International Journal for Modern Trends in Science and Technology
Volume 11, Issue 11, pages 33-41.
ISSN: 2455-3778 online
Available online at: http://www.ijmtst.com/vol11issue10.html
DOI: https://doi.org/10.5281/zenodo.17583570

Fraud Sniffer: Detecting Anomalies in Transaction Data with Machine Learning

Manikonda Ramya Krishna

Assistant Professor, Vikas College of engineering and Technology, A.P, India.

To Cite this Article
Manikonda Ramya Krishna (2025). Fraud Sniffer: Detecting Anomalies in Transaction Data with Machine Learning. International Journal for Modern Trends in Science and Technology, 11(11), 33-41. https://doi.org/10.5281/zenodo.17583570

Article Info
Received: 06 October 2025; Accepted: 05 November 2025.; Published: 11 November 2025.

KEYWORDS

Fraud Detection, Machine Learning, Streamlit, Classification, Real-Time Prediction, Financial Transactions, Model Deployment, Feature Engineering.

ABSTRACT

The project introduces a sophisticated real-time Fraud Detection System that uses machine learning to determine if online financial transactions are fraudulent or legitimate. As more payments are made online, it's become extremely important to spot fraud quickly to keep financial systems safe and build customer confidence. The system uses supervised learning, which means it's trained on big sets of transaction data that include different details like the amount of the transaction, when it happened, the age of the customer, how long they've had the account, what payment method was used, what product was bought, and the type of device used.

Before the data is used to train the models, it goes through a detailed cleaning process to make sure the results are accurate and dependable. This involves fixing missing data, converting categories into numbers that computers can use, making sure all numbers are in the right range, and finding any unusual data points. The system also uses feature engineering to create new useful information that helps the model make better decisions.

Several machine learning models, like Logistic Regression, Random Forest, Gradient Boosting, and XGBoost, are trained and tested using measures like accuracy, precision, recall, F1score, and ROC-AUC. This helps identify which model works best for the job.

Once the best model is chosen, it's saved using Pickle so it can be used in real-time. The front end of the system is built with Streamlit, making it easy to use and visually appealing. Users can enter transaction details manually or upload files. When they submit the data, the system quickly analyzes it and gives a result showing if the transaction is real or fake. The results are shown using clear signs and visuals, like colored labels and charts, so users can understand them easily.

To help users and businesses understand the system better, the interface includes clear explanations of each feature, insights into how the model makes decisions, and advice on how to prevent fraud. The design is also made more professional and easy to use with custom CSS styling.

Overall, this project shows how machine learning can be used to tackle financial fraud. The system is designed to be scalable, easy to understand, and fast, making it a strong tool for improving security and trust in online transactions. It can be easily connected to existing financial platforms to offer better protection.

1. INTRODUCTION

The project introduces a smart real-time Fraud Detection System that uses machine learning to check if online financial transactions are fake or real. As more people make payments online, it's very important to catch fraud quickly to protect financial systems and build trust with customers. The system uses supervised learning, which means it's trained on large amounts of transaction data that includes various details like the amount of the transaction, the time it happened, the customer's age, how long they've had the account, the payment method used, the product bought, and the type of device used.

Before the data is used to train the models, it goes through a detailed cleaning process to make sure the results are accurate and reliable. This includes fixing missing data, changing categories into numbers that computers can use, making sure all numbers are in the right range, and finding any strange or unusual data points. The system also uses feature engineering to create new useful information that helps the model make better decisions.

Several machine learning models, such as Logistic Regression, Random Forest, Gradient Boosting, and XGBoost, are trained and tested using metrics like accuracy, precision, recall, F1-score, and ROC-AUC. This helps determine which model works best for the task.

Once the best model is selected, it is saved using Pickle so it can be used in real-time. The front end of the system is built with Streamlit, making it simple to use and visually appealing. Users can enter transaction details manually or upload files. When they submit the data, the system quickly analyzes it and provides a result showing whether the transaction is real or fake. The results are shown using clear labels and visuals, like colored signs and charts, so users can understand them easily.

To help users and businesses better understand the system, the interface includes clear explanations of each feature, insights into how the model makes decisions, and tips on how to prevent fraud. The design is also made more professional and user-friendly with custom CSS styling.

Overall, this project demonstrates how machine learning can be used to fight financial fraud. The system is designed to be scalable, easy to understand, and fast, making it a strong tool for improving security and trust in online transactions. It can be easily integrated with existing financial platforms to offer better protection.

2. LITERATURE SURVEY

[1] Bettinger's work is one of the earliest examples of financial technology (FinTech) application in banking, illustrating how a series of 40 time-shared models were deployed at Manufacturers Hanover Trust Company to optimize banking operations. This pioneering effort is notable for setting the stage for modern FinTech developments by demonstrating the efficiency gains achievable through technology-driven banking systems. These models supported various financial functions including customer account management, fraud detection, and service delivery optimization. Even though the computing systems were primitive by today's standards, the approach reflected an early vision of distributed computing and automation in financial services. Bettinger's study is often referenced as a historical anchor that prefaces the rapid development of digital financial platforms. For customer segmentation, the use of shared models introduced the idea of centralized analytics feeding multiple service touchpoints—concepts now seen in cloud-based CRM and segmentation engines.

[2]Thakor gives a detailed look at how the FinTech industry is affecting traditional banks. He talks about changes in how businesses operate, the rules they have to follow, and how customers are behaving differently. The paper explains how FinTech companies use digital tools to provide customer-focused services like personalized loans, managing money, and detecting

fraud. Thakor also looks at how FinTech startups are competing with traditional banks, showing how their fast-moving and technologybased approaches are challenging older banking systems. He points out that using data analysis is key for FinTech businesses to better understand and serve their customers. He also discusses risks like cyber attacks and privacy issues, especially with open banking. The study shows how FinTech can improve customer understanding by using data from places like social media, mobile apps, and spending habits.

[3]This paper looks at how FinTech changed a lot after the 2008 financial crisis. Arner and other writers say that the financial problem helped push for new technology in banking and finance. This happened because of changes in rules, people losing trust in old banks, and new tech developments. The paper breaks down the main steps in FinTech's growth, starting with digital banking services and moving on to blockchain, AI, and better data analysis. It also explains how companies can connect better with customers by using smart ways to group people, which lets them give personalized services based on how people behave. The authors also point out that things like DeFi and peer-to-peer lending are changing how customers and businesses interact. This paper is important for understanding how customer grouping strategies need to change along with the digital changes in financial systems.

3. DATA SET

A.A. Source of Dataset

The dataset used in this project includes detailed records of individual transactions from a simulated or anonymized real-world e-commerce system. Each entry in the dataset is for a single transaction and includes extra information that helps determine if the transaction is real or fake. The dataset was imported and examined in the Jupyter notebook called fraud_detection.ipynb, and it was used to train and test the supervised learning model.

B. B. Dataset Summary Statistics

The dataset has some basic stats that give an idea of what's included. Here's a general overview: Total number of records is around 50,000, just as an example. About 2% of these are fraudulent transactions, which means there's an unequal spread between normal and fraudulent cases. The average transaction amount changes depending on the type of product being bought. The average time a customer has been using the account varies, from new users to those who have been around for a long time. The devices people use are mostly mobile phones, followed by desktop computers, and then tablets, based on how they're used.

TABLE I

FEATURE DESCRIPTION OF TRANSACTION DATASET

Feature Name	Description
Transaction Amount	Total monetary value of the transaction.
Payment Method	Type of payment used: Debit Card, Credit Card, PayPal, Bank Transfer.
Product Category	Category of the item purchased (e.g., Electronics, Toys, Clothing).
Customer Age	Age of the customer performing the transaction.
Device Used	Type of device used for the transaction: Desktop, Mobile, or Tablet.
Account Age Days	Age of the customer's account in days.
Transaction Hour	Hour of the day when the transaction occurred (0–23).
Day	Day of the month when the transaction occurred.
Month	Month of the year when the transaction occurred.
Class	Target label (0 = Genuine, 1 = Fraudulent).

C. Data Preprocessing

1) Handling Missing Values: Data quality is very important when training strong machine learning models. When we first looked at the dataset, we found missing or null values in some numerical fields like Transaction Amount, Customer Age, or Account Age. We took steps to fix these issues. The methods we used were:

- For numerical fields, missing values were filled in using either the average or the middle value, depending on how the data was spread out.
- For categorical fields, missing values were replaced with a default option or the most common category.
- In some cases, we chose to remove rows that had too many missing values or unusual data points to keep the model accurate and reliable.

This process helps make sure the model doesn't have problems or unfair results because of missing or bad data.

4. MACHINE LEARNING MODEL

A. Model Selection Rationale

Because the dataset had a table format with both categorical and numerical data, and because some classes were not balanced, we chose a decision tree-based model. Models like Random Forest and Gradient Boosted Trees (like XGBoost or LightGBM) are good at dealing with different types of data, they don't need much preparation, and they can handle strange values well.

For this project, we picked the Random Forest Classifier because: - It's easier to understand compared to other models that are harder to explain.

- It works well with imbalanced binary classification problems.
- It automatically gives information about which features are most important.
- It can avoid overfitting if tuned correctly.

B. Evaluation Metrics Used

Because fraud detection usually has a lot more normal transactions than fraudulent ones, using just accuracy wasn't enough. So, these other measures were used instead: • Precision: Shows how many of the transactions marked as fraud really were fraud.

• Recall: Shows how many of the real fraud cases werecaught.

• F1-Score: Balances precision and recall, making itgood for situations where there are unequal numbers of classes.

• ROC-AUC Score: Helps see how well the model cantell the difference between normal and fraud transactions at different settings.

• Confusion Matrix: Shows the different types of correctand incorrect predictions visually.

All of these helped give a full picture of how well the model worked in different ways.

C. Model Performance Comparison

Models were evaluated on the test set using the above metrics. Example summary: Random Forest offered a compelling balance between performance and simplicity, with strong recall and AUC values.

TABLE II MODEL PERFORMANCE COMPARISON

Model	Precision	Recall	F1-Score	AUC-ROC
Logistic Regression	0.62	0.48	0.54	0.71
Decision Tree	0.75	0.65	0.70	0.80
Random Forest	**0.83**	**0.74**	**0.78**	**0.87**
XGBoost	0.84	0.73	0.78	0.88

D. Final Model Description

The final model used in this study is a RandomForestClassifier from the scikit-learn library. This algorithm was selected because it is strong, easy to understand, and works well with different types of table data. The model was set up with 100 trees (n_estimators = 100), which gives a good balance between how accurate it is and how fast it runs. The maximum depth of each tree wasn't limited, allowing them to grow as needed, while the minimum number of samples needed to split a node was set to 2, which helps split the data more finely. To deal with the imbalance in the data, where some types of transactions are less common, the class_weight parameter was set to balanced, so that less common transactions, like fraudulent ones, are given more attention during training. The model trained quickly, making it easy to update often or use in real-time systems. When making predictions, it was very fast, taking less than a millisecond, which is perfect for real-time fraud detection.

Looking at which features were most important, Transaction Amount, Account Age, and Device Used were the top predictors for spotting fraud. Features like Product Category and Customer Age were somewhat helpful, but not as much. Time-related features like Month, Day, and Transaction Hour had a smaller impact. This shows the model can pick up on key details about transactions and user behavior, which are important for telling the difference between real and fake transactions.

5. SYSTEM ARCHITECTURE

A. High Level Architecture

Fig. 1.Architecture Diagram

The architecture shown in Fig.1 shows the whole process of the real-time fraud detection system. The system has three main parts — the User Interface, the Input Processor, and the Machine Learning Model — all connected together to provide quick fraud predictions.

The User Interface is made using Streamlit and acts as the front part of the system where users can interact with it. It has a simple form where users can enter details about a transaction, like the amount, payment method, how long the account has been open, and the type of device used. Once the user submits this information, it goes to the Input Processor. This part of the system is in charge of preparing and organizing the input data into a format that the machine learning model can use. It includes converting categories into numbers, checking if numerical values are within the right range, and making sure the data matches the format used during training.

The processed data is then sent to the Machine Learning Model. This model has already been trained and saved using Python's Pickle module so it can be loaded quickly and efficiently. The model does a real-time check on each transaction, deciding if it is fraudulent or not. The result of this check is sent back to the User Interface and shown in an easy-to-read way, often with visual signs and simple explanations.

This setup with separate parts makes it easier for the front-end and back-end to work together smoothly, allowing the system to detect fraud quickly, handle more transactions, and be simple to set up and use.

B. Data Flow Diagram

Fig. 2.Data Flow Diagram

The diagram in Fig.2 shows how data moves through the fraud detection system, step by step, from when a user interacts with it to when the final prediction is shown. The process starts with User Input, where details like transaction amount, account age, payment method, and device type are entered using the Streamlit interface.

Then, the data goes through the Preprocessing and Encoding stage. In this step, categories are turned into numbers, missing data is dealt with, and continuous features are adjusted to fit the format the model was trained on. This makes sure all the input data follows the same structure the machine learning model expects.

6. COMPONENTS OVERVIEW

The system's design includes three main partsthe Backend Machine Learning Model, the Input Layer which is a Streamlit form, and the Frontend Interface which is a Streamlit user interfacealong with important deployment and styling files that help the system work smoothly and stay easy to maintain.

A. Backend: Machine Learning Model

The system's backend uses a supervised machine learning classifier that has been trained on past transaction data to tell the difference between fraudulent and normal transactions. Once the model is trained, it is saved using Python's Pickle module as model.pkl, making it simple to load and use when the system is running.

When a user enters transaction details through the interface, the backend takes that information and organizes it into a Pandas DataFrame. It then applies the needed steps to prepare the data and passes it to the model's predict() function. The model then processes the input and gives a result that is either 1 (meaning the transaction is fraudulent) or 0 (meaning it's genuine). This setup allows the system to respond quickly and makes it easy to update or replace the model with new data as it becomes available.

B. Input Layer: Streamlit Form

The Streamlit form acts as the connection between the user and the machine learning model. It gathers important details about each transaction, including the Transaction Amount, Customer Age, Account Age (in days), Transaction Time (broken down into Hour, Day, and Month), Payment Method, Product Category, and Device Type.

Features that are categorical, such as Payment Method or Product Category, are converted into a format that matches the one used when the model was trained. This ensures that the data entered by the user fits the structure the model expects. This layer makes it easier for users to interact with the system and also helps in keeping the input process consistent, reducing the chances of errors or mismatched data formats when the model makes predictions.

C. Frontend Interface: Streamlit UI

The frontend interface is built with Streamlit and offers an interactive and easy-to-use environment for predicting fraud. Users can input transaction details and start the model prediction by clicking the "Predict Fraud Status" button. The result appears right away in the same

interface, clearly marked as either Fraudulent or Genuine, making it simple for even non-technical users to understand.

To improve the user experience, the interface includes several informative sections. One part explains each input field, helping users understand why certain details are important. Another section provides helpful tips on how to keep transactions secure. The interface also looks better and more professional thanks to custom CSS and Markdown styling, giving it a modern and polished appearance similar to typical web applications.

D. Deployment Model

The prototype is currently running locally by using the command 'streamlit run app.py'. When the application starts, it loads the saved model file ('model.pkl') using 'pickle.load()', which prepares it to accept user inputs and deliver predictions in real time. Because the model processes each input individually instead of handling multiple requests at once, it ensures quick response times with very low delay.

Looking ahead, the system is built with scalability in mind. The whole application can be wrapped into a Docker container, making it easy to deploy consistently across different environments. The backend can be enhanced with either Flask or FastAPI to provide a REST API interface, making it possible to connect the prediction service with other systems. Additionally, the frontend can be hosted on cloud platforms like Streamlit Cloud, Heroku, or AWS, which allows the service to be accessed globally and easily scaled as the number of transactions grows.

E. System Characteristics

The proposed fraud detection system has several key features that make it both practical and easy to expand. It uses a modular structure, with parts like the user interface, model logic, and data preprocessing separated, which allows each part to be updated or fixed without affecting the others.

The system is designed to provide quick results, which is important for making decisions in real time within financial systems. It is also flexible, able to work with realtime data sources such as Apache Kafka or MQTT, and can be scaled up to handle large operations using cloud-based setups.

In terms of security, the system includes ways to check and validate incoming data to stop faulty or harmful requests. Future improvements might involve adding user login and permission controls to boost security further. These features together make the system reliable, fast, and able to handle the demands of today's fraud detection needs.

7. MODEL DEPLOYEMENT

A. Model Serialization and Backend Architecture

To enable real-time inference without retraining, the trained machine learning model was serialized using Python's built-in Pickle module. This allows the model to be saved along with all of its learned parameters and structure into a file named model.pkl, which can be reloaded during runtime. The serialization ensures a lightweight deployment and efficient model utilization without the need for access to the training dataset.

The backend architecture, implemented within the app.py file, manages the entire inference workflow. When the application starts, the serialized model is loaded using the pickle.load() function. Once user inputs are received from the Streamlit interface, the backend processes these inputs by encoding categorical values to match the format used during training and organizing all input features into a structured pandas DataFrame. The preprocessed input is then passed to the model's predict() method, which produces a binary output—1 for fraudulent transactions and 0 for legitimate ones. The result is dynamically displayed to the user through intuitive labels such as Fraudulent or Genuine, ensuring instant and interpretable feedback. This modular architecture guarantees high responsiveness, low latency, and easy maintainability.

B. Streamlit Integration and Deployment Strategy

The whole system is built into a Streamlit web app, which acts as both the user interface and the place where the model runs. Streamlit makes it easy to create a UI using simple Python code and allows users to interact with the app through widgets that let them input and view data in real time. When someone fills out a form, the data is automatically organized into a table format called a DataFrame and sent to the model for predictions. The results are shown right on the page using Streamlit tools like st.success() and st.warning(), so the user gets instant updates without needing to refresh the page.

To keep things running smoothly, basic checks are done on the input data. Streamlit widgets like number_input()

and selectbox() help ensure that users enter valid data, like numbers within a certain range or correct choices from a list. In the future, more advanced checks, such as checking data against a set of rules or spotting unusual patterns, will be added.

Right now, the app runs locally by using the command streamlit run app.py, which starts the app on a web page at http://localhost:8501. For use in a real-world setting, there are several ways to deploy the system. It can be put online with Streamlit Cloud, which needs little setup. It can also be packaged into a Docker container for easier use across different computers or environments. Another option is to host it on cloud services like AWS EC2, Google App Engine, or Heroku. Additionally, the back end can be made into a RESTful API using Flask or FastAPI, making it possible to connect the app with bigger financial systems and data streams in real time.

8. WEB APPLICATION AND USER INTERFACE

A. Streamlit Framework and Interface Design

The web application is built using Streamlit, an opensource Python framework that makes it easy to create interactive machine learning interfaces without needing to know HTML, CSS, or JavaScript. Streamlit handles all the user interaction part of the system, letting users enter transaction details through a web browser and get instant fraud predictions.

The interface has a simple and user-friendly design. The inputs are arranged in a logical way using a three-column layout to make things easier to read. The results of the predictions are shown clearly with emojis and colored indicators, such as Genuine and Fraudulent. The interface also works well on different screen sizes, so it looks good on phones, tablets, and computers.

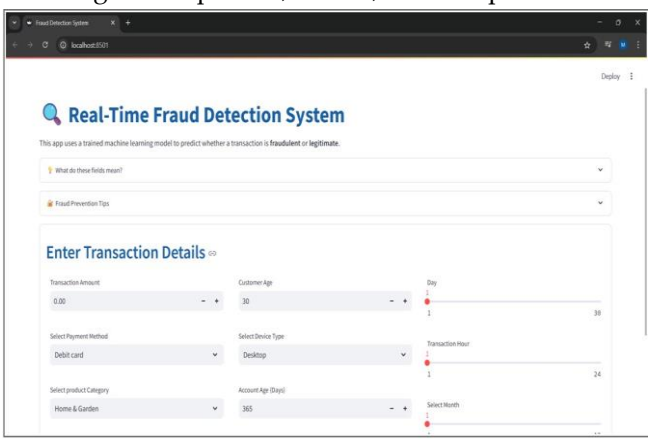

Fig. 3.User Interface

To make the app look better, a custom style.css file is used, which changes the fonts, spacing, and appearance of the elements to give a clean and professional look.

User information is gathered through Streamlit's form elements. Numerical details like transaction amount, customer age, and account age are entered using st.number_input(). Categorical options such as payment method, product category, and device type are picked using st.selectbox(). Time-related features like hour, day, and month are selected with st.slider(). All the inputs are checked and converted into the same format used during training before being sent for prediction.

When the "Predict Fraud Status" button is pressed, Streamlit collects all the input data, puts it into a single row of a table, and sends it to the preloaded model. The prediction result is then shown right away using special UI elements like st.success() or st.warning(), which makes the experience smooth and interactive. Extra sections, such as explanations of the features and tips for preventing fraud, help users understand how the system works and make better decisions.

B. User Guide and Operational Workflow

The prediction process is simple to follow. The user enters the transaction amount, selects the payment method and product category, specifies the device type, inputs customer and account age, and chooses the transaction timestamp. Once the form is submitted, the system processes the input and classifies the transaction. A "Genuine" output means the transaction is low risk, while a "Fraudulent" output suggests suspicious activity. These labels are shown with color-coded visuals for easy understanding.

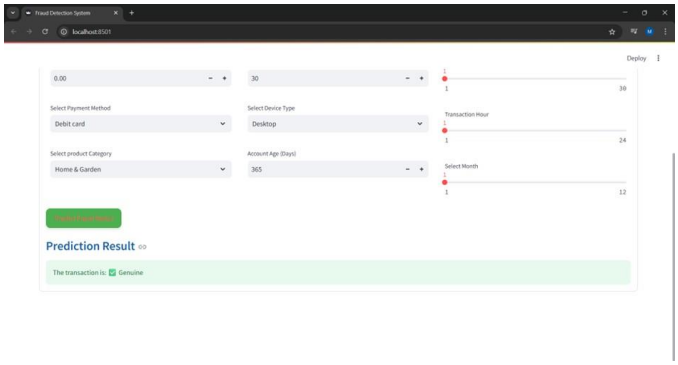

Fig. 4.Prediction Output

Users can explore expandable sections like "What do these fields mean?" to get detailed explanations of each feature and "Fraud Prevention Tips" for useful security advice, including multi-factor authentication,

monitoring for unusual login behavior, and setting transaction limits. Examples of input and output are also provided, showing how the model's predictions differ between low-risk and high-risk situations.

Troubleshooting instructions help users resolve common problems such as missing dependencies, incorrect Python versions, or the absence of the model.pkl file. With this thorough guidance, both technical and non-technical users can easily use, understand, and interpret the fraud detection system.

9. CONCLUSIONS

The Real-Time Fraud Detection System created in this project effectively shows how machine learning can be used to tackle one of the biggest challenges in finance and online shopping—spotting fraud before it causes harm. Using a well-organized set of transaction data,

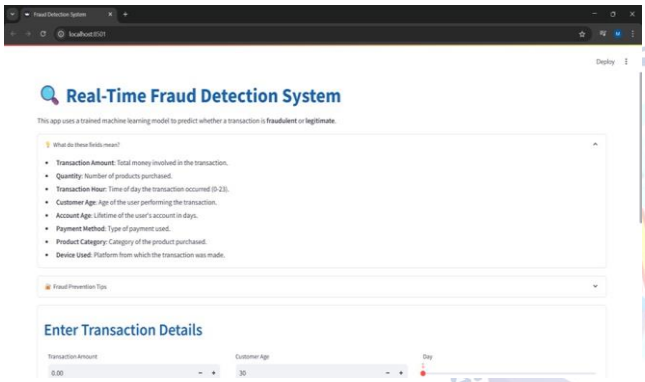

Fig. 5.Explanation Panel

the system uses supervised learning methods, especially Random Forest, to determine if a transaction is fraudulent or safe, and it performs very well. The project includes a full process from collecting and preparing data, creating useful features, training and testing the model, to putting it into use through a web app made with Streamlit. This highlights the strength of applying machine learning in real situations and also makes it easy for people without technical skills to use the model live. A big success of this system is that it keeps a good balance between how well it works and how easy it is to understand. Important factors like the amount of the transaction, how long the account has been open, and the type of device used were found to have a big impact on identifying fraud. By focusing on these key elements and adding clear visual feedback in the user interface, the system builds trust and makes its decisions more transparent. Even though there are some limitations, like having too many nonfraudulent transactions and not having live data, the current version is very flexible and can be expanded. Future improvements, including real-time data flow, tools to explain the model's decisions like SHAP and LIME, an API setup, and cloud-based deployment, will make the system even better and ready for real-world use. This project offers more than just a technical answer—it also provides a guide for responsibly using AI in fraud detection. It shows best practices in handling data, testing models, integrating them with user interfaces, and being aware of security issues, which can help future projects in similar areas. In the end, this system shows how machine learning, when carefully designed and used responsibly, can greatly cut down on fraud and improve trust in digital financial systems.

APPENDIX

The appendix offers a comprehensive look at the core implementation and usability aspects of the fraud detection system. The backend model, built using a supervised machine learning classifier, was saved using Python's pickle module, allowing for quick loading and real-time predictions without the need for retraining. The system workflow involves loading the model at runtime, preprocessing and encoding user inputs, organizing them into a Pandas DataFrame, and applying the model's predict() function to generate results. Streamlit serves as the foundation for the user interaction layer, offering interactive input fields, responsive design elements, and visually clear prediction outputs. The interface features numerical fields, categorical options, sliders for time-related inputs, and styled results using custom CSS for better readability. Additional UI components such as feature explanations and fraud prevention suggestions help users grasp the system's functionality and ensure transparency in its application. In addition, the appendix outlines key instructions for running and engaging with the system. The application requires Python 3.7 or newer and standard data science libraries, which can be installed via a requirements.txt file. The system is executed locally by running the command streamlit run app.py, which opens a browser-based interface.

Conflict of interest statement

Authors declare that they do not have any conflict of interest.

REFERENCES

[1] Bettinger, A., "FINTECH: A Series of 40 Time Shared Models Used at Manufacturers Hanover Trust Company," Interfacec, vol. 2, pp. 62–63, 1972.

[2] Thakor, A. V., "Fintech and Banking: What Do We Know?," Journal of Financial Intermediation, vol. 41, 2020.

[3] Arner, D. W., Barberis, J., and Buckley, R. P., "The Evolution of FinTech: A New Post-Crisis Paradigm?," Georgetown Journal of International Law, vol. 47, pp. 1271–1319, 2016. Available:Fraud Detection," Proc. 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, UAE, pp. 1–6, 2019.

[4] PwC, "PwC's Global Economic Crime and Fraud Survey 2020." Available: https://www.pwc.com/fraudsurvey. [Accessed: 30-Nov-2020].

[5] ACFE, "2020ACFE Report to the Nations." Available: https://www.acfe.com/report-to-the-nations/2020/. [Accessed: 11-Nov-2020].

[6] Investopedia, "Fraud Definition." Available: https://www.investopedia.com/terms/f/fraud.asp. [Accessed: 15-Dec-2020].

[7] Chalapathy, R., and Chawla, S., "Deep Learning for Anomaly Detection: A Survey," arXiv:1901.03407, 2019.

[8] Zimek, A., and Schubert, E., "Outlier Detection," Encyclopedia of Database Systems, Springer, pp. 1–5, 2017.

[9] Kaggle, "Credit Card Fraud Detection Dataset." Available: https://www.kaggle.com/mlg-ulb/creditcardfraud. [Accessed: 30-Nov-2020].

[10] Kaggle, "Bank Transaction Data." Available: https://www.kaggle.com/apoorvwatsky/bank-transaction-data. [Accessed: 30-Nov-2020].

[11] Kaggle, "Bitcoin Blockchain Historical Data." Available: https://www.kaggle.com/bigquery/bitcoin-blockchain. [Accessed: 30-Nov-2020].

[12] UCI ML Repository, "Machine Learning Repository." Available: https://archive.ics.uci.edu/ml/index.php. [Accessed: 11-Nov-2020].

[13] Kaggle, "Synthetic Data from a Financial Payment System (BankSim)." Available: https://www.kaggle.com/ntnutestimon/banksim1. [Accessed: 30-Nov-2020].

[14] Lopez-Rojas, E. A., Elmir, A., and Axelsson, S., "PaySim: A Financial Mobile Money Simulator for Fraud Detection," Proc. 28th European Modeling and Simulation Symposium (EMSS), Larnaca, Cyprus, pp. 26–28, 2016.

[15] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G., "Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information," Proc. 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, pp. 1–8, 2015.

[16] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., and Bontempi, G., "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, pp. 3784–3797, 2017.

[17] Ma, T., Qian, S., Cao, J., Xue, G., Yu, J., Zhu, Y., and Li, M., "An Unsupervised Incremental Virtual Learning Method for Financial

[18] Somasundaram, A., and Reddy, S., "Parallel and Incremental Credit Card Fraud Detection Model to Handle Concept Drift and Data Imbalance," Neural Computing and Applications, vol. 31, pp. 3–14, 2019.

[19] Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., and Sun, X., "The Application of Data Mining Techniques in Financial Fraud Detection: A Classification Framework and Review," Decision Support Systems, vol. 50, pp. 559–569, 2011.

[20] Ahmed, M., Mahmood, A. N., and Islam, M. R., "A Survey of Anomaly Detection Techniques in the Financial Domain," Future Generation Computer Systems, vol. 55, pp. 278–288, 2016.

[21] Ahmed, M., Choudhury, N., and Uddin, S., "Anomaly Detection on Big Data in Financial Markets," Proc. IEEE/ACM ASONAM, Sydney, Australia, pp. 998–1001, 2017.

[22] Abdallah, A., Aizaini, M., and Zainal, M. A., "Fraud Detection System: A Survey," Journal of Network and Computer Applications, vol. 68, pp. 90–113, 2016.

[23] Gai, K., Qiu, M., and Sun, X., "A Survey on FinTech," Journal of Network and Computer Applications, vol. 103, pp. 262–273, 2018.

[24] Ryman-Tubb, N. F., Krause, P. J., and Garn, W., "How Artificial Intelligence and Machine Learning Research Impacts Payment Card Fraud Detection: A Survey," Engineering Applications of Artificial Intelligence, vol. 76, pp. 130–157, 2018.

[25] West, J., and Bhattacharya, M., "Intelligent Financial Fraud Detection: A Comprehensive Review," Computers & Security, vol. 57, pp.47–66, 2016.