (bootstrap): True, False
- Maximum Samples (max_samples): {0.7, 0.8, 0.9} (only if bootstrap is True)
- Class Weighting (class_weight): balanced, balanced_subsample, None
- Splitting Criterion (criterion): gini, entropy

3) XGBoost:

- Learning Rate (learning_rate): {0.01, 0.05, 0.1}
- Maximum Depth (max_depth): {3, 5, 7}
- Subsampling Rate (subsample): {0.6, 0.8, 1.0}
- Column Subsampling (colsample_bytree): {0.6, 0.8, 1.0}
- Gamma (gamma): {0, 0.1, 0.2}
- L1 Regularization (reg_alpha): {0, 0.1, 1}
- L2 Regularization (reg_lambda): {0, 0.1, 1}

4) Support Vector Machine (SVM):

- Kernel Type (kernel): rbf, linear, poly, sigmoid
- Regularization Parameter (C): sampled from a continuous uniform distribution between 0.1 and 100
- Polynomial Degree (degree): sampled from a discrete uniform distribution between 3 and 10 (only for polynomial kernel)
- Gamma (gamma): scale, auto

B. Evaluation Metrics

After hyperparameter tuning, the best models were retrained on the full training combined with validation set before being evaluated on the test set. We evaluated the models based on macro F1 and accuracy scores. The macro F1 score is an evaluation metric that averages the F1 scores of all classes, treating each class equally regardless of its frequency, and providing a balanced measure of a model's precision and recall. Accuracy is a metric that measures the percentage of correct predictions made by a model, calculated as the ratio of correct predictions to the total number of predictions.

VI. Results

Accuracy and F1 score metrics pick up essential information regarding data balancing, feature selection, and to some extent, model behavior. In this fake review detection task, six features were selected for training. One of the interesting findings was that the number of typos (NTP) was important in identifying fake reviews, especially for tree-based models like Random Forest and XGBoost. However, Logistic Regression and SVM did not rely on any single feature, which means they likely used a combination of different factors to make predictions.

Based on model performance, Logistic Regression had the best generalization, meaning it performed consistently on both training and test data. It had an F1-score of 0.688 on training data and 0.641 on test data, showing that it did not overfit. Although it was not the best model, it was more stable than some of the more complex models.

Random Forest had the highest test accuracy (0.673) and was fairly balanced in its performance. However, its test F1-score (0.622) was slightly lower than Logistic Regression, meaning it sometimes struggled with predicting fake reviews

TABLE I: Top 3 hyperparameters for each classification model

Model	Hyperparameters	Mean F1 Score (CV)
LR	{solver: saga, penalty: l2, class_weight: None, C: 1}	0.6816
	{solver: saga, penalty: l2, class_weight: None, C: 0.1}	0.6787
	{solver: saga, penalty: l1, class_weight: balanced, C: 10}	0.6721
RF	{n_estimators: 200, min_samples_split: 15, min_samples_leaf: 2, max_samples: 0.9, max_features: 0.33, max_depth: 10, criterion: gini, class_weight: balanced, bootstrap: True}	0.6308
	{n_estimators: 200, min_samples_split: 5, min_samples_leaf: 4, max_samples: 0.9, max_features: log2, max_depth: 20, criterion: entropy, class_weight: None, bootstrap: True}	0.6247
	{n_estimators: 200, min_samples_split: 15, min_samples_leaf: 4, max_samples: 0.9, max_features: 0.33, max_depth: 15, criterion: entropy, class_weight: None, bootstrap: True}	0.6235
XGB	{subsample: 0.8, reg_lambda: 1, reg_alpha: 0, max_depth: 7, learning_rate: 0.1, gamma: 0.1, colsample_bytree: 0.6}	0.6368
	{subsample: 1.0, reg_lambda: 0.1, reg_alpha: 1, max_depth: 5, learning_rate: 0.05, gamma: 0, colsample_bytree: 0.8}	0.6272
	{subsample: 1.0, reg_lambda: 0.1, reg_alpha: 1, max_depth: 7, learning_rate: 0.05, gamma: 0.1, colsample_bytree: 0.8}	0.6182
SVM	{C: 78.07, degree: 7, gamma: scale, kernel: linear}	0.6577
	{C: 97.09, degree: 6, gamma: auto, kernel: linear}	0.6514
	{C: 23.38, degree: 8, gamma: scale, kernel: linear}	0.6486

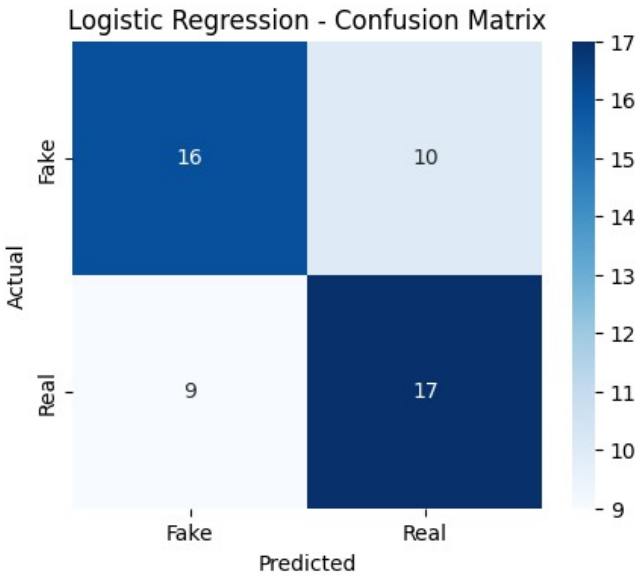
TABLE II: Model Performance

Model	Train Accuracy	Train F1-score	Test Accuracy	Test F1-score
Logistic Regression	0.6731	0.6881	0.6346	**0.6415**
Random Forest	0.8510	0.8410	**0.6731**	0.6383
XGBoost	0.9904	0.9904	0.6346	0.6122
SVM	0.6923	0.6768	0.6538	0.5909

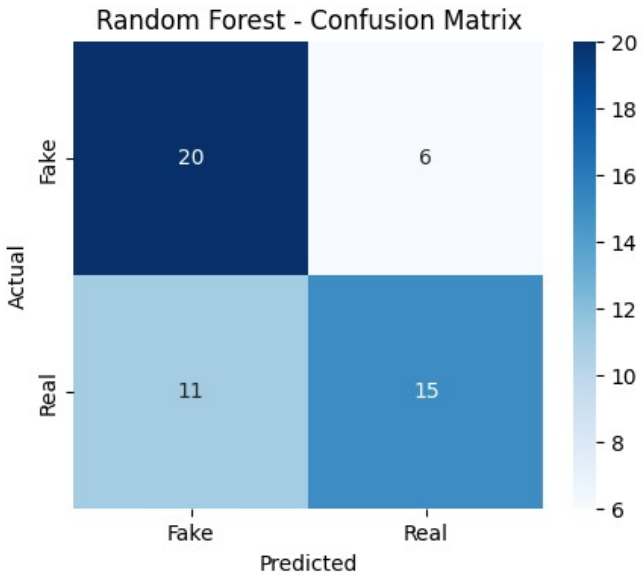
correctly. Like XGBoost, Random Forest found typos to be the most important feature.

XGBoost, on the other hand, struggled with overfitting. It performed extremely well on training data (99% accuracy) but dropped to 0.612 F1-score on test data. This indicates that the model memorized the training data and struggled to generalize to new reviews. This overfitting might be due to the model's complexity, and it might need further exhaustive fine tuning or more diverse training data to help it generalize better to unseen examples.

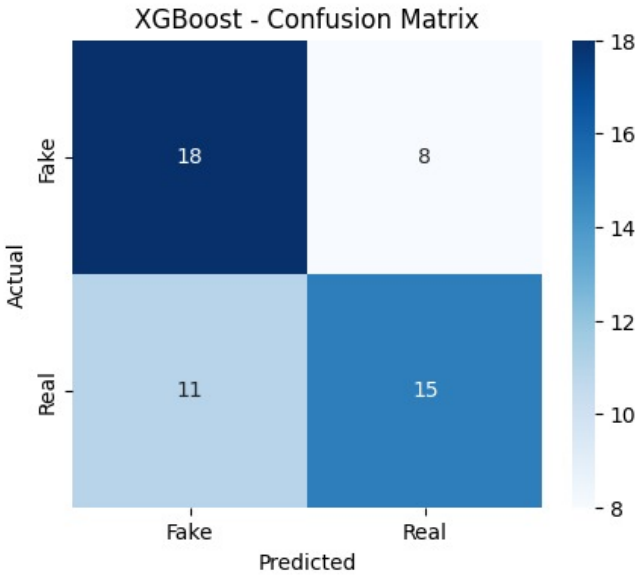
SVM had the highest test accuracy (0.654), but the lowest F1-score (0.591). This suggests that while it correctly identified many reviews, it may have been biased towards the fake reviews class (as per the confusion matrix) and was not as effective at identifying real reviews.



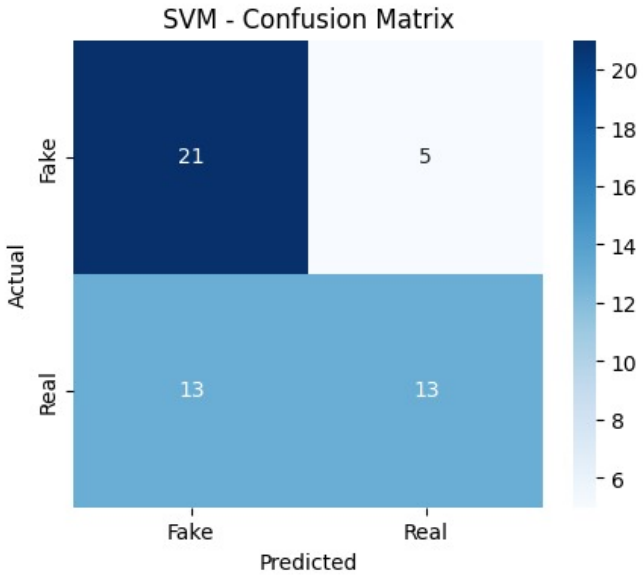
(a) Linear Regression



(b) Random Forest



(c) XG-Boost



(d) Support Vector Machine

Fig. 2: Confusion Matrix for classification models

VII. Conclusion and Future Work

In conclusion, Random Forest had the best balance between accuracy and generalization, making it the most reliable model overall for fake review detection. XGBoost overfitted to the training data, which suggests that reducing model complexity or increasing the amount of diverse training data could help improve its generalization. Logistic Regression was stable and generalized well, even though its overall performance was not the highest. SVM achieved the highest accuracy but struggled with F1-score, likely due to class imbalance, which suggests that techniques like resampling or adjusting decision thresholds could enhance its effectiveness. Incorporating additional features, particularly those related to sentiment analysis or review structure, could improve classification performance, as fake and real reviews might exhibit different linguistic patterns. Furthermore, using text embeddings instead of handcrafted linguistic features could allow models to capture deeper semantic relationships, leading to better generalization. Overall, Random Forest emerged as the best-performing model in this dataset, as it effectively balanced learning patterns while avoiding significant overfitting.