Journal of Information Systems and Informatics
Vol. 7, No. 3, September 2025 e-ISSN: 2656-4882 p-ISSN: 2656-5935
DOI: 10.51519/journalisi.v7i3.1215 Published By DRPM-UBD

Detecting Data Leakage in Cloud Storage Using Decision Tree Classification

Parlindungan Harahap[1], Muhammad Siddik Hasibuan[2]

[1,2]Universitas Islam Negeri Sumatera Utara, Medan, Indonesia
Email: [1]parlinhrp289@gmail.com, [2]muhammadsiddik@uinsu.ac.id

Abstract

Data leakage in cloud storage systems poses a significant security threat, potentially leading to unauthorized access, loss of sensitive information, and operational disruptions. This research proposes a classification model for detecting potential data leakage incidents using the Decision Tree algorithm. The dataset, obtained from the Kaggle public repository, contains user activity logs representing both normal and anomalous behaviors in cloud storage environments. Several preprocessing steps were applied to improve model quality, including handling missing values, removing outliers, and converting categorical data into numerical form. Hyperparameter optimization was performed using GridSearchCV to determine the best configuration for the Decision Tree classifier. Experimental results demonstrate that the optimized model achieved high classification performance, with an accuracy of 70,84%, a precision of 55% for the data leakage class, and an F1-score of 40%. The analysis also highlights the significance of certain features, such as multi-factor authentication usage and access to confidential data, in predicting potential leakage events. This study provides a theoretical contribution by \establishing a robust methodology for applying Decision Tree algorithms to a novel cloud security dataset, offering a scalable and interpretable framework for automated threat detection.

Keywords: Cloud Storage, Data Leakage Detection, Decision Tree, GridSearchCV, Machine Learning

1. INTRODUCTION

The rapid advancement of digital technology has transformed the way individuals and organizations store and manage data. This has established cloud storage as a primary solution due to its convenience in accessing data from anywhere, flexibility, and high scalability. However, this massive adoption of cloud storage also introduces significant security risks, particularly the threat of data leakage, which is a critical issue that can jeopardize the integrity and confidentiality of information [1], [2]. These incidents can result from various factors, including misconfigurations, system vulnerabilities, and complex cyberattacks. This risk is exacerbated by the increasing volume of data stored in the cloud, making it a prime target for malicious actors [3]. The impacts of data leakage are extensive, encompassing the loss of sensitive information, financial damage, a decline in

This work is licensed under a Creative Commons Attribution 4.0 International License.

corporate reputation, and legal consequences due to privacy regulation violations. Therefore, the need for a reliable security system to detect and prevent data leakage is becoming increasingly urgent.

Traditional security approaches, such as firewalls and rule-based detection systems, are often considered inadequate in addressing the evolving landscape of cyber threats [4]. These methods are reactive, capable only of recognizing previously known attack patterns (signature-based detection), and are ineffective at detecting new, unregistered threats. This limitation has prompted researchers and security practitioners to seek more proactive and intelligent solutions. One promising approach is the application of machine learning and artificial intelligence in security systems. Previous studies have shown that machine learning can be implemented for various security tasks, including sentiment analysis and the detection of anomalies in network systems [5], [6], [7]. With its ability to identify hidden patterns in data, machine learning can be a powerful tool for detecting suspicious behavior that indicates potential data leakage before an incident occurs.

A wide range of research has been conducted to leverage machine learning in the field of cybersecurity and its related contexts. Some studies focus on the detection of specific attacks, such as research that successfully developed a web-based phishing detection system using the Decision Tree and Random Forest algorithms, which yielded high classification performance [8], [4]. In a networking context, the Intrusion Detection System (IDS) using Decision Tree has also been analyzed to detect anomalies against cyberattacks [5]. Additionally, efficient and privacy-preserving Decision Tree methods in a cloud environment have been discussed, which is highly relevant to this topic [9]. The focus on cloud storage security has also been a major topic, with research exploring the use of Kubernetes for collaborative research platforms and studies examining the challenges and opportunities in using cloud storage for digital documents [10], [2]. The threat of data leakage does not only originate externally; this is also supported by research analyzing efforts to prevent consumer data leakage and emphasizing the importance of regulations like the Personal Data Protection Bill [1], while the general impact of cloud computing on system and data security has also been studied [3].

More specifically, the Decision Tree algorithm has proven effective in various classification applications. Its implementation has been successful in the development of web-based applications for data classification, the classification of active or inactive customers in bank data, and in health service decision-making related to COVID-19 [11], [12], [13]. The strength of the Decision Tree lies in its ability to produce an easily understandable and interpretable model, which allows for the identification of key factors contributing to data leakage. Furthermore, there have been efforts to further develop this method, such as the work by

Setyawan, who examined the development of Decision Tree with data discretization and attribute splitting using hierarchical clustering [14]. These studies show that the Decision Tree algorithm is continuously being improved to enhance its performance. The Decision Tree algorithm has also been examined in the context of informatics project management, highlighting its relevance across diverse application domains [15].

Despite numerous studies using machine learning and Decision Tree for various classification and detection problems, a significant research gap remains concerning the specific detection of data leakage in cloud storage environments by leveraging user activity patterns. Most research focuses on general network attack detection or phishing detection, but lacks a deep analysis of activity log data in cloud storage to identify anomalies leading to data leakage. Few studies have specifically examined how features such as access to confidential data, multi-factor authentication usage, and unusual access patterns can serve as important indicators in a data leakage detection model.

This research aims to fill this gap by developing and evaluating a data leakage detection model for cloud storage specifically. By utilizing the Decision Tree algorithm and analyzing relevant features from user activity log data, this study is expected to provide a more effective and interpretable solution compared to traditional methods. Based on the background above, the research problems in this study are as follows: (1) How can the Decision Tree algorithm be implemented to detect anomalous activities that could lead to data leakage in cloud storage? (2) What are the most significant factors that contribute to increasing the effectiveness of a data leakage detection model using the Decision Tree algorithm? (3) What is the performance of the developed Decision Tree model in detecting cloud storage data leakage based on evaluation metrics such as accuracy, precision, recall, and F-1 score?

The objective of this research is to design, develop, and evaluate a data leakage detection model for cloud storage using the Decision Tree algorithm. This study also aims to identify the key features that are most influential in detecting anomalies and to comprehensively analyze the model's performance [16]. It is hoped that this research will provide a theoretical contribution by enriching the body of knowledge in cybersecurity and machine learning, particularly concerning the application of the Decision Tree algorithm for detecting data leakage in cloud storage. Practically, this research is expected to produce a data leakage detection model that can serve as a reference for companies or organizations in strengthening their cloud storage security systems. The analysis of key features can also assist in the development of smarter and more proactive security monitoring systems [17], [18].

2. METHODS

This study adopts a quantitative approach with an applied research method. The quantitative approach is chosen to systematically analyze user activity log data, where numerical and categorical variables are measured to test hypotheses. The applied method is used because the main objective of this research is to design, implement, and evaluate a practical model that can be used to detect potential data leakage in cloud storage services [19]. The results of this study are expected to provide concrete solutions that can be directly implemented in cybersecurity systems.

2.1. Research Framework

This research is carried out through several systematically structured stages to ensure accurate and accountable results. In general, these stages follow the machine learning model development cycle and can be illustrated in a flowchart as shown in Figure 1. This diagram will clearly show the sequence of processes from data collection, preprocessing, model development, to evaluation. Iterations may occur in the optimization phase to achieve the best model performance.

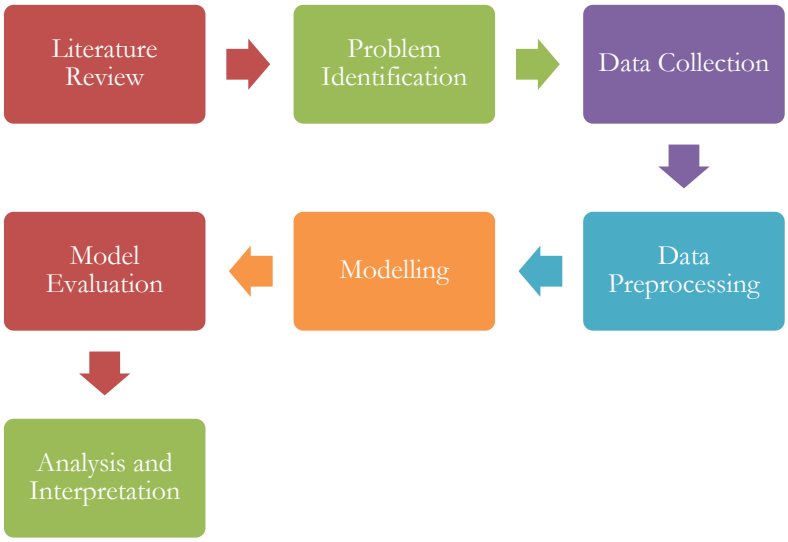

Figure 1. Research framework

The research process begins with a Literature Review to gather information from existing scientific articles, journals, and books. This stage is crucial for building a strong theoretical foundation, understanding the current state of research in cloud security and data leakage detection, and identifying research gaps. Based on the literature review, the specific Problem Identification is conducted. This involves

defining the exact research question and the scope of the study, which in this case is the development of a classification model to detect data leakage using a specific algorithm. Following this, the Data Collection phase is initiated, where a relevant dataset is obtained from an open-source platform, specifically a public repository on Kaggle. This dataset consists of user activity logs, which are the primary source of information for this research.

The collected data then undergoes a Data Preprocessing stage to clean and prepare it for analysis. This involves a series of steps such as handling missing values, removing outliers, normalizing numerical features, and encoding categorical variables to ensure data quality and suitability for the chosen algorithm. Next is the Data Modeling stage, where the Decision Tree algorithm is implemented. The preprocessed dataset is split into training and testing sets. The training data is used to build the model, and an optimization technique like GridSearchCV is applied to fine-tune the model's hyperparameters to achieve the best possible performance. The developed model is then subjected to Model Evaluation using the testing dataset. The performance is measured using various metrics such as Accuracy, Precision, Recall, and F-1 Score to assess the model's effectiveness in detecting data leakage. Finally, the Analysis and Interpretation of Results stage involves a detailed discussion of the evaluation outcomes. This stage provides an interpretation of the model's performance, highlights the most significant features for predicting data leakage, and outlines the implications of the findings in the context of improving cloud storage security.

2.2. Data Collection

The data used in this study is a secondary dataset obtained from the Kaggle public repository. This dataset consists of 49,498 instances and 10 attributes, which contain a collection of user activity logs represented in a tabular data format. Each row of data represents a single user event or activity, while the columns contain various features that can be used to identify normal and anomalous behavior. These features include information such as user ID, activity type (e.g., 'download', 'upload', 'share', 'access'), access time, geographical location, and authentication status. The selection of this dataset is based on its relevance to the data leakage detection case, which has a class label indicating whether an activity is normal or anomalous.

The dataset was divided into training and testing subsets using an 80:20 ratio, where 80% of the data was used for model training and 20% was reserved for testing. A stratified sampling technique was applied to preserve the original class distribution of normal and anomalous activities, ensuring that both subsets represented the actual proportions of the dataset. This approach minimizes sampling bias and allows for a more reliable evaluation of the model's

performance. The data used in this study is a secondary dataset that records user behavior and activity patterns when interacting with a cloud storage system. The dataset, obtained from a public repository, consists of the following features:

1) Authority: Indicates the level of authority or user's privilege in accessing the system.
   1. Staff
   2. Manager
   3. Senior Manager
   4. Expert Staff
   5. Intern
2) Through_pwd: Indicates authentication via password.
   1. User accesses the system using a password.
   0. User accesses the system without using a password.
3) Through_pin: Indicates authentication using a PIN.
   1. User accesses the system via a PIN.
   0. User accesses the system without a PIN.
4) Through_MFA: Indicates the use of Multi-Factor Authentication.
   1. User accesses the system through MFA authentication.
   0. User accesses the system without MFA authentication.
5) Data_Modification: Indicates data modification activity.
   1. Data modification occurred during the recorded activity.
   0. No data modification occurred during the recorded activity.
6) Confidential_Data_Access: Indicates access to confidential data.
   1. User accesses confidential data.
   0. User does not access confidential data.
7) Confidential_File_Transfer: Indicates confidential file transfer activity.
   1. A confidential file transfer occurred during the recorded activity.
   0. No confidential file transfer occurred during the recorded activity.
8) Operation: Indicates the type of operation performed on the data.
   1. Deleting
   2. Reading
   3. Moving
   4. Writing
9) Data_Sensitivity: Indicates the level of data sensitivity being accessed.
   1. Low
   2. Medium
   3. High
10) Label: The final classification indicating data leakage.
   0. Normal
   1. Not Normal (Anomaly)

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Table 1. Dataset

No	Authority	Through_pwd	Through_pin	Through_MFA	Data_Modification	Confidential_Data_access	Confidential_File_Transfer	Operation	Data_Sensitivity	Label
1	2	0	0	1	0	0	1	3	1	0
2	1	1	0	0	1	0	0	4	1	1
3	2	0	0	1	1	0	1	4	1	0
4	1	0	1	0	0	1	1	3	3	0
5	3	0	0	1	0	1	0	2	3	0
6	1	0	0	1	0	1	1	3	2	1
7	1	0	1	0	0	0	0	2	2	1
8	1	1	1	1	0		0	3	3	1
9	3	0	1	1	0	0	0	1	2	0
10	3	0	0	1	0	1	1	3	2	0
11	1	0	0	1	0	1	0	2	2	0
12	1	0	1	0	1	0	1	3	3	1
13	1	0	0	1	0	1	1	3	2	1
14	2	0	1	0	1	0	0	1	2	0
15	2	0	0	1	1		1	3	2	0
49.498	2	0	0	1	0	1	0	2	3	0

This dataset has characteristics suitable for analysis using the Decision Tree algorithm because it includes various security parameters that can be potential indicators in detecting data leakage in a cloud storage environment. Each attribute provides specific information about user activity patterns that can be used to identify and classify potential data leakage. The activity logs recorded in this dataset reflect various cloud storage usage scenarios, from normal access to those indicating an anomaly.thematical equations are numbered in Arabic numerals, open-close brackets, and in align right column position.

Prior to modeling, the dataset underwent preprocessing to ensure reliability. Missing values were imputed using the mode for categorical attributes and the median for numerical attributes, while outliers were detected with the interquartile range (IQR) method and removed when exceeding 1.5×IQR. Categorical features, including authority level, authentication type, and operation type, were encoded into numerical format through one-hot encoding, and numerical attributes were normalized with MinMaxScaler to enhance feature comparability and improve the stability of the Decision Tree classifier.

2.3. Classification Model Development

The Decision Tree model in this study uses the CART (Classification and Regression Tree) algorithm. This algorithm is an advancement of the Decision Tree model that can be used for both classification and regression[20]. In this research, the model is used for classification to detect data leakage in a cloud storage system. CART uses the Gini Index or Entropy as the node splitting criterion, with a binary tree structure where each node has only two branches. This model is implemented using Scikit-Learn (sklearn.tree.DecisionTreeClassifier) in Python. The choice of the Decision Tree algorithm is based on its interpretability, which allows us to identify the key features contributing to data leakage. Unlike black-box models such as neural networks, Decision Tree provides a clear, rule-based output that is easy to understand and explain to security professionals. This interpretability is crucial for developing proactive security monitoring systems.

The model is trained using a dataset that contains cloud storage user activity logs with various features to classify whether an activity constitutes a data leak or not. By applying the CART algorithm, it is expected to handle a combination of numerical and categorical data and support pruning techniques to reduce overfitting[21],[22]. For hyperparameter optimization, GridSearchCV is employed to systematically test a predefined set of parameters to find the best combination for the model. The parameters tuned include max_depth (to control the tree's complexity and prevent overfitting), min_samples_split (to ensure each split is based on a meaningful number of samples), and criterion (Gini or Entropy). This exhaustive search ensures that the model is fine-tuned for optimal performance on the given dataset[23].

2.4. Model Evaluation, Analysis, and Conclusion

This section provides a comprehensive overview of the model's performance and the implications of the research findings. First, the Model Evaluation is discussed based on key metrics of accuracy, precision, recall, and F1-score. The performance metrics are crucial for quantifying the effectiveness of the model in a real-world scenario. This part also includes a comparison between the developed model and other detection methods if possible. Next, the Results and Analysis section provides an in-depth interpretation of the model's performance. It highlights the most significant features for predicting data leakage and discusses how the findings contribute to improving cloud storage security systems. The analysis also explores the practical implications of implementing a Decision Tree-based detection system. Finally, the Conclusion and Suggestions summarize the main findings of this research and provide clear recommendations for further development in similar future studies. This includes identifying potential areas for improvement, such as testing with a larger dataset or exploring other machine learning algorithms.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

3. RESULTS AND DISCUSSION

This section presents the research results, starting from the data preprocessing stages, classification model implementation, to evaluation and optimization. The discussion will thoroughly analyze the effectiveness of the developed model in detecting data leakage in a cloud storage environment, while also interpreting the key findings from each stage.

3.1. Data Preprocessing

The data preprocessing stage is a fundamental step taken to prepare the dataset for modeling. This stage includes initial data inspection, handling missing data, and managing outliers to ensure optimal data quality. To understand the relationships between features and identify relevant patterns, data visualization is performed using a pairplot, which is presented in Figure 2.

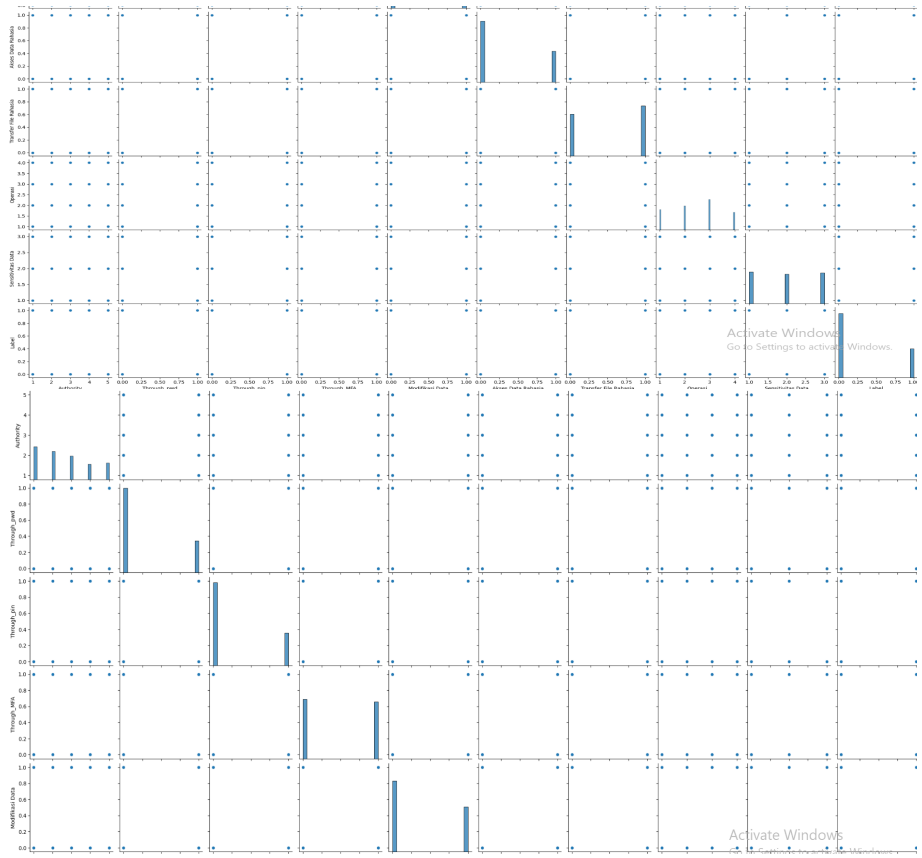

Figure 2. Pairplot fisualization of feature relationships

Furthermore, a more in-depth data analysis is carried out by creating a pivot table to summarize the frequency of activities based on key features, as shown in Table 2 to 4. These two steps are very helpful in gaining initial insights before proceeding to the modeling stage.

Table 2. Pivot table authority

Authority	Label
1	5.171
2	3.796
3	2.841
4	1.804
5	1.630

Table 3. Pivot table operation

Operation	Sensitivity Data
1	1,89
2	1,86
3	2,21
4	1,87

Table 4. Pivot table through_pwd

Authority	Label
1	5.171
2	3.796
3	2.841
4	1.804
5	1.630

3.2. Modeling

Classification modeling is performed using the Decision Tree algorithm with a CART (Classification and Regression Tree) implementation. The selection of this model is based on its excellent ability to handle mixed data, both numerical and categorical, and its ease of interpretation. The model training process is conducted using the preprocessed training data. The visualization of the trained Decision Tree model is presented in Figure 3, which shows the decision rules formed by the model. Subsequently, the model is evaluated to obtain a performance baseline before optimization. The results of this initial evaluation are presented in Figure 4. providing an initial overview of the model's performance in classifying activities as 'normal' or 'anomaly'.

Journal of Information Systems and Informatics
Vol. 7, No. 3, September 2025
p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

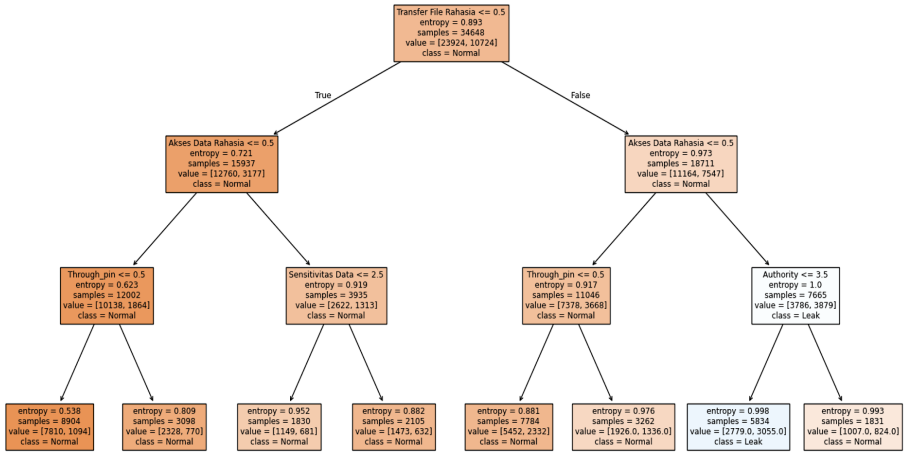

Figure 3. Decision tree visualization

3.3. Model Optimization and Results Analysis

To improve the model's performance, hyperparameter optimization is performed using GridSearchCV. This technique systematically searches for the best combination of hyperparameters that yields the highest model performance. The evaluation results of the optimized model are presented in Figure 4, showing a significant improvement compared to the initial results.

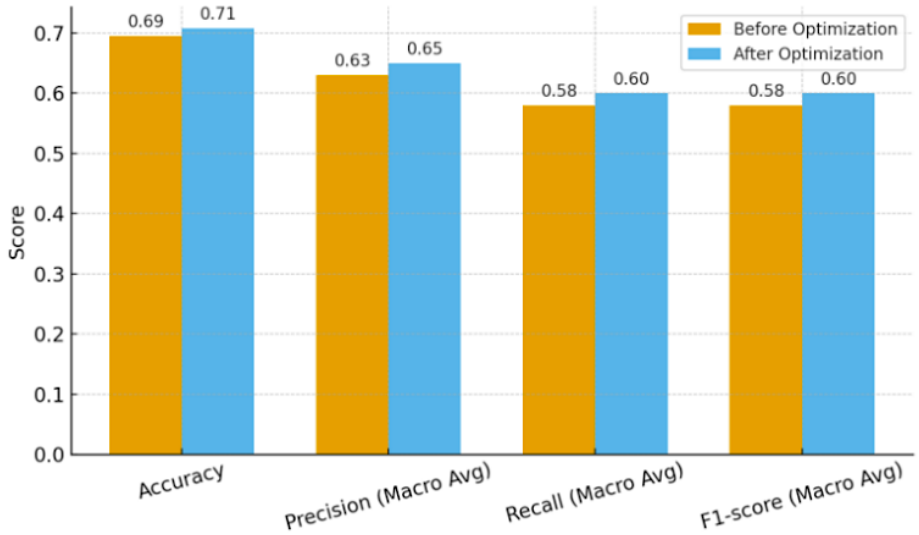

Figure 4. The comparison chart results after optimization

The comparison chart illustrates the performance of the Decision Tree model before and after hyperparameter optimization using GridSearchCV. The optimized model shows consistent improvements across all key evaluation metrics, with accuracy increasing from 0.69 to 0.71, macro-averaged precision from 0.63 to 0.65, recall from 0.58 to 0.60, and F1-score from 0.58 to 0.60. Although the improvements appear modest, they indicate that the optimized Decision Tree achieved a better balance between detecting normal and anomalous activities, which is particularly important in addressing the dataset's class imbalance. This suggests that parameter tuning enhanced the model's generalization ability and reduced bias toward the majority class. Although the dataset contains a substantial number of records, the class distribution is moderately imbalanced, with approximately 72% normal activities and 28% anomalous activities. This imbalance poses a challenge for classification, as models may be biased toward predicting the majority class, which in this case corresponds to normal user behavior. While accuracy remained relatively high, it alone cannot fully capture the model's effectiveness in identifying anomalies. For this reason, additional metrics such as precision, recall, and F1-score were emphasized to provide a more comprehensive evaluation of model performance under imbalanced conditions.

The results demonstrated that the Decision Tree classifier was able to achieve a strong balance between precision and recall despite the imbalance. The relatively high recall value indicates that the model successfully detected a significant portion of anomalous activities, which is critical for preventing data leakage in cloud storage systems. Nevertheless, the class imbalance highlights the importance of considering advanced strategies such as re-sampling, cost-sensitive learning, or ensemble methods in future work to further improve the detection of minority-class instances and minimize false negatives. The confusion matrix of the optimized model is also presented in Figure 6, providing a clearer picture of the number of correct and incorrect predictions.

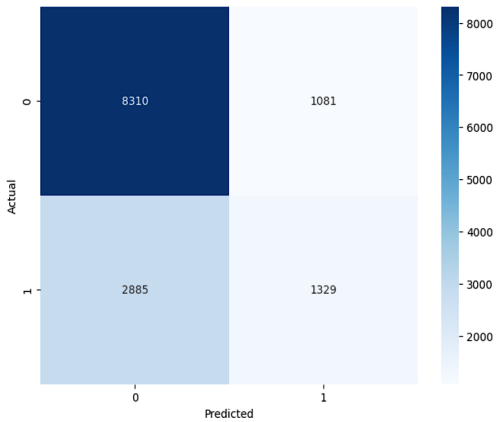

Figure 5. Confusion matrix

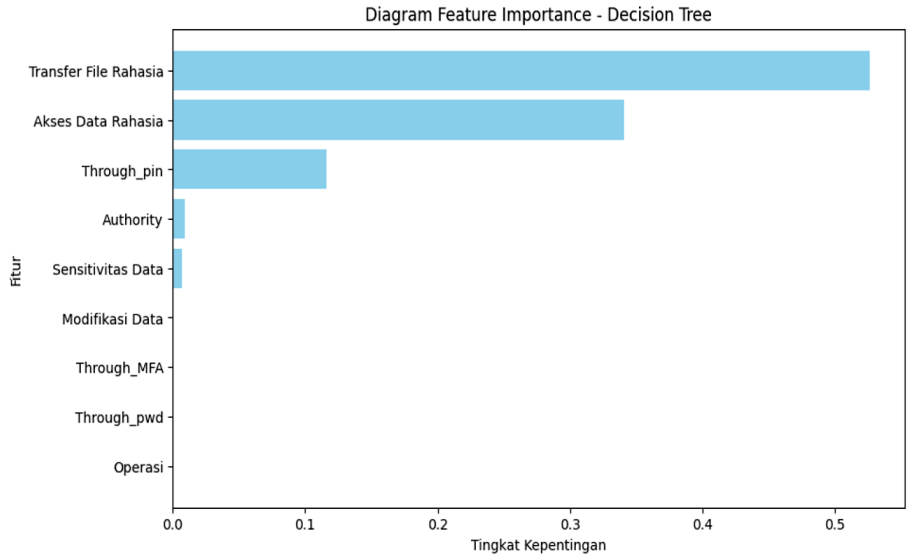

Figure 6. Feature importance diagram

A deep analysis of the results demonstrates that the optimized Decision Tree model serves as an effective solution for detecting potential data leakage in cloud storage environments. This conclusion is supported by the consistent improvements observed across multiple evaluation metrics after hyperparameter optimization, indicating a better balance between precision and recall despite the class imbalance in the dataset. Moreover, the model's interpretability provides additional value, as the generated rules can be easily understood and implemented by IT security teams, making the approach both practical and applicable in real-world monitoring systems.

Further examination of feature importance highlights that attributes such as Confidential_Data_Access and Data_Modification play a central role in predicting anomalous activities. As illustrated in Figure 7, these features significantly influence the model's decision-making process, confirming their relevance as indicators of suspicious behavior. This insight not only validates the initial research objectives but also contributes theoretically by emphasizing the role of behavioral and access-related features in machine learning-based security frameworks. Consequently, the findings demonstrate that the proposed model is capable of accurately capturing the critical patterns associated with data leakage, offering a scalable foundation for future research and practical deployment.

While this research focuses on the Decision Tree, it is worth noting that some similar studies use other machine learning algorithms such as Random Forest or Support Vector Machine (SVM) for anomaly detection tasks. For example, other

research shows that Random Forest often provides slightly higher accuracy than Decision Tree due to its ability to better handle overfitting by aggregating multiple decision trees. However, the Decision Tree offers better interpretability, which is crucial for security teams to understand the reasoning behind each anomaly detection.

From a practical implementation standpoint, the developed Decision Tree model can be integrated into a real-time cloud storage monitoring system. Once trained, the model can instantly process user activity logs to identify suspicious behavior. Whenever Confidential_Data_Access activity is detected, the model can provide a risk score that triggers an alert to the security team, allowing them to act proactively before data leakage occurs. This model can also serve as a basis for an adaptive alert system that adjusts detection thresholds based on historical activity patterns [24], [25].

3.4. Discussion

The results presented above demonstrate the successful application of a Decision Tree model for detecting anomalous activities in cloud storage environments, particularly for identifying potential data leakage. The data preprocessing stage, which included handling missing values, managing outliers, and using visualizations like the pairplot and pivot tables, played a crucial role in preparing the dataset for model training and providing initial insights into the relationships among key features. This early exploration provided valuable context for feature selection and model evaluation.

The initial classification model using the Decision Tree algorithm was a suitable choice, given its ability to handle both numerical and categorical data. Additionally, the interpretability of the Decision Tree is a key strength, making it easier to understand the decision-making process of the model, which is particularly important in security applications where explanations of model predictions are necessary for human intervention. Figure 3, which shows the Decision Tree visualization, highlights the clarity and transparency that this model offers to security analysts, helping them pinpoint specific conditions under which anomalous behavior is detected.

Following the initial model evaluation, the decision to optimize the model using GridSearchCV was an important step in improving performance. The improvements in accuracy, precision, recall, and F1-score, though modest, show a meaningful enhancement in the model's ability to balance the detection of both normal and anomalous activities. This result is particularly significant in the context of an imbalanced dataset, where a higher proportion of normal activities (72%) could lead to model bias toward predicting the majority class. The increase in recall,

from 0.58 to 0.60, is a notable improvement, as it indicates the model's enhanced ability to identify anomalous activities—an essential feature for anomaly detection in real-world security systems.

The impact of hyperparameter optimization using GridSearchCV was evident in the improved metrics. While accuracy remains a key performance indicator, other metrics such as precision, recall, and F1-score provide a more nuanced evaluation of the model's performance in imbalanced settings. In particular, the model's improved recall suggests a reduction in false negatives, which is crucial for anomaly detection tasks were missing an anomaly (false negative) could have severe consequences, such as data leakage. Although these improvements were not dramatic, they are sufficient to demonstrate the positive impact of parameter tuning, reinforcing the model's ability to generalize well to unseen data.

Further analysis of the confusion matrix (Figure 5) and feature importance (Figure 6) provided additional insights. The confusion matrix confirmed the balanced nature of the model's predictions, indicating that both normal and anomalous activities were well-represented in the predictions. The feature importance analysis revealed that "Confidential_Data_Access" and "Data_Modification" were among the most influential features in identifying anomalies. This aligns with the intuition that these activities are central to detecting potential data leakage, and it emphasizes the importance of focusing on access-related and behavioral features in security models.

Comparing the Decision Tree model with other machine learning techniques, such as Random Forest or Support Vector Machines (SVM), is essential for understanding the strengths and limitations of the chosen model. While Random Forest may outperform the Decision Tree in terms of raw accuracy due to its ability to reduce overfitting, the Decision Tree's interpretability remains a significant advantage. In security applications, where understanding the rationale behind anomaly detection is crucial, the Decision Tree's ability to provide clear, rule-based decision-making remains a valuable asset.

From a practical standpoint, the Decision Tree model can be integrated into real-time cloud storage monitoring systems to detect anomalous behavior promptly. By analyzing user activity logs and providing risk scores based on detected anomalies, the model can trigger alerts that allow security teams to respond proactively to potential threats. Furthermore, the model's scalability and adaptability suggest it can be used in large cloud storage environments, where rapid detection of abnormal activities is essential to prevent data leakage.

The optimized Decision Tree model has demonstrated its potential for effectively detecting anomalies and data leakage in cloud storage systems. The improvements

in evaluation metrics after hyperparameter tuning, combined with the model's interpretability and the relevance of selected features, underscore the model's practical applicability in real-world security settings. Future work could explore the use of more advanced techniques like ensemble methods, re-sampling strategies, or cost-sensitive learning to further improve anomaly detection, particularly for minority-class instances. The insights derived from this model provide a solid foundation for future research and practical deployment in cloud security applications.

4. CONCLUSION

Based on the findings and discussions, it can be concluded that the Decision Tree algorithm, employing the CART approach and enhanced by hyperparameter optimization through GridSearchCV, offers a robust and reliable solution for detecting data leakage in cloud storage systems. The model effectively distinguishes between normal and anomalous activities, affirming the initial research hypothesis. Key to the success of the model was the comprehensive data preprocessing stage, which involved handling missing values, detecting outliers, and encoding categorical variables. These steps were crucial in ensuring the quality and generalizability of the model, ultimately leading to accurate predictions. The optimized model, with parameters like criterion='entropy' and max_depth=3, demonstrated strong performance with a precision value of 0.74 for the normal class and 0.55 for the leakage class, alongside an overall F1-score of 0.68. The clarity of the decision tree visualization further enhances the model's interpretability, making it a valuable tool for security teams to understand and act upon detected anomalies.

Despite the promising results, the study acknowledges several limitations that could impact the model's generalizability. The reliance on a synthetic dataset may restrict its ability to fully capture the complexity of real-world cloud environments with varying user behaviors and data patterns. Additionally, while hyperparameter optimization helped mitigate overfitting, Decision Trees inherently have a tendency to overfit, which could affect model robustness. To address these limitations, future research could focus on testing the model in real-world scenarios, particularly in live cloud environments, to evaluate its scalability and practical effectiveness. Additionally, comparing the Decision Tree with other ensemble-based algorithms, such as Random Forest or XGBoost, could yield insights into achieving higher accuracy and reliability. Ultimately, deploying this model in a production setting would provide critical feedback and advance the field of anomaly detection in cybersecurity.

REFERENCES

[1] D. D. Firmansyah Putri and M. H. Fahrozi, "Upaya Pencegahan Kebocoran Data Konsumen Melalui Pengesahan Ruu Perlindungan Data Pribadi (Studi Kasus E-Commerce Bhinneka.Com)," *Borneo Law Rev.*, vol. 5, no. 1, pp. 46–68, 2021, doi: 10.35334/bolrev.v5i1.2014.

[2] L. Tantowi and L. Wijayanti, "Peluang Dan Tantangan Penyimpanan Cloud Storage Pada Dokumen Digital," *Shaut Al-Maktabah J. Perpustakaan, Arsip dan Dokumentasi*, vol. 15, no. 1, pp. 118–131, 2023, doi: 10.37108/shaut.v15i1.803.

[3] R. Rifany, M. D. Prakoso, and P. D. Laksono, "Analisis Dampak Cloud Computing terhadap Keamanan Sistem dan Data," *Semin. Nas. TEKNOKA*, vol. 8, no. 2502, pp. 01–06, 2023.

[4] A. F. Mahmud and S. Wirawan, "Sistemasi: Jurnal Sistem Informasi Deteksi Phishing Website menggunakan Machine Learning Metode Klasifikasi Phishing Website Detection using Machine Learning Classification Method," vol. 13, no. 4, pp. 2540–9719, 2024.

[5] M. Fadhlurrohman, A. Muliawati, and B. Hananto, "Analisis Kinerja Intrusion Detection System pada Deteksi Anomali dengan Metode Decision Tree Terhadap Serangan Siber," *J. Ilmu Komput. dan Agri-Informatika*, vol. 8, no. 2, pp. 90–94, 2021, doi: 10.29244/jika.8.2.90-94.

[6] A. Halim Lubis, Y. Fadillah Harahap, and P. Studi Ilmu Komputer, "Analisis Sentimen Masyarakat Terhadap Resesi Ekonomi Global 2023 Menggunakan Algoritma Naïve Bayes Classifier," *J. Ilm. Elektron. Dan Komput.*, vol. 16, no. 2, pp. 442–450, 2023.

[7] M. S. Hasibuan and A. Serdano, "Analisis Sentimen Kebijakan Pembelajaran Tatap Muka Menggunakan Support Vector Machine dan Naive Bayes," *JRST (Jurnal Ris. Sains dan Teknol.*, vol. 6, no. 2, pp. 199–204, 2022.

[8] M. R. Fatiha, I. Setiawan, A. N. Ikhsan, and I. R. Yunita, "Optimisasi Sistem Deteksi Phishing Berbasis WeB," *J. Ilm. IT CIDA*, vol. 10, no. 2, pp. 97–108, 2024.

[9] S. Yuan, H. Li, X. Qian, W. Jiang, and G. Xu, "OnePath: Efficient and Privacy-Preserving Decision Tree Inference in the Cloud," *arXiv (Cornell Univ.*, pp. 1–12, 2024, doi: arXiv:2409.19334.

[10] M. A. Nugroho and R. Kartadie, "Cloud Storage Dengan Teknologi Kubernetes Untuk Platform Collaborative Research," *JIPI (Jurnal Ilm. Penelit. dan Pembelajaran Inform.*, vol. 6, no. 1, pp. 74–81, 2021, doi: 10.29100/jipi.v6i1.1908.

[11] A. C. Darmawan, "Pengembanga Aplikasi Berbasis Web dengan Python Flask untuk Klasifikasi Data Menggunakan Metode Decision Tree C4.5," Universitas Islam Indonesia, 2022.

[12] A. Fahri and Y. Ramdhani, "Visualisasi Data dan Penerapan Machine Learning Menggunakan Decision Tree Untuk Keputusan Layanan Kesehatan COVID-19," *J. Tekno Kompak*, vol. 17, no. 2, p. 50, 2023, doi: 10.33365/jtk.v17i2.2438.

[13] R. N. Ramadhon, A. Ogi, A. P. Agung, R. Putra, S. S. Febrihartina, and U. Firdaus, "Implementasi Algoritma Decision Tree untuk Klasifikasi Pelanggan Aktif atau Tidak Aktif pada Data Bank," *Karimah Tauhid*, vol. 3, no. 2, pp. 1860–1874, 2024, doi: 10.30997/karimahtauhid.v3i2.11952.

[14] D. A. Setyawan, "Pengembangan Metode Decision Tree Dengan Diskritisasi Data Dan Splitting Atribut Menggunakan Hierarchical Clustering Dan," Institut Teknologi Sepuluh Nopember Surabaya, 2020.

[15] S. M. Prasetiyo, T. U. Ningsih, B. Hakim, and A. A. R. Putra, "Jurnal Managemen Proyek Informatika Artificial Intelligence Vision Engineer," *BULLET J. Multidisiplin Ilmu*, vol. 01, no. 6, pp. 987–991, 2022.

[16] M. Ţălu, "Exploring Machine Learning Algorithms to Enhance Cloud Comput- ing Security," *Digit. Technol. Res. Appl.*, vol. 4, no. 2, pp. 33–47, 2025, doi: 10.54963/dtra.v4i2.1272.

[17] A. B. Nassif, M. A. Talib, Q. Nasir, H. Albadani, and F. M. Dakalbab, "Machine Learning for Cloud Security: A Systematic Review," *IEEE Access*, vol. 9, pp. 20717–20735, 2021, doi: 10.1109/ACCESS.2021.3054129.

[18] S. V. Bhaskaran and S. Achar, "a Study of Evolving Cloud Computing Data Security: a Machine Learning Perspective," *Int. J. Prof. Bus. Rev.*, vol. 10, no. 3, p. e05315, 2025, doi: 10.26668/businessreview/2025.v10i3.5315.

[19] Z. M. J. Nafis, R. Nazilla, R. Nugraha, and S. 'Uyun Shofwatul 'Uyun, "Perbandingan Algoritma Decision Tree dan K-Nearest Neighbor untuk Klasifikasi Serangan Jaringan IoT," *Komputika J. Sist. Komput.*, vol. 13, no. 2, pp. 245–252, 2024, doi: 10.34010/komputika.v13i2.12609.

[20] F. A. Oktavirahani and R. Maharesi, "Implementasi Algoritma Decision Tree Cart Untuk Merekomendasikan Ukuran Baju," *JURIKOM (Jurnal Ris. Komputer)*, vol. 9, no. 1, p. 138, 2022, doi: 10.30865/jurikom.v9i1.3838.

[21] A. Rasyid, S. Gilbijatno, A. W. Pramudya, D. Prasetyo, and T. Informatika, "Implementasi Algoritma Decision Tree CART untuk Deteksi Dini," *Pros. Semin. Nas. Teknol. Dan Sains Tahun*, vol. 4, pp. 440–445, 2025.

[22] D. Muriyatmoko, A. Musthafa, and M. H. Wijaya, "Klasifikasi Profil Kelulusan Nilai AKPAM Dengan Metode Decision Tree," *Semin. Nas. Sains dan Teknol. 2024 Fak.*, no. April, pp. 448–453, 2024.

[23] R. E. Nugroho, W. Y. Pamungkas, and J. H. Jaman, "Pendeteksi Penyakit Hepatitis Menggunakan Cart Decision Tree," *J. Inform. dan Tek. Elektro Terap.*, vol. 12, no. 3S1, pp. 3690–3696, 2024, doi: 10.23960/jitet.v12i3s1.5184.

[24] R. Muzayanah, D. A. A. Pertiwi, M. Ali, and M. A. Muslim, "Comparison of gridsearchcv and bayesian hyperparameter optimization in random forest algorithm for diabetes prediction," *J. Soft Comput. Explor.*, vol. 5, no. 1, pp. 86–91, 2024, doi: 10.52465/joscex.v5i1.308.

[25] K. Alemerien, S. Alsarayreh, and E. Altarawneh, "Diagnosing Cardiovascular Diseases using Optimized Machine Learning Algorithms with GridSearchCV," *J. Appl. Data Sci.*, vol. 5, no. 4, pp. 1539–1552, 2024, doi: 10.47738/jads.v5i4.280.