Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Spam detection for Youtube video comments using machine learning approaches

Andrew S. Xiao[a], Qilian Liang[b,*]

[a] Purdue University, Department of Computer Science, West Lafayette, 47907, IN, USA
[b] University of Texas at Arlington, Department of Electrical Engineering, Arlington, 76019, TX, USA

ARTICLE INFO

MSC:
03C45
68T05
68T50

Keywords:
Machine learning
Spam detection
Random forest
Naive Bayes
Logistic regression
Multi-layer perceptron
Voting classifier

ABSTRACT

Machine Learning models have the ability to streamline the process by which Youtube video comments are filtered between legitimate comments (ham) and spam. In order to integrate machine learning models into regular usage on media-sharing platforms, recent approaches have aimed to develop models trained on Youtube comments, which have emerged as valuable tools for the classification and have enabled the identification of spam content and enhancing user experience. In this paper, eight machine learning approaches are applied to spam detection for YouTube comments. The eight machine learning models include Gaussian Naive Bayes, logistic regression, K-nearest neighbors (KNN) classifier, multi-layer perceptron (MLP), support vector machine (SVM) classifier, random forest classifier, decision tree classifier, and voting classifier. All eight models perform very well, specifically random forest approach can achieve almost perfect performance with average precision of 100% and AUC-ROC of 0.9841. The computational complexity of the eight machine learning approaches are compared.

1. Introduction

YouTube, the world's largest video-sharing platform, has revolutionized the way we consume and engage with online content. With billions of users and countless videos uploaded daily, the platform fosters a dynamic ecosystem of user-generated content and corresponding interactive communities. A fundamental aspect of this environment is the comments section, where viewers can express their thoughts, engage in discussions, and provide feedback.

However, alongside these personalized contributions, YouTube comments are also plagued with an influx of spam content. Spam comments, characterized by repetitive irrelevant, promotional, or malicious content, pose a significant challenge for both YouTube and its users. They not only clutter the comments section but also undermine user experience, impede meaningful conversations, and degrade the quality of interactions. Adversarial and malicious accounts exploit platforms like Youtube to spread misinformation, promote inappropriate content, and exploit vulnerabilities that can lead to malware infections and other cybersecurity threats.

Companies hosting online platforms like YouTube have a vested interest in combating spam and ensuring a safe and engaging environment for their users. The proliferation of spam comments not only tarnishes the platforms' reputation but also hinders the effective exchange of ideas and engagement with authentic content. Therefore, the development of robust spam detection and classification systems has become a critical necessity.

To address these concerns, the utilization of deep learning models for YouTube comment classification offers promising avenues for filtering out spam and enhancing the overall user experience. By automating the identification of spam comments, content moderators and platform administrators can more efficiently and effectively maintain the quality and safety of the comments section. In this context, leveraging machine learning models for YouTube comment classification offers promising avenues to filter out spam and enhance the user experience. By automating the identification of spam comments, content moderators and platform administrators can efficiently and effectively maintain the quality, safety, and integrity of the comments section.

In the recent development of Artificial Intelligence (AI) and deep learning, generative AI has been a hot topic since the birth of ChatGPT (Fui-Hoon Nah, Zheng, Cai, Siau, & Chen, 2023). ChatGPT is based on large-scale language models, reinforcement learning from human feedback and in-context learning (Wu et al., 2023). Generative AI is an unsupervised machine learning approach which has very promising applications. It can generate multimodal information such as video,

* Corresponding author.
E-mail addresses: xiao318@purdue.edu (A.S. Xiao), liang@uta.edu (Q. Liang).

https://doi.org/10.1016/j.mlwa.2024.100550
Received 26 December 2023; Received in revised form 10 February 2024; Accepted 9 April 2024
Available online 16 April 2024

audio, text, image. Transformer is another hot topic which is based on attention mechanism (Vaswani et al., 2017). In Lin, Wang, Liu, and Qiu (2022), transformer and its variants X-transformers were overviewed. Transformer networks were introduced in Khan et al. (2022). The scaling of Transformers is a breakthrough technique for language models and the largest large language models (LLMs) has 100B parameters. Recently, scaling vision transformers to 22 billion parameters were reported (Dehghani et al., 2023). These large scale language processing models take several days for training and processing.

In this paper, we focus on the development and evaluation of machine learning models for classifying YouTube comments as legitimate (ham) or spam. The project aims to provide valuable insights into the effectiveness of various classifiers and contribute to the advancement of comment classification systems. By doing so, it seeks to not only improve user engagement and foster meaningful conversations but also prevent threats of malware infections and other vulnerabilities that can arise from spam comments. This research ultimately contributes to creating a safer and more secure environment for YouTube users, protecting their online experiences from potential risks and enhancing their overall satisfaction with the platform.

Various machine learning models are applied to YouTube comments spam detection in this paper, including Gaussian Naive Bayes, Logistic Regression, KNN Classifier, SVM Classifier, MLP Classifier, Decision Tree Classifier, Random Forest Classifier, and a Voting Classifier. Subsequently, the classifiers were trained on preprocessed data, and their performance was evaluated on a separate test dataset. The models were evaluated based on precision, accuracy, and AUC-ROC scores to assess their efficacy in discriminating ham from spam comments The study highlights variations in the classifiers' performance, providing insights into their individual strengths and weaknesses. The outcomes contribute to the development of effective comment classification systems, empowering content moderation and enhancing user experiences on online platforms.

The rest of this paper is organized as follows. In Section 2, related works on spam detection using machine learning are reviewed. In Section 3, we propose to apply seven machine learning approaches on spam detection for YouTube comments. Experimental results and performance analysis are presented in Section 4. We conclude this paper and discuss some future work in Section 5.

2. Related works

In the realm of spam detection, various studies have leveraged machine learning techniques to enhance the identification and mitigation of unwanted content. In Ahmed et al. (2022), a comprehensive overview of spam detection in both email and Internet of Things (IoT) platforms was presented. The study employed machine learning algorithms such as random forest, decision tree, Naive Bayes, and neural networks. For Twitter spam detection, Sun et al. Sun, Lin, Qiu, and Rimba (2022) used a combination of account-based and content-based feature extraction methods for data processing. In Danilchenko, Segal, and Vilenchik (2022), a message-passing algorithm based on users' graph structure was introduced for the identification of fake reviews (Danilchenko et al., 2022). In the survey conducted by Crawford, Khoshgoftaar, Prusa, Richter, and Al Najada (2015), the prevalence of supervised learning methods as the mainstream in spam detection using machine learning approaches was highlighted. In the context of IoT device security, five machine learning models were evaluated for spam decision making based on spam scores (Makkar et al., 2020). Manasa et al. (2022) adopted a swarm optimization approach for tweet-by-tweet spam detection. In Guo, Mustafaoglu, and Koundal (2023), bidirectional transformers and a machine learning classifier were applied to spam detection. Mashaleh, Ibrahim, Al-Betar, Mustafa, and Yaseen (2022) utilized the Harris Hawks Optimizer algorithm to optimize machine learning for spam email detection. In Bacanin et al. (2022), natural language processing and swarm intelligence were employed for spam email filtering. Shaaban, Hassan, and Guirguis (2022) proposed a deep convolutional forest for spam detection in text. In Grewal, Nijhawan, and Mittal (2022), feature optimization and machine learning were applied to email spam detection. Social media network spam detection was studied using machine learning in Niranjani, Agalya, Charunandhini, Gayathri, and Gayathri (2022). In summary, these studies showcase the diverse applications of machine learning techniques across different platforms and domains for effective spam detection. In this paper, we focus on spam detection on YouTube video comments.

3. Machine learning on spam detection for YouTube comments

We applied eight machine learning approaches to spam detection on the Youtube video comments, which include Gaussian Naive Bayes, logistic regression, K-nearest neighbors (KNN) classifier, multi-layer perceptron (MLP), support vector machine (SVM) classifier, random forest classifier, decision tree classifier, and voting classifier.

3.1. Gaussia Naive Bayes

The Gaussian Naive Bayes classifier is a probabilistic algorithm used for text classification tasks, including the classification of YouTube comments into "ham" or "spam" categories. It assumes that the features (TF-IDF values) within each class (ham and spam) follow a Gaussian (normal) distribution, where the probability density function is given by:

$$p(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp(-\frac{(x_i^2-\mu_y)^2}{2\sigma_y^2}) \tag{1}$$

To classify a new YouTube comment, the GNB model leverages Bayes' theorem and calculates the class conditional probability $(P(x|y))$ for each feature value. The model uses the class-specific feature statistics (mean and variance) learned during training to estimate these probabilities.

During training, the GNB model analyzes the training dataset to compute the mean and variance of each feature within both classes. These class-specific statistics capture the distribution of feature values within each class and form the basis for making probabilistic predictions during the classification process.

By combining the probabilities of individual feature values using Bayes' theorem, the GNB model calculates the posterior probabilities for each class. The class with the highest posterior probability is assigned to the comment, effectively determining its classification as either "ham" or "spam". The GNB classifier's strength lies in its simplicity, efficiency, and effectiveness in handling high-dimensional datasets like TF-IDF representations of text data. By making the assumption of feature independence, the model significantly reduces computational complexity, making it suitable for real-world applications with large datasets.

We implemented the Gaussian Naive Bayes classifier using Scikit-learn's GaussianNB class. By utilizing the class conditional probabilities based on the Gaussian distribution assumption, the model efficiently processes YouTube comments and assigns them to the appropriate class. The KNN Classifier is a non-parametric algorithm that classifies data points based on their proximity to neighboring points in the feature space. By calculating the distance between a query point (a YouTube comment's feature vector) and its k number of nearest neighbors, the model assigns the query point to the majority class among its neighbors. KNN leverages the inherent spatial characteristics of the data to make predictions, allowing it to capture local patterns and non-linear decision boundaries. However, this approach might be sensitive to the choice of k, and the performance can degrade in high-dimensional feature spaces.

3.2. Logistic regression

Logistic Regression (James et al., 2013) is a widely-used linear classification algorithm that models the relationship between the input features (in this case, TF-IDF values of YouTube comments) and the binary output classes (ham or spam) through the logistic function. Unlike linear regression, logistic regression is designed to predict probabilities and make binary decisions.

The algorithm estimates the probability of a comment belonging to a particular class (ham or spam) using the logistic function, also known as the sigmoid function:

$$f(x) = \frac{e^{a+bx}}{1+e^{a+bx}} \tag{2}$$

During training, the logistic regression model uses optimization techniques like the maximum likelihood estimation or gradient descent to find the optimal coefficients that minimize the error between the predicted probabilities and the actual class labels in the training data.

To classify a new YouTube comment, the model uses the estimated coefficients to compute the linear combination of features and then passes the result through the sigmoid function to obtain the probability of the comment belonging to the "spam" class. If the probability exceeds a threshold (typically 0.5), the comment is classified as "spam"; otherwise, it is classified as "ham".

The provided code implements logistic regression using Scikit-learn's LogisticRegression class. It uses cross-validation to find the optimal regularization parameter (lambda or C) that helps prevent overfitting. By fitting the model on the training data and evaluating its performance on the test data, the classifier can accurately classify YouTube comments into "ham" or "spam" categories.

Overall, logistic regression is a simple yet effective classification algorithm for YouTube comment classification tasks. Its ability to model non-linear relationships and handle high-dimensional data makes it a valuable choice for various text classification applications, including spam detection and sentiment analysis.

3.3. K-nearest neighbor classifier

The k-Nearest Neighbors (k-NN) algorithm is a simple and intuitive classification algorithm used for classification problems. It is a type of instance-based learning algorithm, meaning it does not explicitly build a model during the training phase. Instead, it memorizes the training dataset and makes predictions based on the similarity between new data points and the existing training data. The choice of the parameter k is crucial in the k-NN algorithm. If k is too small, the algorithm can be sensitive to noise and outliers in the data. If k is too large, the decision boundary becomes too smooth, and the algorithm may fail to capture local patterns. The optimal k value depends on the specific dataset and problem. The KNN algorithm has no training phase and the model adapts to new data in real-time. However, its computationally expensive, especially for large datasets. It is very sensitive to irrelevant or redundant features.

3.4. Support vector machine classifier

The SVM Classifier excels at separating data points into different classes by finding an ideal hyperplane that maximizes the margin between classes. SVM successfully distinguishes between ham and spam comments by transforming the input characteristics into a high-dimensional space and establishing a decision boundary that reduces classification errors. To provide resilience to outliers and enhance generalization, SVM makes use of support vectors, which are data points closest to the decision border. Additionally, SVM can handle non-linearly separable data and capture intricate correlations between features and labels by using kernel functions.

The objective function of the SVM is to find the optimal hyperplane that separates the data points with the maximum margin. The hyperplane can be represented as

$$w^T x + b = 0 \tag{3}$$

where w is the normal vector to the hyperplane, x is the input feature vector, and b is the bias term or intercept.

Given a dataset with m samples and n features, each sample $x^{(i)}$ is associated with a label $y^{(i)}$ where $y^{(i)} \in \{-1, 1\}$.

The optimization problem can be formulated as follows (James et al., 2013)

$$\min_{w,b} \frac{1}{2}\|w\|^2 \tag{4}$$

subject to the constraints

$$y^{(i)}(w^T x^{(i)} + b) \geq 1, \text{ for } i = 1, 2, \ldots, m \tag{5}$$

The Lagrangian for the SVM optimization problem is (James et al., 2013)

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m} \alpha^{(i)}[y^{(i)}(w^T x^{(i)} + b) - 1] \tag{6}$$

where $\alpha^{(i)}$ are the Lagrange multipliers, one for each data point.

The decision function for predicting the class of a new data point $x^{(new)}$ is given by James et al. (2013)

$$f(x^{(new)}) = \text{sign}\left(\sum_{i=1}^{m} \alpha^{(i)} y^{(i)} K(x^{(i)}, x^{(new)}) + b\right) \tag{7}$$

where $K(x^{(i)}, x^{(new)})$ is the kernel function, typically the radial basis function (RBF) kernel $K(x, y) = \exp(-\gamma\|x - y\|^2)$.

3.5. MultiLayer perceptron

The MLP Classifier is a neural network architecture with multiple hidden layers. Each layer consists of interconnected nodes (neurons) that apply weighted transformations to the input data (Windeatt, 2006). Through forward propagation, the network learns to approximate complex non-linear relationships between features and labels. The hidden layers allow the model to capture intricate patterns and hierarchical representations, making it a powerful tool for text classification tasks like identifying spam comments. However, training an MLP requires careful consideration of hyperparameters, such as the number of hidden layers and neurons, to prevent overfitting.

3.6. Random forest classifier

The Random Forest Classifier leverages the strength of decision trees and ensemble learning. By constructing multiple decision trees, each trained on different subsets of the data, and aggregating their predictions through voting and averaging, Random Forest mitigates the over-fitting of individual trees. The ensemble approach increases the model's accuracy, robustness, and resilience to noise. Random Forest also provides a measure of feature importance, allowing us to identify the most relevant features for classifying comments as ham or spam. Random forest has been widely used for classification, for example, it was applied to remote sensing classification (Pal, 2005).

3.7. Decision tree classifier

The Decision Tree Classifier adopts a hierarchical tree structure to make decisions based on feature values. It recursively splits the data at different nodes based on the most discriminative features, seeking to minimize impurity (e.g., Gini impurity or entropy). Decision trees are interpretable and easy to visualize, making them insightful tools for understanding the classification process. They handle both numerical and categorical features, enabling effective feature selection. However, decision trees can suffer from overfitting if they become excessively deep, leading to poor generalization.

We use Gini index to decide the split. The Gini index is defined as (James et al., 2013)

$$G = \sum_{i=1}^{K} \hat{p}_{mi}(1 - \hat{p}_{mi}) \tag{8}$$

where $\hat{p}_{mi}$ represents the probability that the training data in the mth region that are from the ith class. The classification error rate is represented as (James et al., 2013)

$$p_e = 1 - \max_i(\hat{p}_{mi}) \tag{9}$$

3.8. Voting classifier

The Voting Classifier is an ensemble model that combines the predictions of multiple classifiers in use, including KNN, SVM, MLP, Decision Tree, and Random Forest. It uses majority voting or weighted averaging to make the final decision. By leveraging the diverse perspectives of individual classifiers, the Voting Classifier capitalizes on their complementary strengths, often yielding improved overall performance compared to any single classifier. reducing the risk of overfitting and improving generalization, but it comes at the cost of increased computational resources.

Voting classifier is based on the majority logic. In our design, we use seven classifiers, namely Gaussian Naive Bayes, logistic regression, K-nearest neighbors (KNN) classifier, multi-layer perceptron (MLP), support vector machine (SVM) classifier, random forest classifier, decision tree classifier, to make a decision, a spam or ham. Assume all the seven classifiers are independent and each classifier has equal probability of decision error ϵ, then we can make analysis of the probability of decision error for voting classifier.

$$p_e = \sum_{i=4}^{7} \binom{7}{i} \epsilon^i (1-\epsilon)^{7-i} \tag{10}$$

$$= 35\epsilon^4(1-\epsilon)^3 + 21\epsilon^5(1-\epsilon)^2 + 7\epsilon^6(1-\epsilon) + \epsilon^7 \tag{11}$$

Given that $\epsilon \ll 1$ and in most cases $\epsilon \to 0$, so $p \to 0$. This demonstrates that theoretically the voting classifier can achieves decision error approaching to 0. For example, if $\epsilon = 0.1$, then $p_e = 0.0027$ and the accuracy is 99.73%.

4. Experimental results

4.1. Performance and comparison

The Youtube comments dataset underwent preprocessing using TF-IDF (Term Frequency-Inverse Document Frequency)vectorization, which encoded the comments into numerical feature vectors. The term frequency (TF) component measures how frequently a word appears in a comment, providing local importance information. Conversely, the inverse document frequency (IDF) component evaluates the rarity of a word across the entire dataset, providing global importance information. The TF-IDF score, a combination of TF and IDF, signifies the relative significance of a word in a comment relative to the entire dataset, with higher TF-IDF scores indicating more crucial words for classification.

By transforming the textual comments into numerical representations, we enable machine learning models to effectively process and classify the data. Hyperparameter tuning was conducted to optimize the classifiers' performance, focusing on parameters such as C and γ for the SVM Classifier, $n_{estimators}$ and $\max_{depth}$ for the Random Forest Classifier, and $hiddenlayersizes$ for the MLP Classifier. Additionally, feature engineering techniques were explored to enhance performance, including customized text preprocessing aimed at improving the quality of input features.

Similarly, hyperparameter tuning addressed the $\max_{depth}$ parameter, which governs the maximum depth of each decision tree in the Random Forest. A deeper tree can capture more intricate relationships within the data, potentially improving accuracy on the training set. However, excessive depth can lead to overfitting, limiting the model's ability to generalize to new, unseen data. Through hyperparameter tuning, we searched for the optimal $\max_{depth}$ that strikes a balance between capturing essential patterns and maintaining generalization capabilities.

To accomplish hyperparameter tuning for the Random Forest, we employed GridSearchCV to perform an exhaustive search over a predefined hyperparameter grid. This method systematically explored various combinations of $n_{estimators}$ and $\max_{depth}$ values, enabling a comprehensive assessment of the model's performance across a wide range of settings.

During each iteration of the hyperparameter tuning process, the Random Forest was trained on the training dataset, and performance was evaluated using cross-dp2validation on the validation set. By utilizing metrics such as precision, accuracy, and AUC-ROC, we objectively evaluated the model's performance under different hyperparameter settings.

For the Random Forest Classifier, hyperparameter tuning focused on two key parameters: $n_{estimators}$ and $\max_{depth}$. The $n_{estimators}$ parameter represents the number of decision trees to be generated within the ensemble. Each decision tree captures different aspects of the data, and the aggregation of their predictions leads to the final classification. By tuning $n_{estimators}$, we explored the trade-off between model complexity and performance. A larger value of $n_{estimators}$ may lead to a more diverse and robust ensemble,

Several assessment point metrics, including accuracy, precision, recall, area under the receiver operating characteristic curve (AUC), and area under the precision–recall curve (AUPRC), were used to evaluate the performance of the model. Accuracy, precision, and recall were used to evaluate model performance at a fixed threshold of 0.5 while AUC and AUPRC were used to evaluate model performance over all potential thresholds. Precision and recall were also utilized to more comprehensively assess the performance of the model since accuracy overemphasizes performance on the majority class in an unbalanced dataset. The percentage of accurate positive predictions is measured by precision.

In Figs. 1 to 5, we plot the ROC curves of spam detection for five YouTube video using eight machine learning approaches. The ROC curve shows the trade-off between sensitivity (true positive rate) and specificity (true negative rate) at various thresholds. As we move along the curve, we can choose a threshold that balances these two metrics based on the specific requirements of the problem. We can make the following observations based on these figures:

1. The AUC is a quantitative measure of the model's overall performance. All eight machine learning models have training and test curves hugging the upper left corner which indicate that the seven machine learning models (except KNN) perform very well.
2. Points in the upper left corner of the ROC space represent high sensitivity and high specificity. As a result, the eight machine learning models have curves closer to the upper left corner are desirable.
3. The steepness of the ROC curve is also indicative of the model's performance. A steeper curve suggests better discrimination at different threshold levels. The eight machine learning models have very steep ROC curves.
4. Comparing the AUC values and shapes can help identify the model with superior discrimination ability. We can conclude that the eight machine learning models are suitable for spam detection.

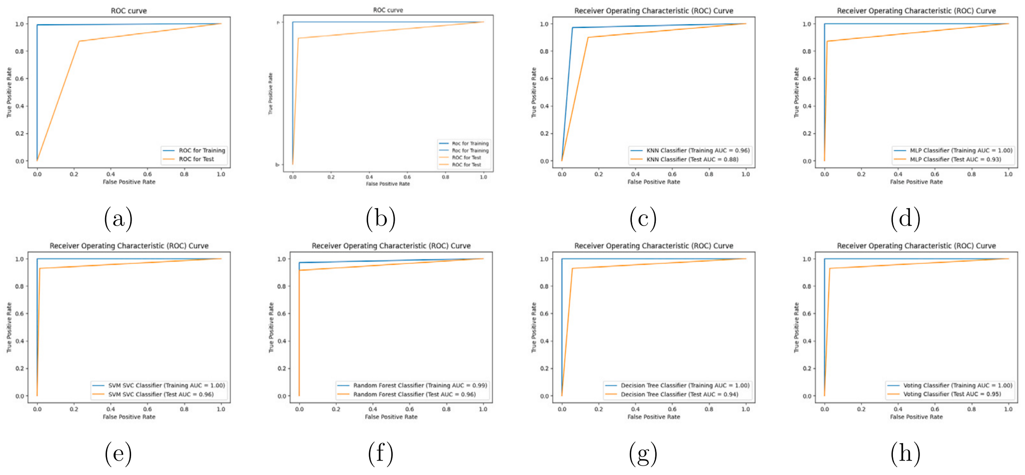

Fig. 1. ROC curves of spam detection for Psy YouTube video using eight machine learning approaches. (a) Naive Bayes, (b) Logistic Regression, (c) KNN, (d) MLP, (e) SVC, (f) Random Forest, (g) Decision Tree, (h) Voting Classifier.

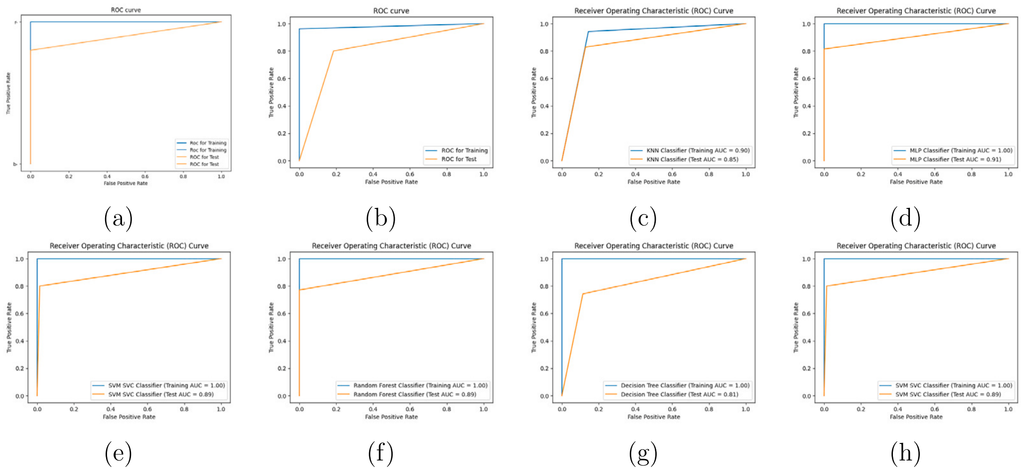

Fig. 2. ROC curves of spam detection for KatyPerry YouTube video using eight machine learning approaches. (a) Naive Bayes, (b) Logistic Regression, (c) KNN, (d) MLP, (e) SVC, (f) Random Forest, (g) Decision Tree, (h) Voting Classifier.

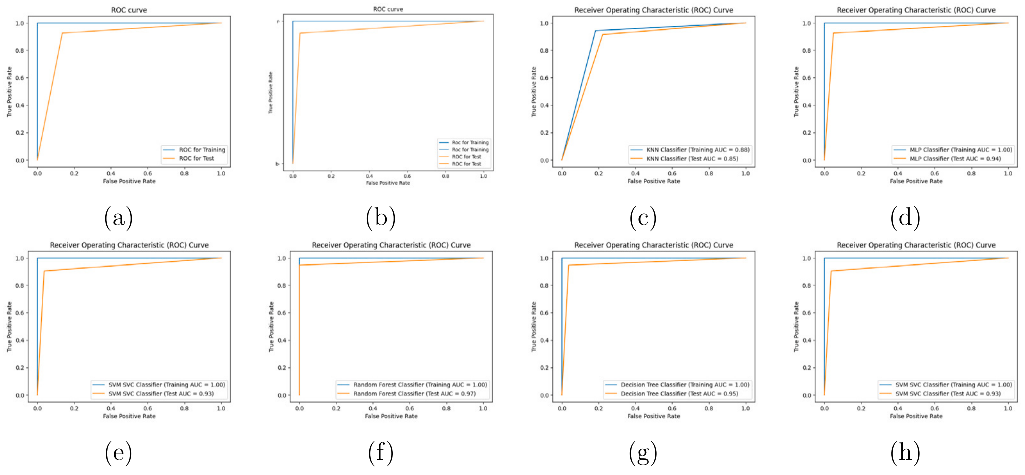

Fig. 3. ROC curves of spam detection for LMFAO YouTube video using eight machine learning approaches. (a) Naive Bayes, (b) Logistic Regression, (c) KNN, (d) MLP, (e) SVC, (f) Random Forest, (g) Decision Tree, (h) Voting Classifier.

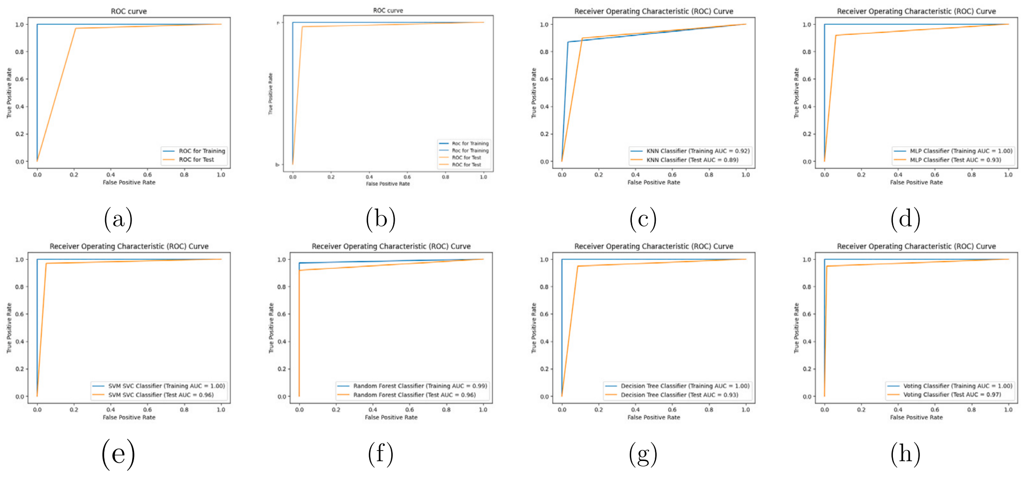

Fig. 4. ROC curves of spam detection for Eminem YouTube video using eight machine learning approaches. (a) Naive Bayes, (b) Logistic Regression, (c) KNN, (d) MLP, (e) SVC, (f) Random Forest, (g) Decision Tree, (h) Voting Classifier.

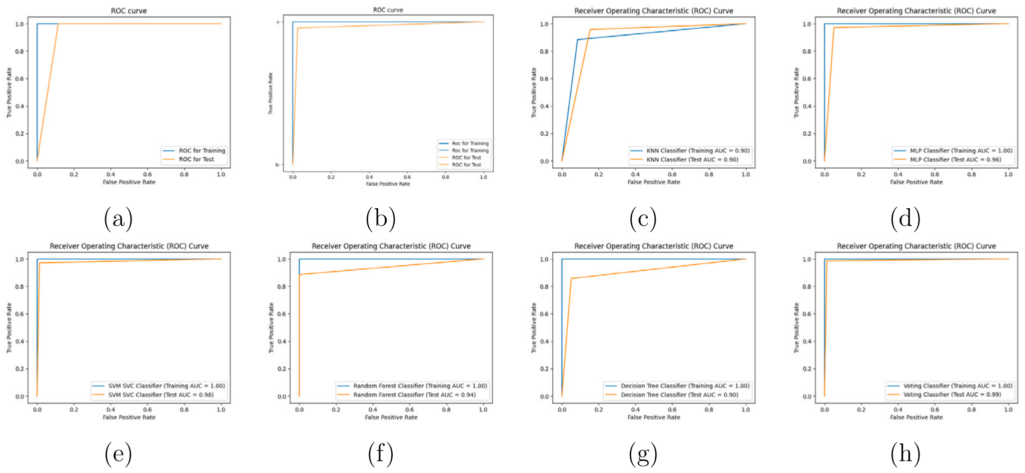

Fig. 5. ROC curves of spam detection for Shakira YouTube video using eight machine learning approaches. (a) Naive Bayes, (b) Logistic Regression, (c) KNN, (d) MLP, (e) SVC, (f) Random Forest, (g) Decision Tree, (h) Voting Classifier.

In Tables 1 to 5, we summarize the performance metrics of accuracy, precision, AUC, F1, and recall for five YouTube video using eight machine learning approaches. All these performance metrics are based on confusion matrix (James et al., 2013). The confusion matrix is a 2×2 matrix. The first row consists of True Positive (TP) and False Positive (FP), and the second row consists of False Negative (FN) and True Negative (TN). TP happens when both the actual and predicted are positive; FP is the case when the actual is negative but the predicted is positive; FN happens when the actual is positive but the predicted is negative; TN happens when both the actual and predicted are negative. The performance metrics are defined as (James et al., 2013)

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

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

$$\text{F1} = \frac{\text{2TP}}{\text{2TP+FP+FN}} \tag{14}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{15}$$

AUC is used to measure Area Under the ROC Curve (AUC).

We can make the following observations based on the five Tables:

1. High accuracy means that the model is making correct predictions overall. Random forest, decision tree, and voting classifier achieved the best accuracy.

Table 1
Spam detection for Psy YouTube video comments using eight machine learning approaches.

ML Approach	Accuracy	Precision	AUC	F1	Recall
Naive Bayes	0.839080	0.786408	0.839080	0.852632	0.931034
Logistic Regression	0.93678	0.936782	0.93491	0.934911	0.908046
KNN	0.921428	0.953846	0.921428	0.918518	0.885714
MLP	0.928571	0.96875	0.92857	0.92537	0.885714
SVC	0.957143	0.98485	0.957143	0.95586	0.928571
Random Forest	0.964285	1.0	0.96429	0.962962	0.928571
Decision Tree	0.92857	0.928571	0.928571	0.928571	0.9285714
Voting Classifier	0.957143	0.98485	0.957143	0.955882	0.928571

2. High precision indicates that among the instances predicted as positive, a large proportion are actually positive. Precision is the ratio of true positives to the total predicted positives. For all the five set of YouTube video comments, logistic regression, SVC, random forest, and voting classifier achieved the best precision
3. A high AUC value signifies good overall discriminatory ability of the model. Considering all five sets of comments, logistic regression, SVC, random forest, and voting classifier achieved the best AUC performance.
4. The F1 score combines precision and recall into a single metric. It is the harmonic mean of precision and recall. A high F1 score

Table 2
Spam detection for KatyPerry YouTube comments using eight machine learning approaches.

ML Approach	Accuracy	Precision	AUC	F1	Recall
Naive Bayes	0.8571432	0.820513	0.857143	0.864865	0.914286
Logistic Regression	0.914286	0.96774	0.914286	0.90909	0.857143
KNN	0.854054	0.857142	0.852568	0.842105	0.82758
MLP	0.929729	0.902173	0.931093	0.927374	0.954023
SVC	0.93513	0.94117	0.93426	0.930232	0.91954
Random Forest	0.945945	0.975308	0.943818	0.940476	0.908045
Decision Tree	0.928571	0.92857	0.9285714	0.928571	0.928571
Voting Classifier	0.945945	0.96385	0.944463	0.9411764	0.9195402

Table 3
Spam detection for LMFAO YouTube comments using eight machine learning approaches.

ML Approach	Accuracy	Precision	AUC	F1	Recall
Naive Bayes	0.908046	0.975610	0.909091	0.913032	0.851064
Logistic Regression	0.919540	0.954545	0.921809	0.923077	0.893617
KNN	0.8378	0.96	0.83003	0.79999	0.685714
MLP	0.9257142	0.94505	0.9265826	0.929729	0.914893
SVC	0.931428	0.9659	0.93360	0.93406	0.904255
Random Forest	0.942857	0.988372	0.94595	0.94444	0.904255
Decision Tree	0.954285	0.96739	0.954885	0.956989	0.94680
Voting Classifier	0.9485714	0.97752	0.950420	0.950819	0.925531

Table 4
Spam detection for Eminem YouTube comments using eight machine learning approaches.

ML Approach	Accuracy	Precision	AUC	F1	Recall
Naive Bayes	0.883929	0.848921	0.875714	0.900763	0.959350
Logistic Regression	0.924107	0.934426	0.930612	0.923811	0.926829
KNN	0.892857	0.93805	0.89624	0.898305	0.861788
MLP	0.89285714	0.90243	0.89181	0.90243	0.902439
SVC	0.928571	0.9495798	0.929646	0.933884	0.918699
Random Forest	0.9464285	0.982608	0.949448	0.94957	0.918699
Decision Tree	0.915178	0.90625	0.912138	0.924302	0.943089
Voting Classifier	0.946428	0.974358	0.948563	0.950000	0.92682

Table 5
Spam detection for Shakira YouTube comments using eight machine learning approaches.

ML Approach	Accuracy	Precision	AUC	F1	Recall
Naive Bayes	0.905405	0.833333	0.909091	0.910256	1.000000
Logistic Regression	0.959459	0.94444	0.960073	0.957746	0.971429
KNN	0.83783	0.83783	0.83003	0.799999	0.685714
MLP	0.959459	0.94444	0.960073	0.95774	0.971428
SVC	0.972972	0.9714285	0.9728937	0.971428	0.971428
Random Forest	0.9864864	1.0	0.985714	0.985507	0.97142
Decision Tree	0.972972	0.945945	0.974358	0.97222	1.0
Voting Classifier	0.972972	0.971428	0.972893+	0.97142	0.9714285

indicates a balance between precision and recall. It is especially useful in situations where there is an uneven class distribution. Based on the Tables, SVC, random forest, and voting classifier achieved the best F1 score.

5. Recall measures Sensitivity or True Positive Rate. High recall means that the model is able to correctly identify a large proportion of the actual positive instances. Random forest, decision tree, and voting classifier achieved the best performance in recall.

4.2. Computational complexity analysis

The complexity of the eight machine learning models are very different, which can be summarized as follows.

4.2.1. Naive Bayes

Training Complexity: During training, Naive Bayes computes the probability of each feature given each class. This involves scanning the dataset once to calculate the frequency of each feature for each class. Since this operation involves going through the dataset once and updating counts for each feature, the complexity is linear with respect to the number of instances in the dataset, which is $O(n)$.

Prediction Complexity: For prediction, Naive Bayes computes the posterior probability of each class given the features of a new instance. This computation also involves linear operations, as it requires iterating over the features once and updating the probabilities accordingly. Therefore, both the training and prediction phases of Naive Bayes have linear time complexity, $O(n)$, where n is the number of features in the dataset.

4.2.2. Logistic regression

Training Complexity: The training complexity of logistic regression depends on the optimization algorithm. Common optimization algorithms for logistic regression include gradient descent, stochastic gradient descent (SGD), mini-batch gradient descent, and advanced optimization techniques like L-BFGS. We use gradient descent. The complexity of gradient descent for logistic regression is $O(k \cdot n \cdot d)$, where k is the number of iterations, n is the number of features, and d is the number of instances in the dataset.

Prediction Complexity: Once logistic regression is trained, predicting the class of a new instance involves computing a weighted sum of the features and applying the logistic function. The prediction complexity is $O(n)$, where n is the number of features, because it involves multiplying the feature values by the learned coefficients and summing them up.

4.2.3. KNN

Training Complexity: The training complexity of K-nearest neighbor (KNN) is essentially $O(1)$. KNN does not explicitly build a model during the training phase; it just memorizes the training dataset. Hence, its training complexity is constant and not dependent on the size of the dataset.

Prediction Complexity: The prediction complexity of KNN depends on the number of instances in the training dataset (n), the number of features (d), and the value of k, the number of nearest neighbors to consider. The overall prediction complexity can be $O(n \cdot d)$ for computing distances, $O(n \cdot log(n))$ or $O(n)$ for finding the nearest neighbors depending on the method used, and $O(k)$ for the majority voting among the neighbors. So the total complexity of the prediction phase in KNN can be expressed as $O(n \cdot d + n \cdot log(n) + k)$.

4.2.4. MLP

Forward Pass: During the forward pass, the input is propagated through the network, and activations are computed at each layer. The complexity of the forward pass is $O(n)$, where n is the number of parameters in the network. It involves matrix multiplications and element-wise operations at each layer, which can be computed efficiently using optimized linear algebra libraries.

Backward Pass (Backpropagation): Backpropagation computes the gradients of the loss function with respect to the parameters of the network. The complexity of backpropagation depends on the number of parameters in the network and the number of training samples. For each training sample, backpropagation involves computing the gradients layer by layer, which is $O(n)$ for gradient descent.

Training Complexity: The overall complexity of training an MLP depends on the number of iterations (epochs) and the complexity of the forward and backward passes. The training complexity is typically $O(n \cdot e \cdot f)$, where n is the number of training samples, e is the number of epochs, and f is the complexity of the forward and backward passes.

Prediction Complexity: Once the MLP is trained, making predictions for new samples involves a forward pass through the network, which has a complexity similar to the forward pass during training, typically $O(n)$ where n is the size of input data.

4.2.5. SVC

Training Complexity: In our design, we use Radial Basis Function (RBF) kernel, the training complexity is $O(n^2 \cdot d)$ where n is the number of training samples and d is the number of features.

Prediction Complexity: The prediction complexity for SVC using the RBF kernel is $O(s \cdot d)$, where s is the number of support vectors and d is the number of features. During prediction, the SVC computes the decision function using a subset of training samples called support vectors. The number of support vectors, s, is usually much smaller than the total number of training samples, leading to efficient prediction.

4.2.6. Random forest

Training Complexity: For each decision tree in the Random Forest, constructing a decision tree generally has a complexity of $O(m \cdot n \cdot \log(n))$, where: m is the number of features and n is the number of training samples. However, Random Forest typically uses a subset of features at each split, which can reduce the effective number of features considered at each node.

Prediction Complexity: The prediction complexity of a Random Forest is $O(k \cdot \log(m))$, where: k is the number of trees in the forest and m is the number of features. Predicting with a Random Forest involves traversing each tree in the forest and aggregating the predictions. The logarithmic factor comes from the depth of the decision trees.

4.2.7. Decision tree

Training Complexity: The overall training complexity of a decision tree is $O(m \cdot n \cdot \log(n))$ in the average case, where m is the number of features and n is the number of training samples.

Prediction Complexity: Predicting with a decision tree involves traversing the tree from the root to a leaf node based on the feature values of the instance to be classified. The prediction complexity is $O(log(n))$, where n is the number of nodes in the decision tree. The logarithmic time complexity arises because the decision tree has a depth proportional to the logarithm of the number of nodes.

4.2.8. Voting Classifier

Training Complexity: The training complexity of a Voting Classifier depends on the complexity of training each individual estimator or base model. In our design, the Voting Classifier uses a hard voting strategy, where the final prediction is based on a simple majority vote, the training complexity is primarily determined by the training complexity of each base model.

Prediction Complexity: The prediction complexity of a Voting Classifier is determined by the prediction complexity of each individual base estimator. During prediction, the Voting Classifier collects predictions from each base estimator and combines them according to the specified voting strategy. The prediction complexity is typically determined by the complexity of predicting with the most complex base estimator used within the Voting Classifier.

5. Conclusions and future work

We have applied eight machine approaches, namely Gaussian Naive Bayes, logistic regression, K-nearest neighbors classifier, multi-layer perceptron, support vector machine classifier, random forest classifier, decision tree classifier, and voting classifier, to spam detection for YouTube video comments. The machine learning model design and parameters selection are presented. Simulation results show that the eight approaches could perform successfully in spam detection. Random forest could achieve almost perfect detection performance.

Further investigation may explore alternative feature engineering approach, evaluate ensemble learning techniques, or address the challenges associated with class imbalance in the dataset. Such advancements would foster the continual enhancement of comment classification systems, ensuring improved user engagement and content management across digital platforms.

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

Data and Python codes links are shared in the Appendix in paper.

Acknowledgments

This work was supported by U.S. National Science Foundation (NSF) under Grant CCF-2219753.

Appendix

We used the YouTube Video Comments online dataset in Kaggle: https://www.kaggle.com/datasets/lakshmi25npathi/images/data

Our Python codes is available online:
https://colab.research.google.com/drive/16eG4fDIPBPlq8aLK-8D9smh-kTVGUsbS?usp=sharing4

References

Ahmed, N., Amin, R., Aldabbas, H., Koundal, D., Alouffi, B., & Shah, T. (2022). Machine learning techniques for spam detection in email and IoT platforms: Analysis and research challenges. *Security and Communication Networks*, *2022*, 1–19.

Bacanin, N., Zivkovic, M., Stoean, C., Antonijevic, M., Janicijevic, S., Sarac, M., et al. (2022). Application of natural language processing and machine learning boosted with swarm intelligence for spam email filtering. *Mathematics*, *10*(22), 4173.

Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Al Najada, H. (2015). Survey of review spam detection using machine learning techniques. *Journal of Big Data*, *2*(1), 1–24.

Danilchenko, K., Segal, M., & Vilenchik, D. (2022). Opinion spam detection: A new approach using machine learning and network-based algorithms. In *Proceedings of the international AAAI conference on web and social media, vol. 16* (pp. 125–134).

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., et al. (2023). Scaling vision transformers to 22 billion parameters. In *International conference on machine learning* (pp. 7480–7512). PMLR.

Fui-Hoon Nah, F., Zheng, R., Cai, J., Siau, K., & Chen, L. (2023). Generative AI and ChatGPT: Applications, challenges, and AI-human collaboration. *Journal of Information Technology Case and Application Research*, *25*(3), 277–304.

Grewal, N., Nijhawan, R., & Mittal, A. (2022). Email spam detection using machine learning and feature optimization method. In *Distributed computing and optimization techniques: select proceedings of ICDCOT 2021* (pp. 435–447). Springer.

Guo, Y., Mustafaoglu, Z., & Koundal, D. (2023). Spam detection using bidirectional transformers and machine learning classifier algorithms. *Journal of Computational and Cognitive Engineering*, *2*(1), 5–9.

James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*: *vol. 112*, Springer.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in vision: A survey. *ACM Computing Surveys (CSUR)*, *54*(10s), 1–41.

Lin, T., Wang, Y., Liu, X., & Qiu, X. (2022). A survey of transformers. *AI Open*.

Makkar, A., Garg, S., Kumar, N., Hossain, M. S., Ghoneim, A., & Alrashoud, M. (2020). An efficient spam detection technique for IoT devices using machine learning. *IEEE Transactions on Industrial Informatics*, *17*(2), 903–912.

Manasa, P., Malik, A., Alqahtani, K. N., Alomar, M. A., Basingab, M. S., Soni, M., et al. (2022). Tweet spam detection using machine learning and swarm optimization techniques. *IEEE Transactions on Computational Social Systems*.

Mashaleh, A. S., Ibrahim, N. F. B., Al-Betar, M. A., Mustafa, H. M., & Yaseen, Q. M. (2022). Detecting spam email with machine learning optimized with harris hawks optimizer (hho) algorithm. *Procedia Computer Science*, *201*, 659–664.

Niranjani, V., Agalya, Y., Charunandhini, K., Gayathri, K., & Gayathri, R. (2022). Spam detection for social media networks using machine learning. In *2022 8th international conference on advanced computing and communication systems, vol. 1* (pp. 2082–2088). IEEE.

Pal, M. (2005). Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, *26*(1), 217–222.

Shaaban, M. A., Hassan, Y. F., & Guirguis, S. K. (2022). Deep convolutional forest: a dynamic deep ensemble approach for spam detection in text. *Complex & Intelligent Systems*, *8*(6), 4897–4909.

Sun, N., Lin, G., Qiu, J., & Rimba, P. (2022). Near real-time twitter spam detection with machine learning techniques. *International Journal of Computers and Applications*, *44*(4), 338–348.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *30*.

Windeatt, T. (2006). Accuracy/diversity and ensemble MLP classifier design. *IEEE Transactions on Neural Networks*, *17*(5), 1194–1211.

Wu, T., He, S., Liu, J., Sun, S., Liu, K., Han, Q.-L., et al. (2023). A brief overview of ChatGPT: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, *10*(5), 1122–1136.