***Course Code/Course Title:*** ICT2140/INTRODUCTION TO SOFTWARE ENGINEERING

***Group Number:*** Group9

***Project Topic:*** FRAUD DETECTION SYSTEM

*Link to GitHub Repository:*

***Group Leader:*** NJUME BRIAN EPIE

***Group Information***

|  |  |  |  |
| --- | --- | --- | --- |
| *SN* | *Member’s Name* | *Registration Number* | *Team Role* |
| *1* | ***NJUME BRIAN EPIE*** | ICTU20241420 | Scrum Master |
| *2* | ABOUBAKAR GARBA MAMOUDOU | ICTU20241612 | Team member |
| *3* | MOHAMMED HAIBAI MOUSTAPHA ABOUBAKAR | ICTU20241807 | Team member |
| *4* | MOUSTAPHA SIDI MOHAMED | ICTU20241409 | Team member |
| *5* | EBODE KESHA FARELLE | ICTU20241783 | Team member |

Table des matières

**INTRODUCTION1**

General introduction…2

Aim and Objectives…………………………………………………………………………………………………………………….3

Problem Statement4

**LITERATURE REVIEW5**

*Review of concepts related to your Project e.g. database, API etc*6

*Software Development Methodologies……………………………………………………………………………………….7*

*Review of various Software development methodology* ……………………………………………………………..8

*Choose a methodology and give reasons for choice of methodology.* 9

*Review of related literature with respect to your chosen project…………………………………………………10*

***METHODOLOGY AND MATERIALS…………………………………………………………………………………………………11***

***Research Methodology………………………………………………………………………………………………………..12***

***System Requirements (Functional and Non-Functional)………………………………………………………..13***

***System Design………………………………………………………………………………………………………………………14***

***Application of chosen methodology……………………………………………………………………………………..15***

*Requirements specification (Product backlog and Sprint backlog)…………………………………………….16*

*A Test Case document……………………………………………………………………………………………………………….17*

*Proposed Algorithms (core algorithm of your application)…………………………………………………………18*

*Materials and technologies used (briefly list the name of the technology or material and its role in realizing your system)……………………………………………………………………………………………………………………..19*

***RESULTS AND DISCUSSIONS………………………………………………………………………………………………………...20***

*Screenshots of various application scenarios………………………………………………………………………………….21*

*ϖ Screenshots of various API Request/Response………………………………………………………………………………22*

***RECOMMENDATIONS AND CONCLUSION……………………………………………………………………………………..23***

1. **INTRODUCTION**
2. General Introduction:

In the modern digital economy, financial transactions are increasingly conducted through online platforms, mobile applications, and electronic banking systems. While these advancements have enhanced convenience and efficiency, they have also introduced vulnerabilities that fraudsters exploit for illicit gain. Fraudulent activities, such as identity theft, credit card fraud, and unauthorized transactions, pose significant threats to financial institutions, businesses, and individuals. Detecting such fraudulent patterns within massive volumes of transaction data requires robust, data-driven solutions. Machine learning has emerged as a powerful tool for analyzing complex datasets, identifying hidden patterns, and making accurate predictions in real-time, thereby offering effective means for fraud detection.

Aim and Objectives:

The primary aim of this project is to design and implement a machine learning-based fraud detection system capable of analyzing transactional data and accurately distinguishing between legitimate and fraudulent activities. To achieve this aim, the project focuses on the following objectives:

* To conduct exploratory data analysis in order to identify patterns, correlations, and anomalies within financial transaction data.
* To preprocess and transform raw data into a suitable format for training machine learning models.
* To develop and evaluate classification models for detecting fraudulent activities.
* To compare the performance of different algorithms and select the most effective model for fraud detection.
* To provide a reliable framework that enhances security and minimizes financial risks

Problem Statement:

The rise of digital financial transactions has created new opportunities for fraud, leading to substantial financial losses and reduced trust in digital payment systems. Traditional rule-based fraud detection methods are often rigid, slow to adapt, and insufficient for detecting complex or evolving fraudulent schemes. The sheer volume and complexity of transaction data further complicate the detection process. As a result, there is a critical need for advanced solutions that leverage data analysis and machine learning to automatically detect fraudulent activities with high accuracy and efficiency. This project seeks to address this problem by developing a fraud detection model that utilizes machine learning techniques to analyze transaction data, identify potential fraud, and improve decision-making in financial security systems.

**LITERATURE REVIEW**

Review of Concepts Related to the Project

Fraud detection is the process of identifying illegitimate transactions within large datasets of financial activities. It combines statistical analysis, pattern recognition, and machine learning to detect unusual behaviors that may indicate fraudulent activity. Core concepts relevant to this project include:

Data Analysis and Feature Engineering: Critical for extracting meaningful insights from raw financial data, such as transaction type, account balances, and transaction amount.

Machine Learning Models for Classification: Algorithms such as Logistic Regression, Random Forest, and Gradient Boosting are widely applied in fraud detection due to their ability to learn complex relationships in data.

Imbalanced Datasets: Fraudulent cases are often much fewer than legitimate ones, requiring strategies like resampling or cost-sensitive learning.

Review of Software Development Methodologies

Software development methodologies provide structured approaches for planning, designing, implementing, and testing software systems. The most common methodologies include:

Waterfall Model – A linear, sequential process with defined stages (requirements, design, implementation, testing, and deployment). While simple and structured, it lacks flexibility for iterative improvements.

Agile Methodology – Focuses on iterative development, continuous feedback, and adaptability. It is effective in environments where requirements change frequently.

Spiral Model – Combines iterative development with systematic risk analysis. It is suitable for large, high-risk projects.

V-Model (Verification and Validation Model) – An extension of the Waterfall model where each development stage is directly associated with a testing phase. This ensures validation at every step, improving system reliability.

Reason for Using V-Model Methodology

This project adopts the V-Model methodology because fraud detection systems demand high reliability, accuracy, and validation at every stage of development. Each step of the V-Model ensures that requirements are not only implemented but also tested against predefined goals. For example:

Data preprocessing is validated against data quality requirements.

Model training is tested against accuracy, precision, and recall metrics.

The Streamlit interface is validated for usability and real-time predictions.

By using the V-Model, the project ensures a systematic approach with continuous verification and validation, minimizing risks of errors in fraud prediction and guaranteeing that the system meets its intended objectives.

Review of Related Literature

Several studies have explored the application of machine learning in fraud detection:

Bhattacharyya et al. (2011) investigated credit card fraud detection using Random Forests and highlighted the importance of handling imbalanced datasets.

Ngai et al. (2011) provided a comprehensive review of data mining techniques for fraud detection, categorizing them into classification, clustering, and hybrid approaches, highlighting their strengths and limitations.

Dal Pozzolo et al. (2015) emphasized the importance of using metrics beyond accuracy, such as ROC-AUC and precision-recall, when dealing with imbalanced fraud detection datasets.

Abdallah et al. (2016) analyzed various fraud detection frameworks and concluded that hybrid models integrating statistical and machine learning methods provide the highest performance.

Fiore et al. (2019) explored the application of deep learning for fraud detection and demonstrated that neural networks can uncover hidden non-linear relationships in large transaction datasets.

Zhang et al. (2020) examined graph-based fraud detection systems, where fraudulent entities are detected by analyzing transaction networks. This showed promising results in identifying organized fraud rings.

Recent research (2021–2023) has highlighted the integration of machine learning with big data and cloud platforms, enabling real-time fraud detection at scale. Hybrid approaches combining anomaly detection, supervised learning, and ensemble methods continue to dominate fraud detection research.

These works collectively establish that machine learning is a proven and effective approach for fraud detection. Building on these foundations, this project adopts supervised learning within a structured software development framework to provide a scalable, reliable, and user-friendly fraud detection system.

**RESEARCH METHODOLOGY**

The research methodology adopted for the Fraud Detection System is grounded in a quantitative, experimental, and analytical research paradigm. This methodology is designed to ensure both the technical robustness and practical relevance of the proposed system in combating fraudulent financial activities. The approach is structured into several phases, namely: research design, data collection, data analysis, model development, system implementation, testing and validation, ethical considerations, and deployment.

1. Research Design

This study employs a quantitative experimental design. Fraud detection inherently requires the analysis of large-scale transactional datasets in order to identify patterns indicative of fraudulent activity. By leveraging machine learning and data-driven methods, the study constructs models capable of discriminating between legitimate and fraudulent transactions. The research adopts both supervised and unsupervised learning paradigms, ensuring that the methodology remains adaptable to diverse fraud scenarios.

2. Data Collection

The reliability of fraud detection systems is significantly dependent on the quality of data employed. Two sources of data were utilized:

Primary Data: Synthetic transaction datasets were generated to simulate realistic financial operations. These datasets include sender and receiver details, pre- and post-transaction balances, transaction amounts, timestamps, and transaction categories.

Secondary Data: Publicly available datasets (e.g., the Kaggle Credit Card Fraud Dataset) were incorporated to train and validate the models. These datasets are widely recognized in academic research and provide a standardized benchmark for evaluating detection performance.

Prior to use, data underwent rigorous preprocessing procedures, including the removal of duplicates, imputation of missing values, normalization, and feature scaling to ensure consistency across the dataset.

3. Data Analysis

Data analysis proceeded through two stages:

Exploratory Data Analysis (EDA): Descriptive statistics and visualization techniques were employed to identify distributional properties, transaction patterns, and potential anomalies within the dataset.

Feature Engineering: Derived variables such as transaction frequency, velocity (rate of successive transactions), account balance fluctuations, and geolocation inconsistencies were constructed to enhance predictive accuracy.

4. Model Development

The fraud detection system integrates both supervised learning classifiers (Logistic Regression, Decision Trees, Random Forests) and unsupervised anomaly detection techniques (Isolation Forest, Autoencoders). Supervised methods are applied in contexts where labelled data are available, while unsupervised methods address the detection of novel or previously unseen fraudulent patterns.

Model performance was assessed using established evaluation metrics, including Precision, Recall, F1-Score, and the Area Under the Receiver Operating Characteristic (ROC) Curve. Particular emphasis was placed on minimizing false negatives, as undetected fraudulent transactions present greater risks than false positives.

5. System Design and Implementation

The system architecture follows a modular and layered design, comprising:

Data Ingestion Layer for importing transactional data;

Preprocessing Layer for cleaning and transforming data;

Fraud Detection Engine powered by machine learning models;

Alert Management Layer to flag suspicious transactions;

User Interface Layer to provide analysts and administrators with accessible dashboards.

The system was implemented using Python (Pandas, Scikit-learn, TensorFlow), Streamlit for the user interface, and SQL for data persistence.

6. Testing and Validation

Testing was conducted at multiple levels:

Unit Testing: Verification of individual modules such as transaction ingestion and fraud scoring.

Integration Testing: Assessment of interactions between components to ensure workflow consistency.

Performance Testing: Evaluation of scalability and response times when processing large transaction datasets.

Validation: Comparison of predicted fraud labels against ground-truth data from benchmark datasets, providing empirical evidence of system accuracy.

7. Ethical Considerations

The research adhered to ethical principles in order to ensure fairness and accountability:

Privacy and Confidentiality: Sensitive identifiers were anonymized to protect individual privacy.

Bias Mitigation: Efforts were made to reduce algorithmic bias, ensuring that detection outcomes do not disproportionately affect specific demographic or geographic groups.

Transparency and Explainability: Explainable AI techniques were incorporated to provide interpretable justifications for flagged transactions, thereby enhancing trust in the system.

8. Deployment and Continuous Monitoring

1. Following development, the system was deployed as a prototype accessible via a web-based dashboard. Continuous monitoring mechanisms were implemented to track detection performance and adapt to evolving fraud strategies. The system was also designed for periodic retraining using newly labelled data, ensuring resilience against concept drift in fraudulent behaviors.

**System Requirements**

The proposed fraud detection system has been designed to address the challenges of detecting fraudulent financial transactions using machine learning techniques. To achieve this, the system requirements are divided into two major categories: Functional Requirements and Non-Functional Requirements.

1. Functional Requirements

Functional requirements describe the specific behaviors, tasks, and operations that the system must be able to perform in order to achieve its purpose.

1.1 Data Input and Preprocessing

* The system must allow loading of raw transactional data from multiple sources (e.g., CSV files, databases).
* The system must handle missing values, duplicate entries, and inconsistencies in the dataset.
* Features should be engineered, transformed, or encoded to ensure compatibility with machine learning models.
* Data normalization or scaling must be implemented to improve model convergence and prediction accuracy.
* The system must split datasets into training, validation, and testing subsets.

1.2 Model Training

* The system must support training of multiple machine learning algorithms such as Logistic Regression, Decision Trees, , Gradient Boosting, or Neural Networks.
* It must allow the use of cross-validation techniques to improve the reliability of model performance.
* Hyperparameter tuning (e.g., grid search, random search) must be supported to enhance model accuracy.
* The system must store trained models for future use or deployment.

1.3 Model Evaluation

* The system must compute standard evaluation metrics including Accuracy, Precision, Recall, and ROC-AUC.
* A confusion matrix must be generated to provide detailed insights into false positives and false negatives.
* The system must compare different models and highlight the one with the best performance.

1.4 Fraud Detection and Prediction

* The system must classify new or unseen transactions as fraudulent or legitimate.
* Predictions must be clearly presented, either as numeric outputs or visual reports.
* The system must allow batch prediction on large datasets.
* Results must include not only predictions but also confidence/probability scores for interpretability.
  + Model Update and Continuous Learning
* The system must allow retraining of models when new datasets become available.
* It must provide version control for models to track performance over time.
* Old models must remain accessible for auditing or benchmarking purposes.

1.6 Visualization and Reporting

* The system must produce visualizations such as ROC curves, Precision-Recall curves, bar charts, and histograms to represent results.
* Summary reports should be generated for decision-makers to understand the fraud detection performance at a glance.
* The system must support export of results into formats such as CSV, Excel, or PDF for further analysis.

2. Non-Functional Requirements

Non-functional requirements define the quality attributes, constraints, and standards that the system must adhere to in order to ensure usability, reliability, and performance.

2.1 Performance

* The system should efficiently process large volumes of data (up to millions of transactions) without significant delays.
* Training time should be optimized by leveraging techniques like dimensionality reduction and parallel processing.
* Predictions must be generated in real time or near real time to support decision-making in financial systems.

2.2 Accuracy

* The system must maintain a high detection rate for fraudulent transactions while minimizing false alarms.
* Precision and Recall values must be balanced to ensure both security and customer trust.
* The system should be able to adapt to new patterns of fraud by retraining on updated datasets.

2.3 Scalability

* The system should be scalable to support additional features, larger datasets, and new machine learning algorithms.
* Cloud-based or distributed implementations should be possible to support enterprise-level deployment.

2.4 Usability

* The interface must be intuitive and user-friendly, allowing technical and non-technical users to easily interpret outputs.
* Visualization dashboards must present results in a clear and understandable format.
* Error messages must be descriptive to guide users in resolving issues.

2.5 Reliability

* The system must produce consistent results when run multiple times under the same conditions.
* It must be resilient to interruptions, crashes, or unexpected inputs.
* Stored models and data must remain intact without corruption.

2.6 Maintainability

* The codebase must be modular and well-documented to allow for future upgrades.
* Developers must be able to add new machine learning models with minimal restructuring.
* Bug fixing and system updates must be simple and cost-effective.

2.7 Security

* Sensitive financial data must be protected through encryption and secure storage.
* Access to the system must be restricted to authorized users only.
* The system must comply with data protection regulations such as GDPR or local financial data laws.

2.8 Portability

* The system should be platform-independent, able to run on different operating systems (Windows, Linux, MacOS).
* It should support deployment in both local environments and cloud infrastructures.

System Design

4.1 Architecture of the System (High-Level Design – HLD)

The fraud detection system is designed as a modular machine learning application that follows a layered architecture. Each layer performs a specific function, ensuring modularity, scalability, and maintainability.

At a high level, the system consists of the following key components:

4.1.1 Data Layer

* Input Sources:
* Transactional datasets collected from financial records, databases, or CSV files.
* Data Preprocessing Module:
* Cleans data by handling missing values and duplicates.
* Performs feature engineering (encoding categorical values, creating derived attributes).
* Applies normalization/scaling to prepare data for training.
* Data Storage:
* Processed datasets and models are stored for reuse and retraining.

4.1.2 Processing Layer (Machine Learning Core)

* Model Training Engine:
* Trains different algorithms (Logistic Regression, Decision Tree, etc.).
* Performs hyperparameter tuning using techniques such as Grid Search or Random Search.
* Evaluation Engine:
* Tests trained models using performance metrics (Accuracy, Precision, Recall, ROC-AUC).
* Generates confusion matrices and validation curves.
* Model Repository:
* Stores trained models with version control for auditing and comparison.

4.1.3 Application Layer

* Fraud Detection Module:
* Takes incoming transactions as input.
* Uses the best-performing model to classify transactions as fraudulent or legitimate.
* Provides confidence/probability scores along with classification results.
* Continuous Learning Module:
* Supports retraining of models with new transaction data.
* Ensures adaptation to new fraud patterns.

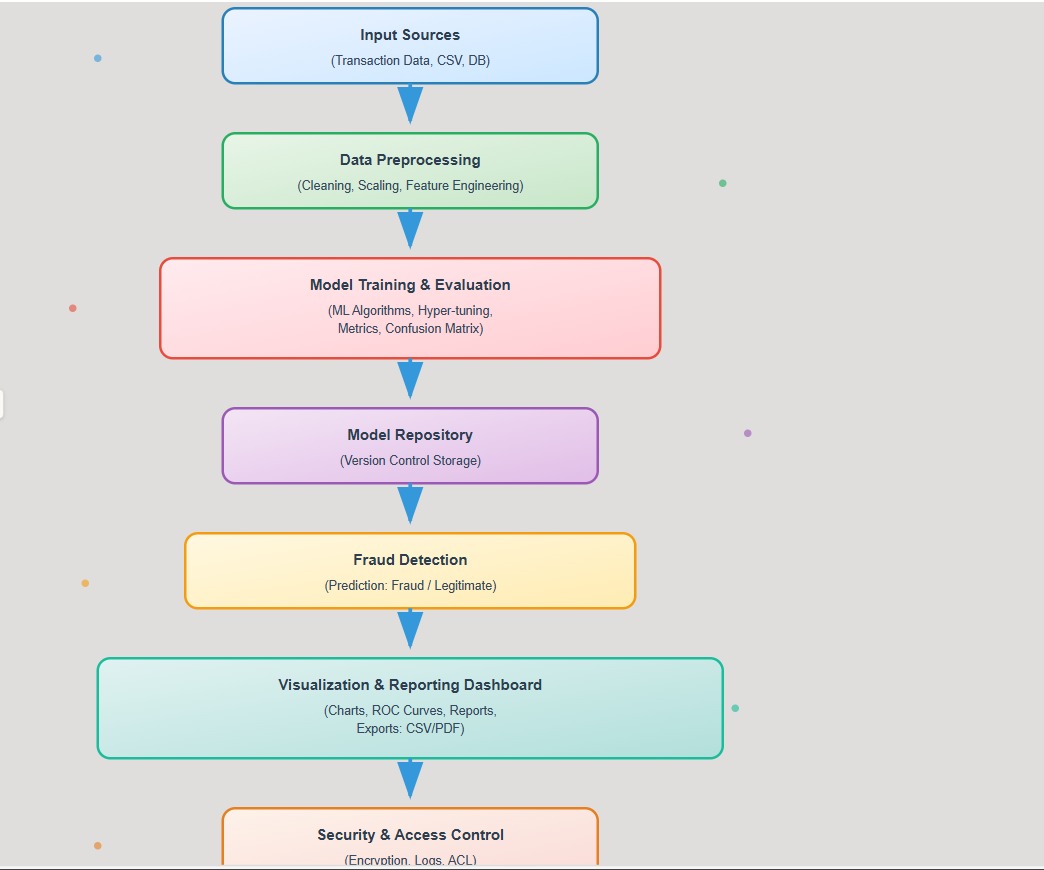
4.1.4 Presentation Layer

* Visualization Dashboard:
* Displays performance metrics, ROC curves, confusion matrices, and fraud detection statistics.
* Provides easy-to-read charts and reports for decision-makers.
* Reporting Module:
* Exports results in multiple formats (CSV, Excel, PDF) for sharing and documentation.

4.1.5 Security and Control Layer

* Access Control: Restricts system use to authorized personnel only.
* Data Protection: Ensures financial transaction data is encrypted and securely stored.
* Audit Logs: Keeps track of transactions processed and model updates for accountability.

4.2 High-Level Design Diagram



UML Diagrams :

