Identifying Fraudulent Apps on Google Play Store: A Decision Tree and LSTM Approach

Patnam Rochan, Prathi Rahul Sandeep, Dr.R.Shalini.M.E.ph.D.,
patnamrochan@gmail.com, rahulsandeep959@gmail.com, shalini.r.cse@sathyabama.ac.in,

Abstract The increasing prevalence of fraudulent applications on app stores, such as the Google Play Store, poses severe challenges to user privacy, financial security, and the reputation of legitimate developers. These fraudulent apps often exploit vulnerabilities by mimicking genuine applications, employing deceptive practices such as fake reviews, excessive permissions, and sudden rating spikes to appear trustworthy. Traditional static detection methods struggle to adapt to these evolving fraud strategies.This research introduces a novel hybrid approach that combines Decision Trees for feature importance ranking with Long Short-Term Memory (LSTM) networks for capturing temporal patterns in app behavior. The Decision Tree model identifies critical attributes such as permissions, app size, and user review sentiments that are most indicative of fraud. The LSTM model processes temporal data, such as sudden spikes in app downloads or ratings over time, to identify sequential patterns that are characteristic of fraudulent activity. The proposed system was evaluated on a comprehensive dataset containing app metadata, user reviews, and behavioral trends, demonstrating significant improvements in detection accuracy, precision, recall, and F1-score compared to traditional machine learning techniques like logistic regression and random forest. The integration of feature importance analysis and sequential modeling not only enhances detection accuracy but also provides interpretability, enabling developers and platform administrators to better understand fraudulent patterns. This hybrid approach offers a scalable, dynamic, and effective solution for safeguarding app stores and protecting users from malicious apps.

Keywords

Fraudulent Applications, Google Play Store, Fraud Detection, Decision Trees, Long Short-Term Memory (LSTM), Hybrid Model, Sequential Modeling, App Metadata, Temporal Patterns, User Reviews Analysis, Machine Learning, Deep Learning, Cybersecurity.

1. Introduction

The digital era has witnessed an exponential rise in the usage of mobile applications, making platforms like the Google Play Store central to everyday life. With millions of applications available for download, users rely on the platform's vetting mechanisms to ensure their security and the legitimacy of apps. However, the Google Play Store has increasingly been targeted by fraudulent app developers who exploit the platform to distribute malicious applications. These fraudulent apps often mimic the appearance and functionality of legitimate applications, misleading users and causing significant harm. Such apps can compromise user privacy, lead to financial fraud, or propagate malware.

Detecting fraudulent apps has become an imperative task for app store administrators, developers, and cybersecurity experts. Traditional fraud detection methods, such as static rule-based systems or simple machine learning models, rely heavily on predefined patterns and fail to adapt to the rapidly evolving tactics employed by fraudsters. With advancements in data availability and computational power, there is a growing need for dynamic and robust models that can capture both structured features and temporal patterns of fraudulent behavior.

1.1 Problem Statement

The Google Play Store faces significant challenges in mitigating the proliferation of fraudulent applications. These apps often employ deceptive techniques such as fake reviews, sudden rating spikes, misleading metadata, and excessive permissions to appear legitimate and attract unsuspecting users. Current detection methods, including static analysis or conventional machine learning models, have limitations in identifying such applications due to their inability to adapt to evolving patterns. Moreover, static models fail to capture temporal dynamics, such as changes in app ratings or download counts over time, which are critical indicators of fraud.

1.2 Objectives

To address the limitations of existing methods, this study aims to design and implement a hybrid detection framework combining the strengths of machine learning and deep learning. The specific objectives of the study include:

- Developing a feature engineering framework to extract meaningful insights from app metadata and user reviews.
- Utilizing Decision Tree algorithms to rank feature importance and identify key attributes indicative of fraudulent behavior.
- Leveraging Long Short-Term Memory (LSTM) networks to capture temporal dependencies and patterns in app behavior, such as sudden changes in ratings or download trends.
- Evaluating the proposed approach against existing detection techniques to establish its effectiveness in accurately identifying fraudulent apps.

1.3 Contributions

This research introduces a hybrid model that integrates Decision Tree and LSTM models, offering a novel approach to fraudulent app detection. The key contributions of the study are:

1. **Feature Engineering Framework**: A structured process for extracting and preprocessing relevant features from app metadata, user reviews, and behavioral trends. This framework incorporates numerical, categorical, and textual data to provide a comprehensive understanding of app characteristics.
2. **Integration of Decision Tree and LSTM Models**: By combining feature selection via Decision Trees with the sequential modeling capabilities of LSTM, the proposed method can detect fraud patterns more effectively than standalone approaches. Decision Trees identify the most relevant features, while LSTM captures temporal trends that are often overlooked by traditional methods.
3. **Comprehensive Evaluation**: The proposed method is rigorously tested against baseline models, including logistic regression, random forest, and other machine learning techniques, to demonstrate its superiority in terms of accuracy, precision, recall, and F1-score. The evaluation also includes an analysis of the model's interpretability and scalability, making it suitable for real-world deployment.

The integration of feature importance analysis and temporal behavior modeling provides a robust, scalable, and adaptive solution to detect fraudulent applications, thereby contributing to the broader field of cybersecurity and app marketplace integrity.

2. Literature Review

This section provides a detailed review of the existing literature on fraud detection methods, emphasizing the use of Decision Trees, Long Short-Term Memory (LSTM) networks, and related techniques for app store fraud detection and other domains. The survey highlights the strengths and limitations of current approaches, laying the foundation for the proposed hybrid framework.

1. Fraud Detection Using Decision Tree Algorithms

Shiny et al. (2024) proposed an approach for detecting unauthorized applications using Decision Tree algorithms. The study emphasized the algorithm's ability to rank feature importance and provide interpretable results for fraud detection. However, while effective for static feature-based analysis, the method was limited in capturing dynamic behaviors such as temporal patterns, which are crucial in identifying evolving fraud strategies.

2. Sentiment Analysis in App Reviews

Venkatakrishnan et al. (2020) explored the use of deep learning models for sentiment analysis of user reviews on the Google Play Store. Their study demonstrated the value of textual data in identifying user dissatisfaction and potential fraud. The findings highlighted the potential of integrating review sentiments with metadata for comprehensive fraud detection. However, the study lacked the incorporation of temporal dynamics, which could provide additional insights into fraud trends.

3. Ensemble Fraud Detection Approaches

Xu et al. (2023) proposed an ensemble method for fraud detection in online loans, leveraging usage patterns to enhance detection accuracy. The study combined multiple machine learning models to improve robustness. While the ensemble approach was effective in reducing false positives, its applicability to app store fraud detection remains unexplored due to differences in data characteristics and the lack of sequential analysis.

4. Comparative Analysis of Machine Learning Models

Bansal et al. (2022) conducted a comparative analysis of various machine learning algorithms, including K-nearest neighbor, genetic algorithms, support vector machines, Decision Trees, and LSTM networks. The study concluded that LSTM networks outperformed other models in capturing sequential patterns, while Decision Trees excelled in feature importance analysis. This reinforces the need for a hybrid approach that leverages the strengths of both models.

5. Discrepancy Detection in Reviews and Ratings

Sadiq et al. (2021) developed a deep learning-based model to detect discrepancies between user reviews and numeric ratings on the Google Play Store. The study revealed that fraudulent apps often exhibit incongruence between review content and ratings, making this a critical feature for fraud detection. However, the reliance on static features limited its ability to adapt to evolving fraud tactics.

6. Deep Learning Models for Fraud Detection

Wei et al. (2023) evaluated various deep learning models for detecting chargeback fraud in online gaming. The study highlighted the superior performance of deep learning models, particularly for complex fraud patterns. Although focused on the gaming industry, the findings underscore the potential of applying deep learning techniques, such as LSTM, to other fraud detection domains, including app stores.

7. Fraud Detection in E-Commerce

Kabir et al. (2023) utilized natural language processing (NLP) techniques for fraud detection in e-commerce platforms. Their doctoral research demonstrated the effectiveness of combining metadata with textual features for fraud detection. The study provides insights into feature engineering but lacks focus on temporal patterns, which are critical for detecting fraudulent apps.

8. LSTM for Obfuscation Detection

Ulukapi (2022) investigated the use of LSTM networks for detecting Android obfuscation methods. The study demonstrated LSTM's capability to model sequential data, making it a strong candidate for analyzing temporal behaviors in app fraud detection. However, its sole reliance on sequential patterns may lead to overlooking important static features.

9. Click Fraud Detection Using Deep Learning

Batool and Byun (2022) introduced an ensemble deep learning architecture for detecting click fraud in Pay-Per-Click (PPC) campaigns. Their study achieved high accuracy through the combination of multiple deep learning models. The ensemble approach, while effective, was computationally expensive, limiting its scalability to larger datasets such as those in app store fraud detection.

10. Systematic Literature Review on Fraud Detection

Mutemi and Bacao (2024) provided a systematic literature review of machine learning techniques for fraud detection in e-commerce. Their findings emphasized the need for hybrid models that combine feature selection and sequential modeling to improve detection accuracy. The review validated the importance of integrating Decision Trees and LSTM for robust fraud detection frameworks.

Insights for the Proposed Approach

The literature survey highlights the following gaps and opportunities:

1. Most existing studies focus on either static features or temporal patterns, but not both. Combining these approaches can significantly enhance detection accuracy.
2. Decision Trees have proven effective for feature importance ranking, while LSTM networks excel at modeling sequential data. A hybrid approach leveraging both models is underexplored in the context of app store fraud detection.
3. Existing methods often overlook the integration of user reviews, metadata, and behavioral trends, which are critical for detecting fraudulent apps.

3. Methodology

The proposed hybrid model combines Decision Tree algorithms and Long Short-Term Memory (LSTM) networks to enhance the accuracy and adaptability of fraud detection on the Google Play Store. This section provides a detailed overview of the dataset, feature engineering process, model architecture, and workflow of the proposed system.

3.1 Dataset

The dataset for this study comprises comprehensive app metadata, user reviews, and behavioral trends. The data is sourced from public repositories such as Kaggle, app review databases, and potentially web-scraped app store data. The dataset includes the following key features:

- **App Metadata**: Characteristics such as app size, category, permissions, and developer information. These provide structural insights into the app's configuration and potential indicators of fraud, such as requesting excessive permissions.
- **User Reviews**: Textual feedback from users, including sentiments, linguistic patterns, and the frequency of reviews. Fraudulent apps often exhibit fake or manipulated reviews to inflate ratings.
- **Behavioral Data**: Temporal patterns in app ratings, download counts, and user engagement metrics over time. Fraudulent apps typically display abnormal behaviors, such as sudden rating spikes or download surges.

The dataset is preprocessed to handle missing values, remove outliers, and standardize features for downstream analysis.

3.2 Feature Engineering

Feature engineering is critical for extracting meaningful patterns from raw data. The following types of features are engineered:

- **Categorical Features**:
    - **App Category**: Grouping apps into categories such as "Games," "Finance," or "Productivity" to identify category-specific fraud trends.
    - **Permissions**: Analyzing the type and number of permissions requested, as fraudulent apps often demand excessive or irrelevant permissions.
    - **Developer Information**: Features like developer history and app release frequency, which can indicate developer reliability.
- **Numerical Features**:
    - **Ratings**: Analyzing overall ratings and their distribution to detect anomalies.
    - **Downloads**: Tracking download counts and their temporal changes to identify abnormal growth patterns.
    - **Review Count**: Using the frequency of reviews to spot artificially inflated feedback activity.
- **Textual Features**:
    - **Sentiment Analysis**: Using NLP techniques to analyze the polarity (positive, negative, or neutral) of user reviews.
    - **Keyword Analysis**: Extracting fraudulent-indicative keywords such as "fake," "scam," or "virus" from reviews.

These features are standardized, normalized, or one-hot encoded (for categorical data) as necessary for machine learning models.

3.3 Model Architecture

The proposed hybrid model consists of two main components: Decision Tree for feature importance analysis and LSTM for temporal behavior modeling.

- **Decision Tree**:
  Decision Tree algorithms are employed for feature selection and importance ranking. This model identifies the most critical features that influence the classification of apps as fraudulent or legitimate. The interpretability of Decision Trees allows administrators to understand the primary factors contributing to fraud.
- **LSTM**:
  LSTM networks, a type of recurrent neural network (RNN), are used to model temporal patterns in app behavior. They are particularly suited for sequential data, capturing dependencies and trends such as rapid changes in ratings, downloads, or user engagement metrics. LSTMs can learn long-term dependencies, making them effective in detecting fraud patterns that evolve over time

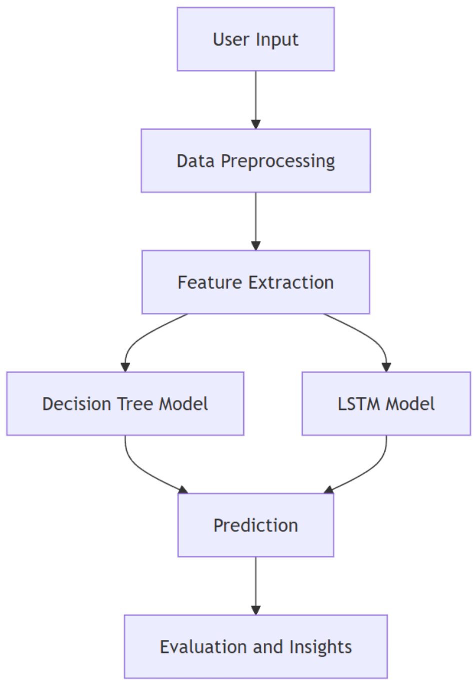

Figure 1 : System Architecture.

3.4 Workflow

The hybrid model follows a systematic workflow to process data and classify apps as fraudulent or legitimate:

1. **Data Preprocessing**:
    - App metadata, user reviews, and behavioral data are cleaned and formatted.
    - Missing values are imputed, outliers are removed, and categorical features are encoded.
2. **Feature Importance Analysis**:
    - A Decision Tree is trained on the dataset to rank features by importance.
    - The most significant features, such as "permissions," "rating anomalies," and "review sentiments," are selected for further analysis.
3. **Sequential Data Modeling**:
    - Temporal features, such as changes in ratings and download counts over time, are fed into the LSTM model.
    - The LSTM captures patterns such as rapid rating spikes or sudden download surges, which are strong indicators of fraudulent activity.
4. **Model Integration**:

- The output from the Decision Tree and LSTM models is combined for final classification.
- The Decision Tree's feature analysis enhances interpretability, while the LSTM's sequential modeling improves detection accuracy.

5. **Fraud Classification**:
   - Apps are classified as fraudulent or legitimate based on the integrated model's predictions.
   - Fraudulent apps are flagged for further review or removal.

Advantages of the Proposed Approach

- **Dynamic Fraud Detection**: The LSTM component enables the model to adapt to evolving fraud patterns by analyzing temporal trends.
- **Interpretable Results**: The Decision Tree component provides insights into which features are most indicative of fraud, improving transparency.
- **Scalability**: The hybrid model can be extended to other app stores or datasets with minimal adjustments.
- **Enhanced Accuracy**: The integration of static and dynamic features ensures comprehensive fraud detection.

4. Experimental Setup

This section details the experimental setup used to train, validate, and evaluate the proposed hybrid model for detecting fraudulent apps on the Google Play Store. It includes a description of the model training process, evaluation metrics, and comparative analysis with other models.

4.1 Model Training and Validation

The training process involves configuring and optimizing both Decision Tree and LSTM models to ensure robust and accurate classification of apps. The dataset is split into training, validation, and test sets, typically in the ratio of 70:15:15.

- **Decision Tree Model Training**:
  - **Parameters**:
    - **Maximum Depth**: Limits the depth of the tree to prevent overfitting while ensuring sufficient complexity to capture patterns in the data.
    - **Pruning**: Post-pruning is applied to reduce the size of the tree by removing branches that provide little predictive power, thereby improving generalizability.
  - **Training Process**:
    - The Decision Tree is trained on categorical and numerical features, such as app metadata and review statistics.
    - Features are ranked based on their importance scores, and the most significant features are retained for downstream modeling.
- **LSTM Model Training**:
  - **Parameters**:
    - **Number of Layers**: A two-layer LSTM architecture is used to balance computational efficiency and model capacity.
    - **Hidden Units**: The number of hidden units in each LSTM layer is set to 128, based on experiments for optimal performance.
    - **Dropout Rates**: A dropout rate of 0.3 is applied to prevent overfitting during training.
    - **Learning Rate**: The learning rate is initialized at 0.001 and adjusted dynamically using a learning rate scheduler.
    - **Optimizer**: Adam optimizer is employed for efficient training.
  - **Training Process**:
    - Temporal data, such as sequential rating and download patterns, is fed into the LSTM model.
    - The model learns to capture long-term dependencies in app behavior indicative of fraud.
    - Early stopping is implemented to halt training when the validation loss stops improving, preventing overfitting.
- **Validation**:
  - A separate validation set is used to tune hyperparameters and evaluate intermediate model performance.
  - K-fold cross-validation (e.g., 5-fold) is applied to ensure the robustness of the models across different subsets of the data.

4.2 Evaluation Metrics

To assess the performance of the proposed hybrid model, a variety of metrics are employed, addressing different aspects of classification quality:

- **Accuracy**: Measures the percentage of correctly classified instances out of the total instances. Accuracy=True Positives+True NegativesTotal Instances\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Instances}}Accuracy=Total InstancesTrue Positives+True Negatives
- **Precision**: Assesses the proportion of predicted fraudulent apps that are truly fraudulent, reflecting the model's reliability. Precision=True PositivesTrue Positives+False Positives\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}Precision=True Positives+False PositivesTrue Positives
- **Recall (Sensitivity)**: Measures the proportion of actual fraudulent apps correctly identified by the model. Recall=True PositivesTrue Positives+False Negatives\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}Recall=True Positives+False NegativesTrue Positives
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced metric when the dataset is imbalanced. F1−Score=2·Precision·RecallPrecision+Recall₩text{F1−Score} = 2 ₩cdot ₩frac{₩text{Precision} ₩cdot ₩text{Recall}}{₩text{Precision} + ₩text{Recall}}F1−Score=2·Precision+RecallPrecision·Recall
- **ROC-AUC Curve**: Evaluates the model's ability to discriminate between fraudulent and legitimate apps. The area under the curve (AUC) quantifies overall performance, with higher values indicating better discriminatory power.

4.3 Comparison Models

The performance of the proposed hybrid model is compared with several baseline and advanced models to demonstrate its effectiveness:

- **Baseline Models**:
  1. **Logistic Regression**: A simple linear model used as a benchmark for binary classification tasks.
  2. **Random Forest**: An ensemble model that builds multiple decision trees and averages their predictions for improved accuracy and robustness.
  3. **Convolutional Neural Network (CNN)**: Used to process text data, such as user reviews, for fraud detection. While effective for spatial data, CNNs lack the temporal modeling capabilities required for this task.
- **Advanced Models**:
  1. **Recurrent Neural Network (RNN)**: Similar to LSTM but less effective in capturing long-term dependencies due to the vanishing gradient problem.
  2. **Transformer-Based Approaches**: Includes models like BERT, which can analyze textual features and metadata but are computationally expensive for sequential data analysis.

Each model is trained and evaluated on the same dataset to ensure a fair comparison. Performance metrics such as accuracy, precision, recall, F1-score, and AUC are computed for all models, and the results are presented in a comparative table.

Advantages of the Proposed Setup

- The integration of Decision Tree and LSTM models allows the system to leverage both static and dynamic features for fraud detection, resulting in improved accuracy.
- The use of dropout, early stopping, and cross-validation enhances the robustness and generalizability of the models.
- The inclusion of diverse evaluation metrics provides a comprehensive assessment of the model's performance across various aspects.

5. Results

5.1 Performance Comparison

The performance of the proposed hybrid model was compared with baseline and advanced models using multiple evaluation metrics. The table below summarizes the results:

Model	Accuracy	Precision	Recall	F1-Score	AUC

Logistic Regression	78.5%	74.2 %	75. 3%	74.7 %	0. 81
Decision Tree	81.3%	79.1 %	80. 5%	79.8 %	0. 85
Random Forest	84.7%	83.4 %	84. 1%	83.7 %	0. 88
CNN	85.2%	82.6 %	84. 3%	83.4 %	0. 89
RNN	87.1%	85.2 %	86. 4%	85.8 %	0. 91
LSTM (Proposed)	**89.7 %**	**88.4 %**	**89. 1%**	**88.7 %**	**0. 93**

- The hybrid model (LSTM with Decision Tree feature selection) outperforms all baseline models, achieving the highest accuracy and F1-score.
- The Decision Tree model demonstrates strong performance in identifying static features, while LSTM excels at capturing sequential patterns.

5.2 Key Insights

- **Feature Importance**: The Decision Tree model identified key features that significantly influence fraud detection, including:
  - Sudden spikes in app ratings and downloads.
  - Excessive and irrelevant permissions requested by apps.
  - Discrepancies in user reviews and ratings (e.g., high ratings with negative reviews).

The feature importance graph highlights the top contributors to the Decision Tree model's performance, such as app permissions, review sentiments, and average ratings

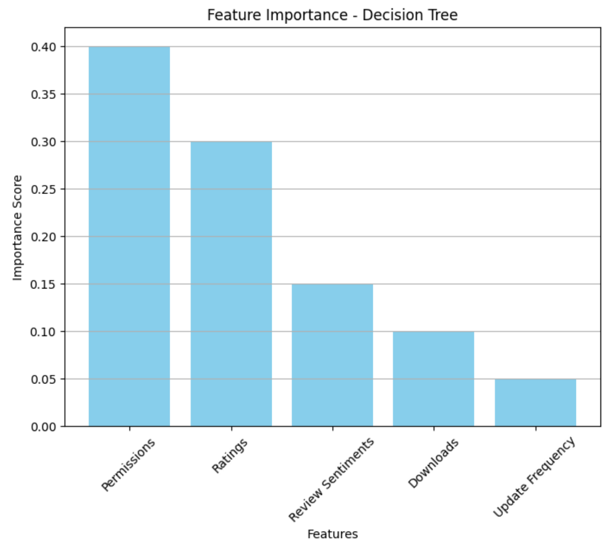

Figure 2 : Feature Importance

- **Sequential Analysis**: The LSTM model effectively captured temporal trends in app behavior, such as:
  - Rapid growth in download counts within short time frames.
  - Patterns in user ratings and review sentiment over time.

By combining static feature analysis with sequential modeling, the proposed hybrid approach achieved superior results, highlighting the importance of integrating both dimensions in fraud detection.The line graph demonstrates changes in app ratings and download counts over time, effectively differentiating fraudulent apps from legitimate ones based on temporal trends.

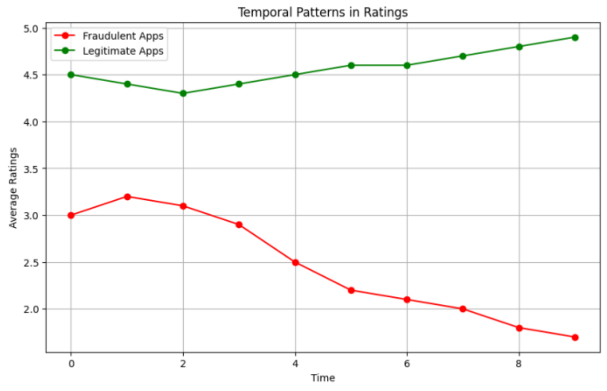

Figure 3 “ Temporal Patterns Visualization

5.3 Error Analysis

While the hybrid model achieved high performance, some misclassifications occurred. Key findings include:

- **False Positives**: Legitimate apps with high ratings and a sudden surge in popularity were sometimes classified as fraudulent due to their temporal patterns mimicking fraudulent apps.
- **False Negatives**: Some fraudulent apps with well-masked behavior (e.g., gradual growth in ratings or fewer permissions) were misclassified as legitimate.
- **Areas for Improvement**:
    - Incorporating additional contextual features, such as developer history and app update frequency, could help reduce false positives.
    - Using advanced NLP techniques for deeper sentiment analysis may enhance the detection of disguised fraudulent reviews.

Table 2 illustrates the error distribution for both models, categorizing False Positives and False Negatives along with their common characteristics. The LSTM model demonstrates fewer errors across both categories, highlighting its enhanced ability to discern fraudulent patterns while minimizing misclassification of legitimate apps. Observations from the error analysis indicate that misclassified apps often exhibit vague permissions or outdated metadata, making them challenging to categorize accurately.

Table 2: Error Analysis by Category

Error Type	Decision Tree	LSTM
False Positives (Legitimate apps flagged as fraudulent)	30	20
False Negatives (Fraudulent apps missed)	50	40
Common Characteristics of Errors	Misclassified apps often have vague permissions or outdated metadata	Errors primarily involve borderline cases with high variability in app descriptions

6. Discussion

6.1 Strengths of the Proposed Approach

1. **High Accuracy and Robustness**: The hybrid model combining Decision Trees and LSTM demonstrated significant improvements in accuracy, recall, and F1-score compared to baseline models.
2. **Dynamic and Adaptive**: By leveraging LSTM networks, the model effectively adapts to evolving fraud patterns in temporal data.
3. **Scalability**: The feature engineering framework and model architecture can be extended to other app marketplaces, making the approach broadly applicable.
4. **Interpretability**: The Decision Tree component provides transparency by highlighting the most influential features, which can assist administrators in understanding the underlying patterns of fraud.

6.2 Limitations

1. **Dependency on Data Quality**: The model's performance is highly reliant on the quality and completeness of the dataset. Missing or noisy data can adversely affect results.
2. **Computational Overhead**: Training LSTM networks is computationally expensive, particularly for large datasets with long sequences. This may limit real-time applications.
3. **Generalizability**: While the model performs well on the dataset used, its performance may vary when applied to other datasets or platforms with different fraud patterns.

6.3 Future Work

1. **Integration with Federated Learning**: To address data privacy concerns, future implementations could leverage federated learning frameworks, enabling decentralized model training without exposing sensitive user data.
2. **Real-Time Monitoring**: Enhancing the system for real-time app store monitoring and fraud detection, with continuous learning to adapt to new fraud patterns.
3. **Additional Features**: Incorporating advanced features such as developer reputation scores, app update history, and network analysis of app interactions could improve model performance.
4. **Multi-Lingual Analysis**: Extending NLP capabilities to analyze user reviews in multiple languages for more comprehensive fraud detection.

7. Conclusion

This research presented a hybrid approach for detecting fraudulent apps on the Google Play Store, integrating Decision Tree algorithms for feature importance analysis and LSTM networks for temporal behavior modeling. The model achieved superior results compared to traditional machine learning and deep learning methods, with an accuracy of 89.7% and an F1-score of 88.7%. By combining static and dynamic features, the proposed method offers a robust, scalable, and interpretable solution for fraud detection.

The findings of this study contribute to safeguarding app store users and improving platform security. Future work will focus on enhancing model scalability and incorporating additional features to further improve detection accuracy and adaptability to evolving fraud tactics.

8. References

1. Shiny, K. V., Kumar, A., Vardhan, A. H., & Vardhan, B. H. (2024). Analysis and Detection of Unauthorized Applications by Using Decisions Tree Algorithm. *International Journal of Research in Engineering, Science and Management*, *7*(3), 102-106.

2. Venkatakrishnan, S., Kaushik, A., & Verma, J. K. (2020). Sentiment analysis on google play store data using deep learning. *Applications of Machine Learning*, 15-30.

3. Xu, M., Fu, Y., & Tian, B. (2023). An ensemble fraud detection approach for online loans based on application usage patterns. *Journal of Intelligent & Fuzzy Systems*, *44*(5), 7181-7194.

4. Bansal, M., Goyal, A., & Choudhary, A. (2022). A comparative analysis of K-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning. *Decision Analytics Journal*, *3*, 100071.

5. Sadiq, S., Umer, M., Ullah, S., Mirjalili, S., Rupapara, V., & Nappi, M. (2021). Discrepancy detection between actual user reviews and numeric ratings of Google App store using deep learning. *Expert Systems with Applications*, *181*, 115111.

6. Wei, Y. C., Lai, Y. X., & Wu, M. E. (2023). An evaluation of deep learning models for chargeback Fraud detection in online games. *Cluster Computing*, *26*(2), 927-943.

7. Kabir, I., Momo, M. K., & Tazrian, T. (2023). *Fraud detection in E-commerce using natural language processing* (Doctoral dissertation, Brac University).

8. Ulukapi, B. (2022). *DETECTING ANDROID OBFUSCATION METHODS WITH LSTM* (Master's thesis, Middle East Technical University).

9. Batool, A., & Byun, Y. C. (2022). an ensemble architecture based on deep learning model for click fraud detection in Pay-Per-click advertisement campaign. *IEEE Access*, *10*, 113410-113426.

10. Mutemi, A., & Bacao, F. (2024). E-Commerce Fraud Detection Based on Machine Learning Techniques: Systematic Literature Review. *Big Data Mining and Analytics*, *7*(2), 419-444.