Gambling Comments Detection on YouTube: A Comparative Study of Tree-Based Boosting, LSTM and GRU Models

Agung Widiyanto [1,*)], Mayesq Prameswari [2)], and Muhammad Abdul Latief [3)]

[1, 2, 3)] Study Program in Data Science, Telkom University Purwokerto, Purwokerto, Indonesia

E-mail: agungwdyy@student.telkomuniversity.ac.id[1)], mayesq@student.telkomuniversity.ac.id[2)], and abdullatief@student.telkomuniversity.ac.id[3)]

ABSTRACT

The exponential growth of online gambling in Indonesia poses significant socio-economic challenges, particularly affecting vulnerable populations through sophisticated digital marketing strategies targeting social media platforms. This study addresses the critical need for automated detection systems to identify gambling-related content in YouTube comments. We scraped and manually labeled 11,673 comments from diverse YouTube videos, creating an extremely imbalanced dataset with gambling comments representing only 10% of the total data. Multiple machine learning approaches were developed and evaluated, comparing traditional gradient boosting methods (LightGBM, XGBoost, CatBoost) using TF-IDF features against deep learning models (LSTM and GRU) with Word2Vec embeddings. The experimental results demonstrate that gradient-boosting methods significantly outperform deep learning approaches in generalization capability. LightGBM achieved the highest holdout F1-score (0.8737) with balanced precision (0.8912) and recall (0.8886), while XGBoost followed closely with comparable performance. In contrast, deep learning models exhibited severe overfitting. While the GRU model showed excellent test performance (precision: 0.9849, recall: 0.9378), its holdout recall drastically reduced to 0.5022, resulting in a low holdout F1-score (0.6647). Similarly, LSTM performed well on the test set (precision: 0.9610, recall: 0.9426) but failed to maintain consistency on holdout data (holdout recall: 0.5733, F1-score: 0.7207). The findings indicate that the dataset size was insufficient for deep learning approaches to learn generalizable representations effectively. For practical deployment in YouTube gambling content detection, gradient boosting methods are recommended due to their superior performance with limited, imbalanced datasets.

Keywords: gambling comment detection, LSTM, GRU, tree-based, boosting, YouTube comments

1. Introduction

The digital age has brought a major shift in how people interact and access information, with social media platforms becoming an integral part of everyday life. This widespread use has also made these platforms vulnerable to misuse. One of the most concerning forms of exploitation is their use in promoting online gambling. Such activities pose serious ethical and legal challenges in the digital landscape. Online gambling promotion, disseminated extensively through social media, has emerged as a pivotal strategy for gambling operators to expand their reach to potential players, facilitated by the round-the-clock accessibility of online gambling sites. In Indonesia, this trend has precipitated a dramatic surge in the exploitation of social media channels for online gambling advertisements, presenting substantial and multifaceted challenges to regulatory bodies and law enforcement agencies [1]. The proliferation of online gambling in Indonesia is now recognized as a significant public health concern, fueled by a complex interplay of psychological, social, and regulatory factors that contribute to increased participation, particularly among vulnerable young adult populations [2]. Gambling operators are observed to employ increasingly sophisticated techniques, such as black hat Search Engine Optimization (SEO), and exploit vulnerabilities within official websites, including government domains like '.go.id', to covertly promote online gambling activities, indicating a complex and highly adaptive operational strategy [3]. Furthermore, the phenomenon of online gambling

* Corresponding author.
Received: May 31st, 2025. Revised: June 25th, 2025. Accepted: July 7th, 2025.

Available online: July 8th, 2025.

DOI: https://doi.org/10.12962/j24068535.v23i1.a1305

in Indonesia is not an isolated issue of addiction but has demonstrated cascading negative societal effects, including documented links to severe organized crimes such as human trafficking, thereby highlighting profound societal repercussions that extend far beyond the immediate scope of gambling addiction itself [4]. The combination of widespread social media use and the increasingly sophisticated and flexible nature of illegal promotional methods presents a particularly difficult challenge in Indonesia, requiring countermeasures that are both advanced and adaptable. As such, Indonesia offers a significant case that reflects global difficulties, where existing regulations and enforcement efforts frequently lag behind the fast-paced, technology-driven development of online gambling promotion tactics. The socio-economic consequences of online gambling are severe, impacting individuals and communities significantly [2][4]. YouTube, a major platform for content and user comments, has become a key venue for gambling-related discussions and promotions [5]. Studies by [6] and [5] highlight YouTube comments as both a reflection of public perception, including stigma, and an active channel for gambling promotion, with promotional comments showing distinct characteristics like higher density and repetition. The sheer volume of this user-generated content makes manual moderation impractical [7], necessitating automated detection systems [8]. Research by [9] and [1] underscores the potential of Machine Learning (ML) and Deep Learning (DL) for this task [10], especially supervised learning approaches [10]. However, the ever-changing characteristics of promotional language call for flexible systems that can effectively balance precision and recall, aiming to reduce both undetected promotional content and incorrect classifications [7].

This study compares two main automated approaches, such as traditional ensemble learning methods and sequential deep learning architectures. Tree-based boosting algorithms like LightGBM, XGBoost, and CatBoost [11], which combine multiple weak learners [12], have shown robust performance using Term Frequency-Inverse Document Frequency (TF-IDF) features. TF-IDF quantifies word importance, providing discriminative features for these algorithms. In contrast, sequential DL models like Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) [13], and Gated Recurrent Units (GRU) [14] are designed to process sequential data and capture long-range dependencies. These models often utilize Word2Vec embeddings [15], which provide dense, semantically rich vector representations of words [16], as seen in [1]. While theoretically advantageous for understanding context, the performance of LSTM/GRU models heavily depends on dataset size and quality.

A major challenge in this research is the highly imbalanced nature of the dataset, with gambling comments representing only 10% of the data, which can bias models towards the majority class [17]. Deep learning models are particularly prone to overfitting on such limited or imbalanced data, learning noise rather than generalizable patterns [18]. This study indicates that while LSTM and GRU models performed well on test data, their holdout recall was drastically reduced, a classic sign of overfitting. Regularization techniques like dropout and early stopping are common countermeasures [19]. The primary objective of this research is to compare these ML paradigms on the task of detecting gambling-related YouTube comments. The findings demonstrate that gradient boosting methods (LightGBM and XGBoost) significantly outperform deep learning approaches in generalization, with LightGBM achieving the highest F1-score. Consequently, for practical deployment on limited, imbalanced datasets, gradient-boosting methods are recommended [20].

This research makes several key contributions to address this gap and differentiate itself from existing studies. Firstly, we introduce a new, manually annotated dataset comprising 11,673 YouTube comments, specifically curated for the task of gambling content detection in the Indonesian context. Secondly, we conduct a rigorous comparative study of two distinct machine learning paradigms: traditional tree-based boosting methods (LightGBM, XGBoost, CatBoost) using TF-IDF features against sequential deep learning models (LSTM and GRU) with Word2Vec embeddings. Unlike prior work that focused on Twitter or the infiltration of official websites, our study specifically targets the nuanced language and characteristics of gambling promotions within user-generated YouTube comments. Furthermore, while research exists on general YouTube spam detection, this study provides a direct comparison of the generalization capabilities of boosting and deep learning models on a highly imbalanced dataset created explicitly for gambling content, thereby assessing their practical deployment readiness.

The remainder of this paper is organized as follows. Section 2 reviews related research in the fields of gambling content detection and text classification. Section 3 details the research methodology, including data collection, preprocessing, feature extraction, and the models employed. Section 4 presents the experimental setup, results, and a detailed analysis of the model performances. Finally, Section 5 concludes the study, summarizes the key findings, and proposes directions for future work.

2. Related Research

This research specifically focuses on detecting gambling comments, particularly on YouTube, an area where existing studies remain scarce. To establish a more comprehensive foundation for this work, the scope has been expanded to include research from closely related domains such as text classification, spam detection, and online gambling site detection. These fields offer methodological approaches and insights that prove valuable when adapted to the challenge of identifying gambling-related comments. Through examining these interconnected areas, this study seeks to uncover proven techniques and understand the challenges that directly inform the research objectives.

A relevant study conducted by Perdana et al. [1] tackled the complex challenge of identifying gambling-related content within the Indonesian social media landscape, where research remains particularly limited [1]. The researchers analyzed a comprehensive dataset of 6,038 tweets, employing multiple classification approaches including Random Forest, Logistic Regression, and Convolutional Neural Networks, while also conducting comparative analyses of various text representation methods. Their investigation revealed frequently occurring promotional terms such as 'link', 'situs', 'prediksi', 'jackpot', 'maxwin', and 'togel', providing valuable insights into the linguistic patterns of Indonesian gambling promotions. The study demonstrated that combining TF-IDF feature extraction with Random Forest classification yielded superior performance, achieving impressive results with a recall of 0.958 and precision of 0.966. These findings not only advance the methodological understanding of gambling content detection but also offer practical applications for cybersecurity initiatives and law enforcement efforts aimed at mitigating the harmful effects of online gambling promotions on Indonesian social media platforms.

While the previously discussed research focused on social media gambling comments detection, a related security concern involves the infiltration of gambling content into official websites. Nurseno et al. [3] investigated how gambling operators exploit government domains. The study analyzed 450,000 .go.id domains using a Python-based web scraping algorithm that identified gambling-related keywords such as 'slot,' 'judi,' 'gacor,' and 'togel.' The research revealed that 958 out of 1,482 suspected government sites had been compromised with hidden gambling URLs, achieving a 99.1% detection accuracy. This work demonstrates the broader scope of online gambling infiltration beyond social media platforms, highlighting the need for comprehensive detection approaches across different digital environments. Additionally, Min and Lee [21] examined the rise of illegal online gambling during the COVID-19 pandemic, employing a machine learning-driven approach that combined textual and image features for high detection performance. Their model analyzed key attributes such as URLs, WHOIS, INDEX, and landing page metadata across 11,172 websites, suggesting a strategy for dynamic resource utilization to enhance classification accuracy. This study further underscores the need for adaptive and multi-modal detection strategies in combating online gambling threats.

Another relevant study by Airlangga [20] examined the effectiveness of various deep learning models in detecting spam comments on YouTube. The research utilized a dataset comprising 1,956 real comments from popular YouTube videos, representing both spam and legitimate content. Preprocessing included tokenization and padding of text sequences to prepare them for input into six different models: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU), and models with attention mechanisms. The findings indicated that the LSTM model outperformed other architectures with a test accuracy of 95.65%, emphasizing the critical role of sequential modeling in capturing contextual information within user comments. CNN models also demonstrated strong performance by effectively recognizing local patterns. These insights highlight the potential of combining sequential modeling and local feature extraction for robust spam detection systems in online platforms. Xiao and Liang [22] evaluated eight traditional

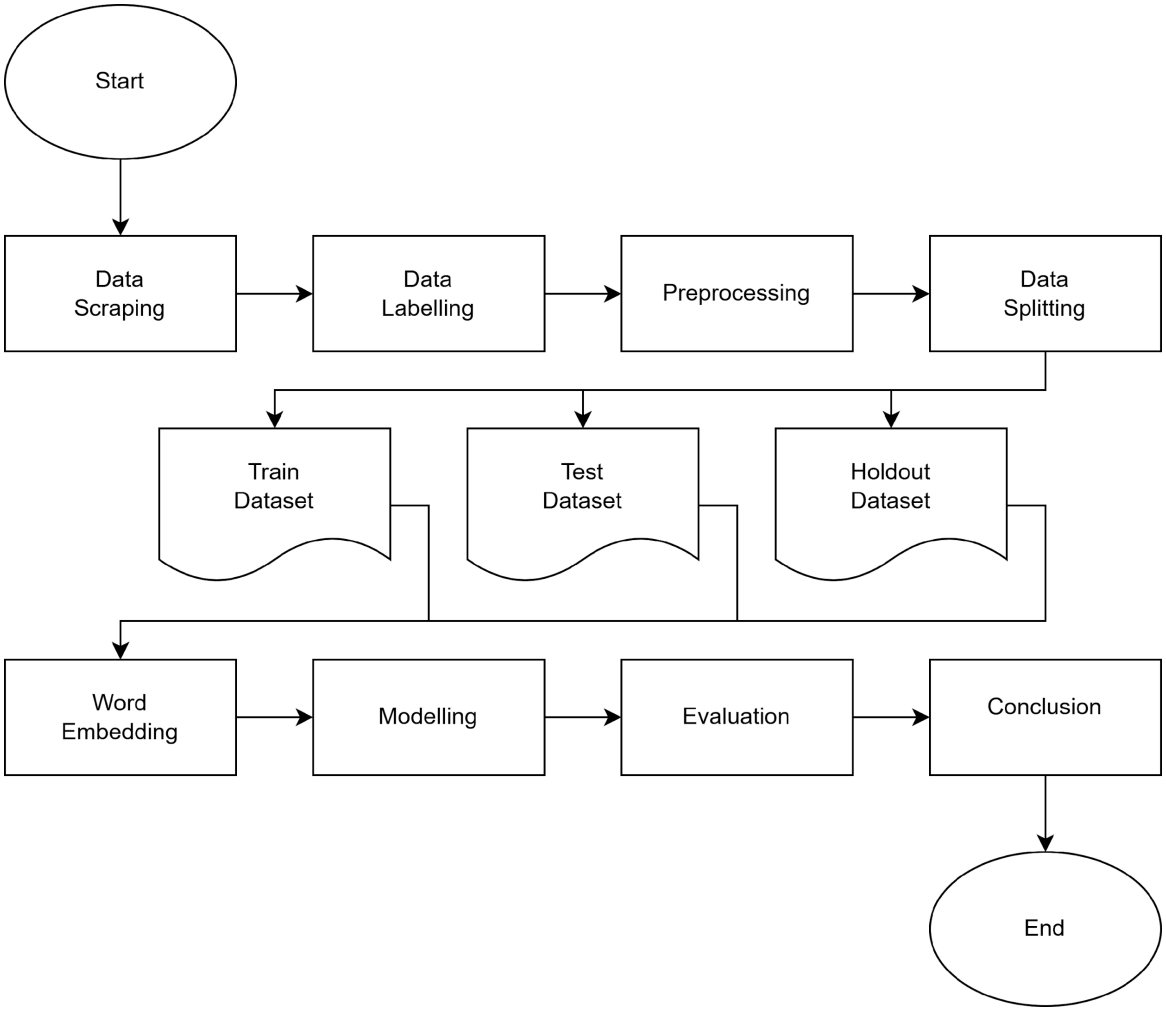

Fig. 1: Research workflows.

machine learning models for spam detection in YouTube comments, including Gaussian Naive Bayes, Logistic Regression, K-Nearest Neighbors, Multilayer Perceptron, Support Vector Machine, Random Forest, Decision Tree, and a voting classifier. Their results showed that Random Forest achieved nearly perfect performance, with an average precision of 100% and an AUC-ROC of 0.9841, highlighting the potential of ensemble models in this domain. These insights further demonstrate the versatility of both deep learning and traditional machine learning approaches in spam detection tasks, reinforcing the importance of exploring diverse methodologies.

3. Research and Methodology

The research methodology follows a structured workflow illustrated in Fig. 1. The process begins with problem identification, clarifying the scope and objectives of detecting gambling comments on YouTube. Data preparation involves scraping relevant data from YouTube and manually labeling comments for classification. Preprocessing is then performed, which includes removing punctuation, special characters, and stopwords to clean and normalize the text data. Following preprocessing, data splitting divides the dataset into training, testing, and holdout subsets to ensure unbiased evaluation.

Word embedding techniques are subsequently applied: tree-based boosting models (XGBoost, LightGBM, CatBoost) utilize TF-IDF features, while Word2Vec embeddings are employed for LSTM and GRU models. This step transforms textual data into numerical vectors suitable for model input. Modeling then implements and trains the chosen algorithms, using the prepared features. Finally, evaluation is conducted using appropriate metrics (such as precision, recall, and F1-score) to compare the models' effectiveness. The structured workflow ensures that each step builds on the previous one, enabling comprehensive analysis and reliable performance measurement.

3.1. Dataset Description

The dataset used in this study was collected by scraping comments from 24 YouTube videos spanning a diverse range of topics including daily life, entertainment, fashion, film, finance, health, hobbies, politics, sport, technology, and travel, to ensure high variability in the data. The dataset comprises 11,673 rows divided into three subsets: 70% for training, 20% for testing, and 10% for holdout evaluation. Notably, gambling comments represent approximately 10% of the total dataset, making them the minority class in this distribution. The specific class distributions are as follows: the training set contains 7,454 non-gambling and 717 gambling-labeled comments, the test set includes 2,126 non-gambling and 209 gambling-labeled comments, and the holdout set comprises 942 non-gambling and 225 gambling-labeled comments. Examples of non-gambling comments include casual discussions such as "coach nova plis suruh pda bljr sepak penalti asli biar gk bapuk2 😢" (coach nova please tell the PDA to learn penalty kicks so they don't suck 😢) and general observations like "hukum ekonomi barang ya murah barang ya mahal aja udah" (economic law: goods are either cheap or expensive, that's it). In contrast, gambling-related comments typically contain promotional content and winning claims, such as "**P L U T O 8 8** emang beda main langsung menang gede 🔥 🔥" (PLUTO 88 is really different, play and win big immediately 🔥 🔥) and withdrawal testimonials like "wd lancar banget nggak pake ribet" (withdrawal is very smooth without any hassle). To ensure label reliability, each data point was annotated based on agreement among multiple authors, with inter-annotator reliability measured using Cohen's kappa coefficient [23], a statistical measure used to evaluate inter-rater agreement for categorical items. This structure promotes consistent labeling and enables robust evaluation of model performance.

3.2. Preprocessing

Minimal text preprocessing was applied to preserve key information relevant to gambling content. Text labeled as gambling often contains unique character formatting, emojis, and numeric combinations that help identify online gambling brands (e.g., dora77, pluto88, king328). Excessive cleaning could risk removing these features and significantly reduce the dataset's informative content. Consequently, only basic preprocessing steps were performed: punctuation and special characters were removed, while letters, digits, and spaces were retained. The text was then tokenized, and stopwords were removed to emphasize the most meaningful words for model input.

3.3. Word Embedding

Numerical text processing plays a crucial role in enabling machine learning algorithms to interpret and analyze textual data. In this study, two primary methods were implemented to convert text into numerical form Term Frequency-Inverse Document Frequency (TF-IDF) and Word2Vec. TF-IDF transforms text into a sparse matrix based on the frequency of words within and across documents, emphasizing the importance of rare but informative terms. On the other hand, Word2Vec generates dense vector representations by capturing the semantic relationships between words based on their surrounding context.

A. TF-IDF

TF-IDF is a statistical model that assesses the importance of words within a collection of documents. It is derived by multiplying two measurements: the TF matrix, which is a two-dimensional array, and the IDF vector, which is a one-dimensional array. Numerous enhancements have been suggested by researchers to refine the traditional TF-IDF model [24]. TF-IDF is straightforward and efficient to calculate, especially in comparison to the more intricate neural embedding techniques. Term Frequency (TF) quantifies how often a term t appears in a document d, often normalized to avoid bias toward longer documents, as shown in (1)

$$TF(t, d) = f_{t,\,d} \,/\, \Sigma_k \, f_{k,\,d} \tag{1}$$

where $f_{t,\,d}$ denotes the raw count of term t in document d, and the denominator is the total word count in document d_t. This step highlights the significance of a word within a particular document. Conversely, Inverse Document Frequency (IDF) assesses the overall importance of a term t throughout a set of documents N. IDF is designed to decrease the weight of frequently occurring words across documents and increase the weight of less common, more informative terms. It is defined in (2)

$$IDF(t) = log(N \,/\, (1 + df_t\,)) \tag{2}$$

where N represents the total number of documents, and t is the number of documents containing the term. Adding 1 to the document frequency count (known as Laplace smoothing) avoids division by zero for terms absent in any document or in novel documents. Logarithms are utilized to diminish the influence of excessively rare words on the weights. The TF-IDF score for a term *t* in document d_t is calculated as the product of its TF and IDF values given in (3)

$$TFIDF(t, d) = TF(t, d) \times IDF(t) \tag{3}$$

This combined measure effectively highlights terms that are frequently used in a specific document but are uncommon across others, thus serving as significant indicators of the document's content [10].

B. Word2Vec

Unlike the frequency-based TF-IDF, Word2Vec is a predictive technique that uses a neural network to learn dense vector representations of words from a large text corpus [25]. These representations, called *word embeddings*, capture semantic and syntactic relationships between words. This means that words appearing in similar contexts will have proximate vector representations in the vector space. A key advantage of these embeddings is their ability to capture analogies, famously demonstrated by the vector operation vector ('King') – vector ('Man') + vector ('Woman'), which results in a vector very close to the vector ('Queen'). This capability makes Word2Vec highly powerful for tasks requiring a deeper understanding of meaning.

Word2vec comprises two architectures, Continuous Bag of Words (CBOW) and Skip-gram. Continuous Bag of Words (CBOW) predicts the current (target) word based on the surrounding context words [26]. For example, given the context 'the cat sat on the …', the CBOW model would be trained to predict the word 'mat'. The context words are usually averaged before predicting the target word, meaning the order of the words in the context is not explicitly preserved in this averaging step. CBOW is generally faster to train and performs well for frequently occurring words. The CBOW model takes one-hot encoded context words, projects them into an N-dimensional embedding space using an input weight matrix (V), averages these context embeddings, and then uses an output weight matrix (U) and softmax to predict the target word. The goal is to maximize the probability of the target word given its context.

Skip-gram works in the opposite way to CBOW: it predicts surrounding context words based on a single target word. Using the same example, given the target word 'sat', the Skip-gram model would be trained to predict context words like 'the', 'cat', 'on', and 'the'. Skip-gram typically performs better for less frequent words and larger datasets and often captures finer-grained semantic relationships. This architecture gives more weight to closer context words. The objective function for Skip-Gram is to maximize the probability of a context word w_{t+j} given a target word w_t given in (4)

$$max \prod_{t=1}^{T} \coprod_{-m \le j \le m,\ j \ne 0} P(w_{t+j} \mid w_j) \tag{4}$$

where m is the size of the context window [26][27]. The input of Skip-Gram is a one-hot encoded target word, projected onto its embedding using V. This embedding is then used together with U and softmax to predict multiple context words in a window [27].

To make the training process efficient on large corpora, Word2Vec is often implemented with optimization techniques such as *negative sampling*. Instead of updating the weights for all words in the vocabulary at each iteration, negative sampling only updates the weights for the correct context words (*positive samples*) and a small number of random, incorrect words (*negative samples*) [28]. During backpropagation, both V and U are updated to maximize the joint probability of the context words given the target word.

3.4. Model Selection

A. XGBoost

XGBoost (Extreme Gradient Boosting) is an optimized and scalable implementation of Gradient Boosting, known for its speed and performance. This algorithm has been widely adopted due to its success in various machine learning challenges [29]. It optimizes the following objective at each iteration as given in (5)

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{5}$$

where l is the loss function (e.g., log loss), and $\Omega(f_t)$ is a regularization term given in (6)

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{6}$$

Where T is the number of leaves in the decision tree, and w_j is the weight of leaf j. This regularization helps in selecting a simpler and more predictive function [29]. XGBoost uses a second-order Taylor expansion of the loss function to optimize the objective more efficiently and accurately during additive training [29]. The algorithm has a novel algorithm to handle sparse data (e.g., from TF-IDF or one-hot encoding) efficiently. The algorithm learns a default direction for missing values at each node, processing only non-missing entries, which significantly speeds up computation on sparse datasets [29].

B. LightGBM

LightGBM is another GBDT (Gradient Boosting Decision Tree) framework developed by Microsoft, designed for high efficiency, speed, and lower memory usage, especially on large datasets. This algorithm often trains faster than XGBoost [30] Unlike XGBoost which uses level-wise (horizontal) tree growth, LightGBM applies a leaf-wise growth strategy, which is expanding branches from leaves with the largest loss reduction [31]. Although this method can significantly improve accuracy, it also has the potential to cause overfitting, so it needs to be combined with regularization techniques. The objective function is given in (7)

$$min \sum_{i=1}^{n} l(y_i, \hat{y_i}) + \sum_{k=1}^{K} \Omega(f_k) \tag{7}$$

To improve training efficiency without sacrificing accuracy, LightGBM introduces the Gradient-based One-Side Sampling (GOSS) technique, which retains all data with large gradients (generally representing data that has not been well-learned), and only randomly samples data with small gradients (instances that have been relatively mastered by the model) [30]. In addition, this algorithm also uses the Exclusive Feature Bundling (EFB) technique, which performs very effective dimensionality reduction for high-dimensional and sparse data such as the results of TF-IDF. The main purpose of EFB is to reduce the number of features by combining mutually exclusive features, namely features that rarely have non-zero values together in a single sample [30]. With the combination of GOSS and EFB, LightGBM is not only fast but also memory and computationally efficient, making it ideal for text-based and big data classification applications.

C. CatBoost

CatBoost, short for Categorical Boosting, is a gradient boosting algorithm developed by Yandex that is specifically designed to handle categorical features effectively and reduce overfitting caused by target leakage [32]. Unlike traditional boosting models, CatBoost introduces advanced techniques such as Ordered Boosting, Permuted Target Statistics (Ordered TS), and Oblivious Decision Trees, making it particularly powerful for classification tasks involving categorical or sparse textual data. Ordered Boosting addresses the issue of target leakage that commonly arises in standard gradient boosting. In this approach, CatBoost applies a random permutation to the training data

and ensures that, for each training instance, the model used to estimate its gradient has been trained only on the preceding samples in that permutation [32].

Let π denote a random permutation of the dataset indices. Then, for the i-th instance, the prediction used to compute the gradient is based only on instances indexed by $\{\pi_1, \pi_2, ..., \pi_{i-1}\}$. This ensures that the model does not "see" the true label of the current instance when computing the gradient, thereby producing less biased residuals and improving the model's generalization ability. Handling categorical features with high cardinality is a common challenge in machine learning. Instead of using typical approaches like one-hot encoding or mean encoding, CatBoost employs Ordered Target Statistics, a method that computes the target-based numeric representation of each categorical value using only preceding samples in the permutation. For a categorical feature x_c , the target statistic for the i-th instance is computed as given in (8)

$$TS_i = \frac{\sum_{j<1} y_j \cdot 1_{x_j=x_i} + a}{\sum_{j<1} y_j \cdot 1_{x_j=x_i} + b} \tag{8}$$

here y_j is the target value of the i-th sample, $1_{x_j=x_i}$ is an indicator function that returns 1 if the category values match, a and b are smoothing parameters to prevent division by zero and reduce overfitting.

This formulation ensures that the transformation of categorical features does not leak any target information from the current instance, which is critical for model reliability. When combined with Ordered Boosting, this mechanism effectively avoids overfitting from categorical encodings [32]. Another distinctive feature of CatBoost is its use of Oblivious Decision Trees. Unlike traditional decision trees where splits can vary at each node, an oblivious tree applies the same split condition at every node at the same level, resulting in a balanced and symmetric structure.

D. LSTM

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) specifically designed to capture long-term dependencies in sequential data. Unlike standard RNNs, which suffer from the vanishing gradient problem during training, LSTM introduces a memory cell mechanism that allows gradients to flow unchanged over long sequences. This makes it highly effective for learning from data with long-range temporal patterns, such as natural language or time series data [33]. The internal structure of an LSTM includes three gates that regulate the flow of information: the input gate, the forget gate, and the output gate.

The input gate i_t determines how much of the new input x_t should influence the current cell state, and is computed as given in (9)

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{9}$$

The forget gate f_t decides what portion of the previous cell state C_{t-1} should be retained as shown in (10)

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{10}$$

The output gate o_t controls how much of the cell state should be exposed to the next layer or output, and is calculated as in (11)

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{11}$$

In these equations (8), (9), (10), W_i, W_f, and W_o are weight matrices; b_i, b_f, and b_o are bias vectors; h_{t-1} is the hidden state from the previous time step; and σ represents the sigmoid activation function, which outputs values in the range [0, 1]. This gating mechanism allows LSTM to selectively update and preserve information over time, making it a powerful model for sequential tasks [33].

E. GRU

The Gated Recurrent Unit (GRU) neural network is a specialized form of the Recurrent Neural Network (RNN), crafted to efficiently keep track of long-term information dependencies. This model streamlines the Long Short-Term Memory (LSTM) neural network. While LSTM employs three gates forget, input, and output, GRU merges these functions into two gates: the update gate and the reset gate. This composition minimizes the number of parameters, streamlines the network architecture, reduces training expenses, and enhances the model's capability to retain information across extended sequences. The GRU architecture includes an input layer, an output layer, and hidden layers composed of GRU units. For an input sequence $(x_1, x_2, ..., x_3)$, the update gate, reset gate, and hidden state of a GRU unit at time t are defined by the following equations (12)-(16) [34].

- Reset Gate

$$r_t = \sigma(W_r[x_t, h_{t-1}]) \tag{12}$$

- Update Gate

$$z_t = \sigma(W_z[x_t, h_{t-1}]) \tag{13}$$

- Candidate Hidden State

$$n_t = tanh(W_h[x_t, r_t, h_{t-1}]) \tag{14}$$

- Hidden State

$$h_t = (1 - z_t)h_{t-1} + z_t \cdot n_t) \tag{15}$$

- Output

$$y_t = \sigma(W_o h_t) \tag{16}$$

where r_t represents the reset gate output at time t, controlling how much past information is forgotten. z_t represents the update gate, deciding how much of the candidate hidden state n_t is retained for the new hidden state: h_t, W_r, W_h , and $W_o$are the weight matrices associated with each gate or transformation. σ denotes the sigmoid activation function. tanh is the hyperbolic tangent activation function. The GRU's simplified gating mechanism enables it to efficiently manage long-term dependencies by continuously discarding irrelevant information and updating the hidden state.

3.5. Evaluation Metrics

To assess the model's performance, this research uses several widely accepted evaluation metrics given in (17)-(20)

- **Accuracy** measures the proportion of correct predictions:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

- **Precision** indicates the correctness of positive predictions:

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

- **Recall** measures the ability to detect true positives:

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

- **F1-Score** combines precision and recall into a single metric:

Table 1: LSTM Network Architecture.

Layer	Input Dimensions	Output Dimensions	Function
Embedding	(64, seq_len)	(64, seq_len, 100)	Maps word indices to 100-dimensional Word2Vec vectors
Dropout	(64, seq_len, 100)	(64, seq_len, 100)	Applies dropout to the embedded sequences to prevent overfitting.
LSTM	(64, seq_len, 100)	(64, seq_len, 128)	Processes Sequence data and outputs hidden states
Dropout	(64,128)	(64,128)	Applies dropout to the last hidden state of the LSTM output before the fully connected layer.
Fully Connected	(64, 128)	(64, 1) → (64,)	Maps hidden state to binary classification output

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{20}$$

where TP, TN, FP, and FN stand for True Positives, True Negatives, False Positives, and False Negatives, respectively.

4. Experiments and Results

4.1. Experimental Setup, Model Training and Testing

All tasks including dataset creation, preprocessing, and modeling were conducted in the Kaggle environment. Kaggle's platform provides a convenient combination of hardware and software resources, supporting efficient experimentation and model development. Data preparation and preprocessing tasks were executed using Kaggle's shared CPU resources, while the T4x2 GPU was utilized for training the LSTM and GRU models, accelerating deep learning workloads significantly. Python was the primary programming language used, and common libraries such as pandas, NumPy, scikit-learn, and Pytorch, were employed to implement the various preprocessing steps, machine learning models, and evaluation metrics. The training and evaluation results of each model are presented in tabular form to facilitate comparison. The results include training time, test set metrics (precision, recall, F1-score), and holdout set metrics (precision, recall, F1-score). To address the extreme class imbalance in the dataset, models are sorted in descending order of holdout set F1-score, highlighting models with better generalization to unseen data. Additionally, learning curves for the LSTM and GRU models (loss vs epoch) are presented to illustrate training convergence and stability.

A. Tree-Based Boosting Models

The tree-based boosting models used in this study include LightGBM, XGBoost, and CatBoost. These models were employed in their standard, off-the-shelf configurations, without any manual hyperparameter tuning. Each model was trained on the TF-IDF vectorized representation of the training dataset, which converts textual data into numerical features suitable for traditional machine learning algorithms. This approach allows the models to capture term importance while maintaining computational efficiency. The default settings provided by the respective libraries were used to ensure a fair baseline comparison with deep learning methods.

B. Deep Learning Models

The architectural details of the LSTM and GRU models are summarized in Table 1 and Table 2. Each model receives tokenized and padded text inputs, maps them through an embedding layer initialized with pre-trained Word2Vec vectors, and processes the sequential data using its respective recurrent unit. To mitigate overfitting, dropout layers are consistently applied within both models. The final hidden state of the sequence is then passed through a fully connected layer to produce a binary output. The main distinction between the models is that they use different types of recurrent units, with one employing GRU and the other LSTM.

Table 2: GRU Network Architecture.

Layer	Input Dimensions	Output Dimensions	Function
Embedding	(64, seq_len)	(64, seq_len, 100)	Maps word indices to 100-dimensional Word2Vec vectors
Dropout	(64, seq_len, 100)	(64, seq_len, 100)	Applies dropout (p = 0.5) to embeddings
GRU	(64, seq_len, 100)	(64, seq_len, 128)	Processes sequence data and outputs hidden states
Dropout	(64, 128)	(64, 128)	Applies dropout (p = 0.5) to the last time step
Fully Connected	(64, 128)	(64, 1) → (64,)	Maps hidden state to binary classification output

Table 3: Evaluation Result for Each Model.

Model	Precision (Test/Holdout)	Recall (Test/Holdout)	F1-Score (Test/Holdout)	Time (s)
LightGBM	0.9613 / 0.8912	0.9632 / 0.8886	0.9612 / **0.8737**	0.42
XGBoost	0.9678 / 0.8745	0.9687 / 0.8680	0.9668 / **0.8428**	0.74
CatBoost	0.9739 / 0.8650	0.9743 / 0.8569	0.9729 / **0.8247**	34.36
LSTM	0.9610 / 0.9699	0.9426 / 0.5733	0.9517 / **0.7207**	12.80
GRU	0.9849 / 0.9826	0.9378 / 0.5022	0.9340 / **0.6647**	11.16

4.2. Analysis of Results

A. Model Performance Evaluation

Table 3 provides a comprehensive evaluation of each model's performance on both test and holdout datasets, considering precision, recall, F1-Score, and training time. In evaluating model performance for this classification task, F1-score emerges as the most critical metric due to the extremely imbalanced nature of the dataset. When dealing with imbalanced datasets, accuracy alone can be misleading as a model could achieve high accuracy simply by predicting the majority class, while precision and recall individually provide incomplete pictures of model performance. F1-score, being the harmonic mean of precision and recall, offers a balanced assessment that penalizes models with poor performance on either metric, making it particularly valuable when both false positives and false negatives carry significant costs. The dataset's severe class imbalance means that a model achieving high precision but low recall would fail to identify many positive instances, while a model with high recall but low precision would generate excessive false alarms. F1-score effectively captures this trade-off by requiring models to maintain reasonable performance across both dimensions. LightGBM demonstrates strong holdout performance, achieving a precision of 0.8912 and recall of 0.8886, with an F1-score of 0.8737. This indicates a balanced and consistent generalization to unseen data, making it a highly effective model. XGBoost follows closely with holdout precision of 0.8745, recall of 0.8680, and an F1-score of 0.8428, suggesting its robustness in handling new samples. Both LightGBM and XGBoost exhibit impressive stability across different datasets, which is crucial for reliable real-world deployment, and achieve their results with relatively fast training times (0.42s and 0.74s, respectively). CatBoost achieves respectable holdout performance, with precision at 0.8650, recall at 0.8569, and an F1-score of 0.8247. While its training time is considerably higher at 34.36s, its consistency positions CatBoost as a competitive and reliable alternative, though slightly behind LightGBM and XGBoost in terms of holdout metrics. In contrast, the deep learning models (GRU and LSTM) show promising test performance but exhibit significant challenges with generalization to the holdout set. GRU achieves exceptionally high test precision (0.9849) and recall (0.9378); however, its holdout recall drops drastically to 0.5022, with an F1-score of 0.6647. Similarly, LSTM performs well on the test set with a precision of 0.9610 and recall of 0.9426, but its holdout recall is considerably lower at 0.5733, resulting in an F1-score of 0.7207. These disparities highlight a clear overfitting issue for the deep learning models.

B. Training Dynamics and Convergence

This subsection analyzes the learning behavior of the deep learning models during training. All deep learning experiments were consistently conducted for a total of 20 epochs. The progression of training and validation loss over these epochs is critically examined through their respective loss curves. These visualizations provide key insights into model convergence, stability, and potential overfitting.

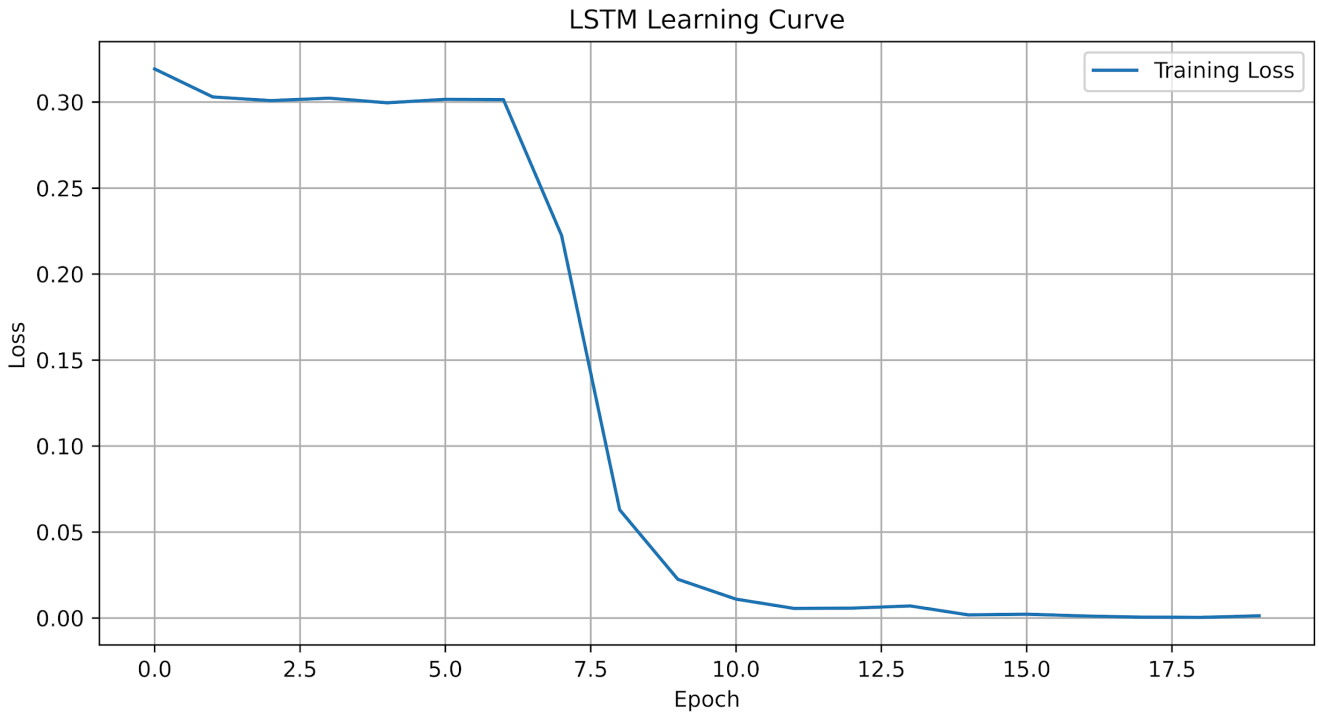

Fig. 2: LSTM learning curve (loss over epochs).

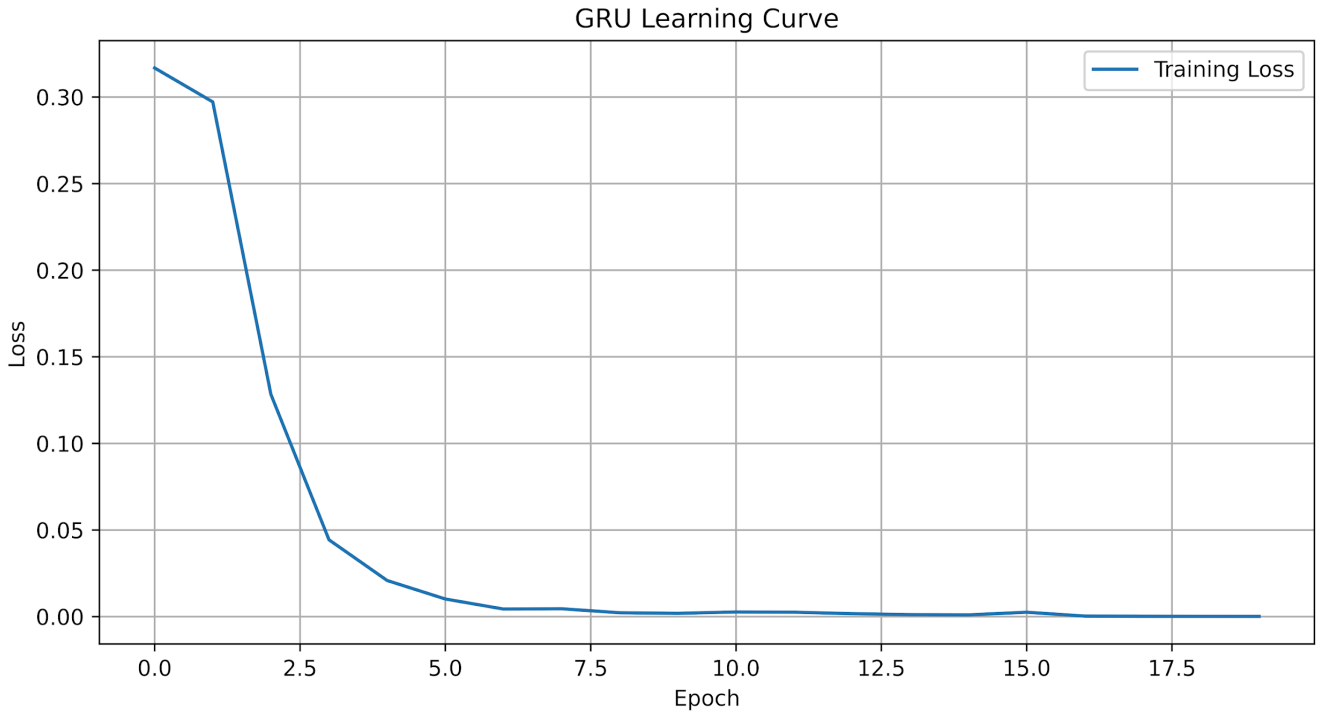

Fig. 3: GRU learning curve (loss over epochs).

The LSTM learning curve, illustrated in Fig. 2, initially exhibits a relatively steady training loss around 0.30 for the first 5 to 6 epochs, indicating a period of slower initial learning. Following this, a sharp decline in training loss is observed, dropping significantly between epoch 6 and epoch 9, before converging to near-zero loss (below 0.01) by approximately epoch 10. This delayed but ultimately strong convergence reflects LSTM's more complex internal mechanisms, which may require more iterations to fully capture intricate sequential patterns. Conversely, the GRU learning curve, presented in Fig. 3, demonstrates a much more rapid and steep decline in training loss from the very first epoch. The loss quickly drops from over 0.30 to below 0.05 by epoch 3-4, and reaches near-

zero loss (below 0.01) by approximately epoch 6. This swift convergence suggests that the GRU model, with its comparatively simpler architecture, is capable of learning essential patterns more quickly from the training data.

For both deep learning models, the rapid convergence to near-zero training loss, coupled with the significant drop in recall on the holdout set (as detailed in Section 4.2.1), strongly indicates a phenomenon of overfitting. This suggests that while the models are highly effective at memorizing the training data, they struggle to generalize the learned features to unseen data. The learning curves in Fig. 2 and Fig. 3 underscore the critical importance of regularization techniques to prevent such overfitting and improve generalization. While dropout has already been applied in both GRU and LSTM models, additional strategies such as early stopping, more aggressive dropout rates, or further tuning of regularization parameters may be necessary to enhance their generalization performance and bridge the gap between training and holdout metrics.

C. Feature Insight

Further insights into the distinctive characteristics of the comment categories are provided by the TF-IDF word frequency analysis. Fig. 4 and 5 reveal key terms that differentiate gambling from non-gambling comments. In Fig. 4 for non-gambling discussions, words like 'bang', 'yg', and 'kiko' frequently appear. These terms are often specific to general conversation, potentially indicating popular culture references, common exclamations, or informal expressions prevalent in everyday online interactions. Conversely, gambling-related comments in Fig. 5 are strongly characterized by terms such as 'main' (play), 'rezeki' (fortune/luck), and 'banget' (very/really), which are directly associated with the activity. 'Main' explicitly points to the act of gambling, 'rezeki' highlights the hope for winnings, and 'banget' often emphasizes the intensity or desire related to the outcome. This clear distinction in vocabulary underscores the unique linguistic patterns within each class, offering a deeper understanding of the textual features that contribute to the classification models' performance. A noticeable aspect of the analysis is the emergence of several high-frequency words, such as 'banget', 'ya', 'aja', 'gw', 'ga', 'gak', 'lu', and 'udah', which often function as stopwords in the Indonesian language. While these words are crucial for grammatical structure and conveying subtle meanings in human communication, their high prevalence across both gambling and non-gambling contexts, as seen in their relatively high TF-IDF scores, suggests they might introduce noise or dilute the distinctiveness of more discriminative terms. Their presence can sometimes inflate the importance of less relevant words, potentially affecting model precision. For future research, it is recommended that the handling of stopwords be revisited with stricter and more nuanced methods. This could involve exploring custom stopword lists tailored to the specific domain, implementing more aggressive stemming or lemmatization techniques, or utilizing advanced word embedding models that can better distinguish between semantically rich words and functional terms, thereby enhancing the models' ability to focus on truly indicative language patterns. Ultimately, a more refined approach to stopword management is anticipated to yield more robust and accurate classification, particularly benefiting the generalization capabilities of the deep learning models.

4.3. Qualitative Prediction Analysis

This section presents an in-depth qualitative prediction analysis of the two most effective machine learning models, with LGBM representing traditional machine learning and LSTM representing deep learning. The goal is to move beyond quantitative metrics and delve into why each model makes certain predictions, examining examples of both correct and incorrect classifications. This in-depth look at the predictive behavior of each approach will illuminate their strengths and weaknesses, providing valuable insights into practical application for this specific text classification task.

)

Based on the qualitative prediction analysis results presented in Table 4 and 5, both LGBM and LSTM models demonstrate distinct predictive behaviors that reveal their underlying strengths and limitations. The LGBM model shows strong performance in identifying clear sentiment indicators, correctly classifying comments with explicit emotional expressions such as "wede aman nggak prnh…" and "gak bosan main KING328…" while successfully recognizing neutral content like plotwist references and popular culture discussions. However, the model struggles

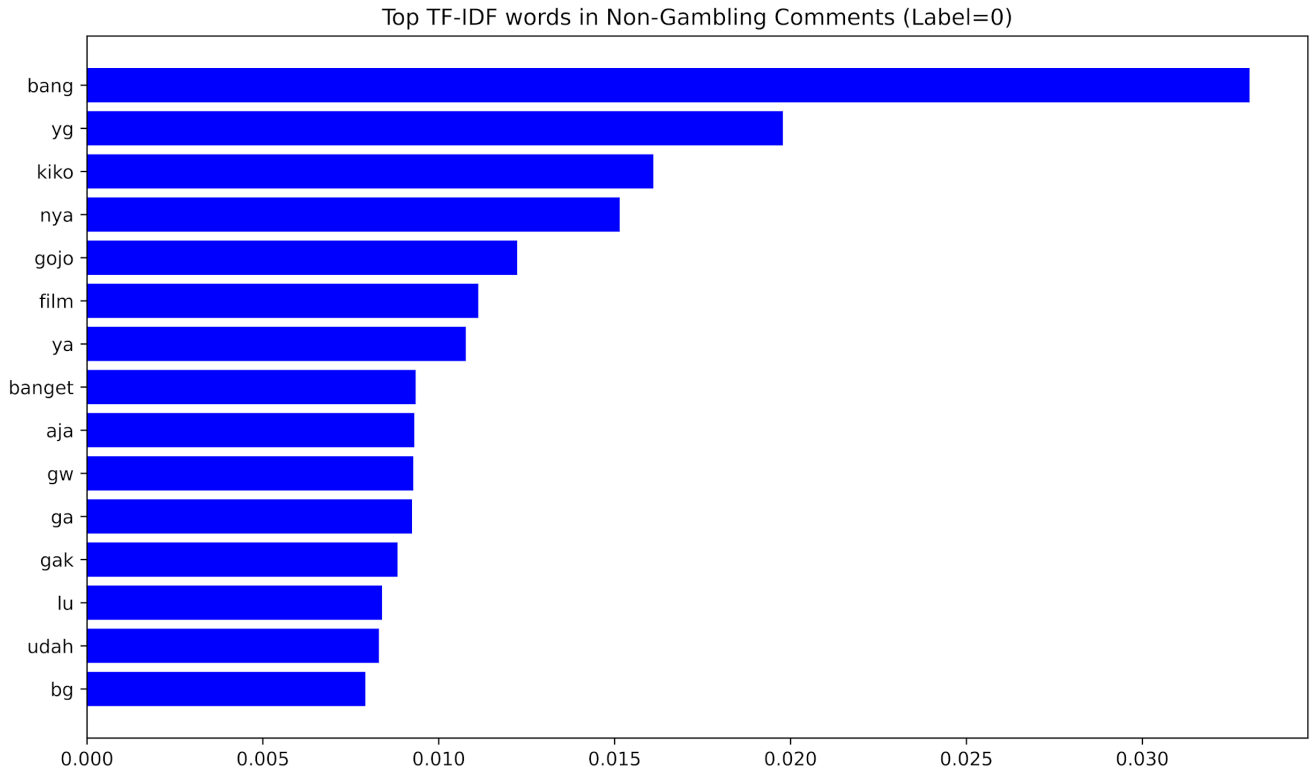

Fig. 4: TF-IDF words in non-gambling comments.

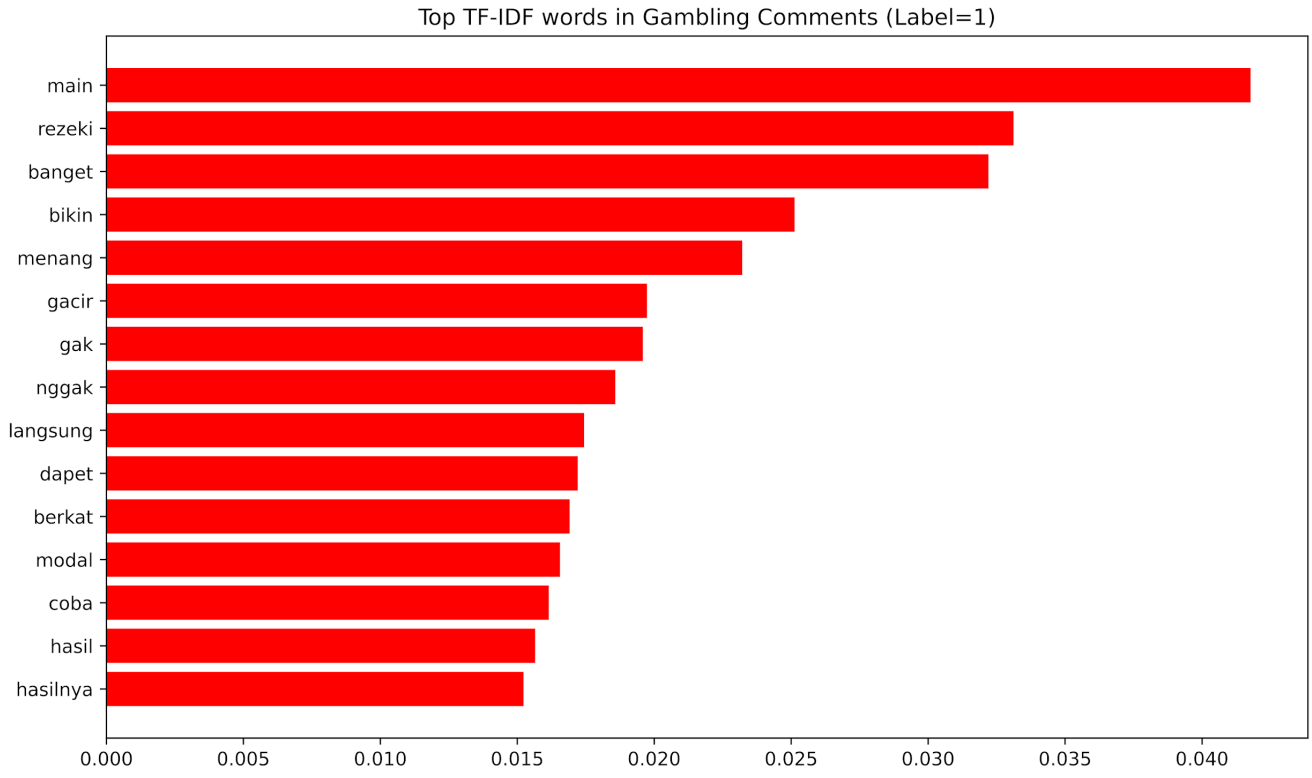

Fig. 5: TF-IDF words in gambling comments.

with nuanced language patterns, incorrectly classifying comments containing gaming terminology ("sketer muncul auto jepek…") and complex contextual expressions that require deeper semantic understanding. The LSTM model exhibits superior contextual awareness, accurately predicting comments with subtle emotional undertones like

Table 4: LGBM Qualitative Prediction Analysis

Prediction Type	Label	Comments
Correct	1	"wede aman nggak prnh…", "gak bosan main ***KING328***…", "modal 50 ribu jepe lgsng wede…"
	0	"plotwist 😂…", "955 busett make kaya gitu…", "manfaat populer ya negara susah fakta…"
Incorrect	1	"sketer muncul auto jepek…", "keberuntungan berpihak ***KING328*** ⚡ 💰🔥…", "**MIYA88** bantu gue lewatin tanggal tua keluh 💸 ✨…"
	0	"berisik cuan rungkad iya 🌪…", "mantap abang ku petualangan terbaik ✨👆…", "bahas cari duit asing bikin ngo lsm…"

Table 5: LSTM Qualitative Prediction Analysis

Prediction Type	Label	Comments
Correct	1	"modal gacur lgsng jepe brusan…", "rezeki nggak kemana makasih ***KING328***…", "gacir bener maen udh jepe…"
	0	"crows zero film surealis yak…", "ijin rangkumkan pertandingan lihat 1 timnas korsel individu tim timnas 17thn indonesia 2 kalah tekni…", "p balap…"
Incorrect	1	"**MONA4D** emang top editing rapi vibesnya asik sukses 🐰…", "modal kaya coba aja ***KING328*** 🔥 💰💎…", "alhamdulillah **MONA4D** menarik cerita relate 🫂…"
	0	"keren banget bg puas menonton video abang nyaman memuaskan salam jateng 🔌…", "udh keras gini mlah nambah suaranya halus…", "berisik cuan rungkad iya 🌪…"

"modal gacur lgsng jepe brusan…" and "rezeki nggak kemana makasih KING328…" while correctly identifying neutral content such as film discussions and technical gaming references. Nevertheless, the LSTM model encounters difficulties with ambiguous expressions and complex semantic relationships, as evidenced by its misclassification of comments containing mixed emotional signals and culturally specific references. The analysis reveals that while both models demonstrate competency in straightforward sentiment detection, their failure cases highlight the ongoing challenges in natural language processing, particularly in handling colloquial expressions, gaming-specific terminology, and culturally embedded linguistic patterns that require sophisticated contextual interpretation.

5. Conclusion

This study conducted a comprehensive comparison of machine learning approaches for handling extremely imbalanced datasets, evaluating both traditional gradient boosting methods and deep learning models within the Kaggle environment, leveraging its CPU and T4x2 GPU resources. The experimental results reveal significant insights into model performance, generalization capabilities, and practical deployment considerations for text classification, specifically focusing on the F1-score due to the severe class imbalance.

The gradient-boosting ensemble methods, particularly LightGBM and XGBoost, demonstrated superior overall performance and reliability. LightGBM achieved the highest holdout F1-score (0.8737) with balanced precision (0.8912) and recall (0.8886), indicating robust generalization to unseen data. XGBoost followed closely with comparable holdout metrics (F1-score: 0.8428, precision: 0.8745, recall: 0.8680), confirming the effectiveness of gradient-boosting approaches for imbalanced classification tasks. Both models also exhibited fast training times (0.42s and 0.74s, respectively). CatBoost also showed competitive performance (holdout F1-score: 0.8247) but with considerably higher computational costs during training (34.36s). All tree-based models were trained on TF-IDF vectorized text without manual hyperparameter tuning.

In contrast, the deep learning models (GRU and LSTM) exhibited a clear pattern of overfitting despite achieving exceptional performance on the test set. The GRU model reached exceptionally high test precision (0.9849) and recall (0.9378) but suffered from severely degraded holdout recall (0.5022), resulting in a lower holdout F1-score (0.6647), indicating poor generalization. Similarly, LSTM showed strong test performance (precision: 0.9610, recall: 0.9426) but failed to maintain consistency on holdout data (holdout recall: 0.5733, F1-score: 0.7207). The learning curves for both GRU and LSTM revealed rapid convergence to near-zero training loss (below 0.01 by

approximately epoch 6 for GRU and epoch 10 for LSTM), suggesting they memorized training patterns rather than learning generalizable features, even with the application of dropout layers and pretrained Word2Vec embeddings.

The stark performance gap between test and holdout sets for deep learning models highlights a critical limitation when working with imbalanced datasets and limited data volumes. While these models can capture complex non-linear patterns, they require substantial amounts of training data to learn generalizable representations effectively. The dataset size in this study appears insufficient for deep learning approaches to reach their full potential, as evidenced by their rapid overfitting and poor generalization performance. This limitation, combined with their susceptibility to overfitting, makes them less suitable for practical deployment without extensive regularization techniques and significantly larger datasets. Qualitative analysis further revealed that while deep learning models showed superior contextual awareness, they struggled with ambiguous expressions and complex semantic relationships.

The gradient boosting methods, conversely, demonstrated more stable and consistent performance across different data distributions and proved to be proficient in identifying clear sentiment indicators but struggled with nuanced language. For real-world applications dealing with extreme class imbalance and limited dataset sizes, this study recommends prioritizing gradient boosting methods, particularly LightGBM, due to their superior generalization capabilities, computational efficiency, and ability to perform well with smaller datasets. Deep learning approaches may be more suitable when larger, more comprehensive datasets become available. Future work should explore advanced regularization techniques for deep learning models, including more aggressive dropout rates, early stopping, and data augmentation strategies to improve their generalization performance. Additionally, investigating ensemble approaches that combine the pattern recognition capabilities of deep learning with the robustness of gradient-boosting methods presents a promising research direction. The findings underscore the importance of evaluating models on truly independent holdout sets rather than relying solely on cross-validation or test set performance, particularly when dealing with imbalanced data where overfitting risks are elevated. Furthermore, future research should consider more nuanced stopword management and advanced word embedding models to enhance the distinctiveness of discriminative terms in text classification. It would also be beneficial to explore more sophisticated deep learning architectures, such as bidirectional models like BiLSTM and BiGRU, or even Transformer-based models, as these could potentially capture more complex contextual dependencies in imbalanced text data.

CRediT Authorship Contribution Statement

Agung Widiyanto: Conceptualization, Software, Formal Analysis, Resources, Investigation, Visualization, Writing – Original Draft, Writing – Review & Editing. **Mayesq Prameswari:** Writing – Original Draft, Writing – Review & Editing. **Muhammad Abdul Latief:** Supervision, Data curation, Methodology, Writing – Original Draft, Writing – Review & Editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Declaration of Generative AI and AI-assisted Technologies in The Writing Process

The authors used generative AI to improve the writing clarity of this paper. They reviewed and edited the AI-assisted content and take full responsibility for the final publication.

References

[1] R. B. Perdana, Ardin, I. Budi, A. B. Santoso, A. Ramadiah, and P. K. Putra, "Detecting Online Gambling Promotions on Indonesian Twitter Using Text Mining Algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 8, pp. 942–949, 2024, doi: 10.14569/IJACSA.2024.0150893.

[2] A. Fahrudin *et al.*, "Online gambling addiction: Problems and solutions for policymakers and stakeholders in Indonesia," 2024, *EnPress Publisher, LLC*. doi: 10.24294/jipd.v8i11.9077.

[3] M. Nurseno, U. Aditiawarman, H. Al Qodri Maarif, and T. Mantoro, "Detecting Hidden Illegal Online Gambling on .go.id Domains Using Web Scraping Algorithms," *MATRIK: Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 23, no. 2, pp. 365–378, Mar. 2024, doi: 10.30812/matrik.v23i2.3824.

[4] D. Ananda, H. Y. Simatupang, and A. Srifauzi, "Trends in Online Gambling and Government's Inability to Achieve Decent Work in Indonesia: What Should We Do?," in *Proceeding of IROFONIC 2024 "Strengthening Partnership for Sustainable Development,"* 2024.

[5] J. Costa *et al.*, "Characterizing YouTube's Role in Online Gambling Promotion: A Case Study of Fortune Tiger in Brazil," in *Proceedings of the 17th ACM Web Science Conference 2025*, New York, NY, USA: ACM, May 2025, pp. 42–51. doi: 10.1145/3717867.3717905.

[6] J. Singer, "Stigmatisation of gambling disorder in social media: a tailored deep learning approach for YouTube comments," *Harm Reduct J*, vol. 22, no. 1, Dec. 2025, doi: 10.1186/s12954-025-01169-0.

[7] A. S. Xiao and Q. Liang, "Spam detection for Youtube video comments using machine learning approaches," *Machine Learning with Applications*, vol. 16, p. 100550, 2024, doi: 10.1016/j.mlwa.2024.100550.

[8] B. Sivaranjani, M. Divyadharshini, and L. Glory, "Harmful Content Detection on Social Media Platforms," *International Journal of Advanced Research in Computer and Communication Engineering Impact Factor*, vol. 8, 2025, doi: 10.17148/IJARCCE.2025.14344.

[9] M. M. P. Bhanddkar, "Machine Learning Base Spam Comments Detection on YouTube," *International Journal of Scientific Research in Engineering and Management*, vol. 9, no. 2, pp. 1–8, Feb. 2025, doi: 10.55041/IJSREM41346.

[10] H. Allam, L. Makubvure, B. Gyamfi, K. N. Graham, and K. Akinwolere, "Text Classification: How Machine Learning Is Revolutionizing Text Categorization," *Information*, vol. 16, no. 2, Feb. 2025, doi: 10.3390/info16020130.

[11] C. Hake, J. Weigele, F. Reichert, and C. Friedrich, "Evaluation of Artificial Intelligence Methods for Lead Time Prediction in Non-Cycled Areas of Automotive Production," Jan. 2025, [Online]. Available: http://arxiv.org/abs/2501.07317

[12] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access*, vol. 10, no. , pp. 99129–99149, 2022, doi: 10.1109/ACCESS.2022.3207287.

[13] M. C. Martinis, "Artificial Neural Networks: LSTM," *Encyclopedia of Bioinformatics and Computational Biology (Second Edition)*. Elsevier, Oxford, pp. 156–158, 2025, doi: 10.1016/B978-0-323-95502-7.00014-2.

[14] L. Feng, F. Tung, M. O. Ahmed, Y. Bengio, and H. Hajimirsadeghi, "Were RNNs All We Needed?," Oct. 2024, [Online]. Available: http://arxiv.org/abs/2410.01201

[15] Y. Goldberg and O. Levy, "Word2Vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," Feb. 2014, [Online]. Available: http://arxiv.org/abs/1402.3722

[16] E. Tufino, "Exploring large language models (LLMs) through interactive Python activities," *Physics Education*, vol. 60, no. 5, p. 55003, Jul. 2025, doi: 10.1088/1361-6552/adea28.

[17] H. J. Dai and C. K. Wang, "Classifying adverse drug reactions from imbalanced twitter data," *Int J Med Inform*, vol. 129, pp. 122–132, Sep. 2019, doi: 10.1016/j.ijmedinf.2019.05.017.

[18] S. Salman and X. Liu, "Overfitting Mechanism and Avoidance in Deep Neural Networks," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.06566

[19] N. Van Otten, "The Vanishing Gradient Problem, How To Detect & Overcome It," *Spot Intelligence*. Accessed: May 31, 2025. [Online]. Available: https://spotintelligence.com/2023/02/06/vanishing-gradient-problem/

[20] G. Airlangga, "Spam Detection on YouTube Comments Using Advanced Machine Learning Models: A Comparative Study," *Brilliance: Research of Artificial Intelligence*, vol. 4, no. 2, pp. 500–508, Oct. 2024, doi: 10.47709/brilliance.v4i2.4670.

[21] M. Min and D. A. Lee, "Illegal Online Gambling Site Detection Using Multiple Resource-Oriented Machine Learning," *J Gambl Stud*, vol. 40, no. 4, pp. 2237–2255, Jul. 2024, doi: 10.1007/s10899-024-10337-z.

[22] A. S. Xiao and Q. Liang, "Spam detection for YouTube video comments using machine learning approaches," *Machine Learning with Applications*, vol. 16, p. 100550, Jun. 2024, doi: 10.1016/j.mlwa.2024.100550.

[23] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educ Psychol Meas*, vol. 20, no. 1, pp. 37–46, Apr. 1960, doi: 10.1177/001316446002000104.

[24] K. Chen, Z. Zhang, J. Long, and H. Zhang, "Turning from TF-IDF to TF-IGM for term weighting in text classification," *Expert Systems with Applications*, vol. 66, pp. 245–260, 2016, doi: 10.1016/j.eswa.2016.09.009.

[25] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013, [Online]. Available: http://arxiv.org/abs/1301.3781.

[26] C. Yang and C. Ding, "Learning Word Embedding with Better Distance Weighting and Window Size Scheduling," Apr. 2024, [Online]. Available: http://arxiv.org/abs/2404.14631

[27] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo, "Evaluating word embedding models: methods and experimental results," *APSIPA Transactions on Signal and Information Processing*, vol. 8, p. e19, 2019, doi: 10.1017/ATSIP.2019.12.

[28] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, 2013, vol. 26, p. , [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.

[29] R. Zhu, T. Guo, G. Xu, R. Zhang, and S. Tesfamariam, "XGBoost-based probabilistic residual displacement demand prediction for self-centering viscous systems under near-fault ground motions," *Journal of Building Engineering*, vol. 111, p. 113572, 2025, doi: 10.1016/j.jobe.2025.113572.

[30] G. Ke et al., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems*, 2017, vol. 30, p. , [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[31] L. Gu, Y. He, H. Liu, Z. Wei, and J. Guo, "Metasurface meta-atoms design based on DNN and LightGBM algorithms," *Optical Materials*, vol. 136, p. 113471, 2023, doi: 10.1016/j.optmat.2023.113471.

[32] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "CatBoost: unbiased boosting with categorical features," in *Advances in Neural Information Processing Systems*, 2018, vol. 31, p. , [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

[33] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.

[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," 2014. http://arxiv.org/abs/1412.3555.