SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

TAMIL NADU, COIMBATORE

Department of Artificial Intelligence and Machine Learning

INTERNSHIP REPORT
AT Unified Mentor

Report Submitted By

Name : **Hareesh kumar K**

Class : **III Year B.Tech., AIML**

Academic Year : **2025 – 2026**

Duration : **15.05.2025 – 15.08.2025**

ACKNOWLEDGEMENT

I am deeply honored to express my heartfelt gratitude to **Unified Mentor** for granting me the opportunity to undertake this enriching internship. The organization provided an exceptional platform that not only enhanced my technical and professional skills but also fostered an environment of learning, collaboration, and innovation. This internship has been a pivotal milestone in my career journey, and I am immensely grateful for the trust and support extended to me throughout this period.

I extend my sincere appreciation to my supervisor whose exemplary guidance, insightful feedback, and unwavering encouragement were instrumental in shaping the outcome of my projects. Their expertise, patience, and commitment to nurturing my skills helped me navigate challenges and achieve the objectives of the internship with confidence and clarity.

I am equally thankful to my colleagues and team members at Unified Mentor. Their collaborative spirit, willingness to share knowledge, and constant support created a dynamic and inspiring work environment. The synergy within the team played a crucial role in the successful completion of all four of my internship projects, and I am grateful for their camaraderie and contributions.

I would also like to acknowledge the contributions of any external mentors, advisors, or institutional collaborators who provided valuable insights and expertise during the course of this internship. Their input enriched my understanding and added depth to my work.

Lastly, I owe a special debt of gratitude to my friends and family, whose unconditional support, encouragement, and belief in my abilities kept me motivated throughout this journey. Their presence provided me with the emotional strength to overcome challenges and stay focused on my goals.

This internship experience has been transformative, and I carry forward the lessons, skills, and connections gained at Unified Mentor with immense gratitude and pride.

INTERNSHIP DETAILS

Name of the Industry	**:**	Unified Mentor
Address	**:**	Cyber City, WeWork DLF Forum, Haryana 122002
Website	**:**	https://www.unifiedmentor.com/
Internship Duration	**:**	3 Months
Contact No	**:**	+91 08645322947
E-mail id	**:**	hello@unifiedmentor.com

INTRODUCTION

Project title : “ Fraud Transaction Detection Using Transactions Dataset “. The rapid growth of digital financial transactions has revolutionized the way individuals and organizations conduct business, offering unprecedented convenience and efficiency. However, this digital transformation has also given rise to sophisticated financial fraud, posing significant risks to financial institutions and their customers. The "Fraud Transaction Detection Using Transactions Dataset" project, undertaken during my internship at Unified Mentor, addresses this critical challenge by developing an advanced artificial intelligence (AI)-based system to detect fraudulent transactions in real time. The primary objective of this project is to create a high-accuracy, scalable, and user-friendly fraud detection system capable of analyzing both individual transactions and large datasets, delivering actionable insights through an interactive web application.

This project leverages a synthetic dataset of 67,000 transactions, carefully designed to simulate realistic fraud scenarios, including high-amount transactions, compromised terminals, and stolen customer credentials. By combining robust data preprocessing, ensemble machine learning techniques, and a Streamlit-based web interface, the system provides a comprehensive solution for identifying and mitigating fraudulent activities. The final deliverables include a trained machine learning model (*ada_model.pkl*) and a fully functional web application (*app.py*) that supports real-time and batch processing with intuitive visualizations. This report provides a detailed account of the project's methodology, technical implementation, results, challenges, learnings, and potential avenues for future improvement, underscoring its significance in enhancing financial security.

Project Overview

The "Fraud Transaction Detection Using Transactions Dataset" project aims to develop a robust AI-driven system that identifies fraudulent transactions with high accuracy and presents results in a user-friendly format. The system is designed to serve financial institutions, payment processors, or e-commerce platforms by enabling real-time fraud detection and batch analysis of transaction data. The project is structured around three core components:

1. Data Preprocessing: Cleaning and transforming the synthetic transaction dataset to ensure compatibility with machine learning algorithms, including handling class

imbalance and standardizing features.

2. Model Development: Building and optimizing an AdaBoostClassifier model to detect fraudulent transactions with high accuracy, recall, and ROC AUC scores, ensuring reliable performance on imbalanced data.
3. Application Development: Creating an interactive Streamlit web application with a custom-designed interface to support single transaction analysis and batch processing, complete with visualizations and downloadable results.

The dataset, comprising 67,000 synthetic transactions, incorporates realistic fraud scenarios to validate the model's effectiveness. The final system achieves exceptional performance metrics and provides a seamless user experience, making it a valuable tool for fraud detection in real-world applications.

PROJECT OVERVIEW

The "Fraud Transaction Detection Using Transactions Dataset" project, undertaken as part of my internship at Unified Mentor, is designed to address the growing challenge of financial fraud in digital transactions by developing a sophisticated artificial intelligence (AI)-based fraud detection system. The primary objective is to create a high-accuracy, scalable, and user-friendly system capable of analyzing financial transactions in real time or in batches, accurately classifying them as fraudulent or safe, and delivering actionable insights through an intuitive interface. By leveraging advanced machine learning techniques and a visually engaging web application, the system aims to empower financial institutions, payment processors, and e-commerce platforms to mitigate fraud effectively, thereby enhancing financial security and trust. The project is built upon a synthetic dataset of 67,000 transactions, meticulously crafted to simulate realistic fraud scenarios inspired by real-world patterns, such as high-amount transactions, compromised terminals, and stolen customer credentials. These scenarios ensure that the system is tested against diverse and challenging fraud patterns, making it robust and applicable to practical use cases. The project is structured around three core components, each addressing a critical aspect of the fraud detection pipeline. Data preprocessing focuses on preparing the raw dataset for machine learning by cleaning, structuring, and transforming the data, including handling class imbalance, standardizing numerical features, and ensuring data integrity, all of which are essential for training a robust model. Model development involves designing, training, and optimizing a machine learning model to detect fraudulent transactions with high accuracy and recall, with the AdaBoostClassifier, an ensemble learning algorithm, selected for its ability to handle imbalanced datasets and deliver reliable performance. Application development includes a Streamlit-based web application to provide an interactive and user-friendly interface for end-users, supporting both single transaction analysis and batch processing, delivering predictions, confidence scores, risk levels, and interactive visualizations to facilitate decision-making. The dataset, stored in processed_fraud_data.csv, comprises 67,000 synthetic transactions, each characterized by eight features: CUSTOMER_ID, TERMINAL_ID, TX_AMOUNT, DAY, MONTH, YEAR, HOUR, and MINUTE, with TX_FRAUD as the binary target variable (1 for fraudulent, 0 for safe). The dataset was designed to reflect real-world fraud patterns through three distinct scenarios, as documented in NoteBook.txt. Transactions exceeding $220 are labeled as fraudulent, simulating cases where unusually large transaction amounts indicate potential fraud, serving as a baseline to validate the model’s ability to detect obvious fraud patterns. Each day, two terminals are randomly selected, and all transactions processed through these terminals over the next 28 days are marked as fraudulent, mimicking real-world scenarios like phishing attacks or compromised point-of-sale systems. Additionally, each day, three customers are randomly chosen, and for the next 14 days, one-third of their transactions have their amounts multiplied by five and are labeled as fraudulent, replicating card-not-present fraud where

fraudsters exploit stolen credentials to make high-value transactions alongside the customer's normal activity. These scenarios ensure that the dataset is both challenging and representative of real-world fraud, providing a robust foundation for model development and evaluation. The project's workflow integrates data preprocessing, model development, and application development into a cohesive system. The raw dataset was cleaned, split into training (70%, 46,900 transactions) and testing (30%, 20,100 transactions) sets using stratified sampling, and processed to address class imbalance using the Synthetic Minority Oversampling Technique (SMOTE), with numerical features standardized to ensure compatibility with the machine learning model, ensuring the dataset was clean, structured, and optimized for training. The AdaBoostClassifier, with a DecisionTreeClassifier as the base estimator, was trained using a pipeline that incorporated SMOTE and feature scaling, with hyperparameter tuning performed using RandomizedSearchCV to optimize performance, resulting in a model with exceptional metrics: 99.83% accuracy, 97.22% recall, and 98.57% ROC AUC on the test set, with the trained model saved as ada_model.pkl for integration with the application. A Streamlit web application (app.py) was developed to provide an intuitive interface for fraud detection, featuring a custom dark theme with gradient animations, metric cards, and interactive visualizations powered by Plotly, supporting two modes: single transaction analysis, where users can input transaction details and receive real-time predictions, confidence scores, risk level classifications (LOW, MEDIUM, HIGH), and a risk factor breakdown visualized as a bar chart, and batch processing, where users can upload a CSV file containing multiple transactions, with the system processing the entire dataset, providing predictions, risk levels, and summary metrics (e.g., fraud count, fraud rate), with visualizations including a histogram of transaction amounts and a pie chart of risk level distribution, and results downloadable as a CSV file. The final deliverables include a trained machine learning model (ada_model.pkl), a high-accuracy AdaBoostClassifier model capable of detecting fraudulent transactions with exceptional performance metrics, suitable for real-time and batch processing, a Streamlit web application (app.py), a fully functional web interface that supports both single transaction analysis and batch processing, delivering predictions, visualizations, and downloadable results in a user-friendly format, and documentation (NoteBook.txt), providing detailed documentation of the fraud simulation logic, ensuring reproducibility and transparency of the dataset generation process. The project's objectives were to develop a machine learning model with high accuracy and recall for fraud detection, create a user-friendly application to make fraud detection accessible to non-technical users, incorporate realistic fraud scenarios to ensure the system's applicability to real-world use cases, and provide actionable insights through visualizations and metrics to support decision-making. The project addresses a critical need in the financial sector by providing a robust and scalable solution for fraud detection, with its high accuracy and recall ensuring reliable identification of fraudulent transactions, minimizing financial losses and enhancing security. The interactive Streamlit application makes the system accessible to a wide range of users, from financial analysts to business stakeholders, by presenting complex model outputs in an intuitive format. The use of a synthetic dataset with realistic fraud scenarios ensures that the

system is well-tested and adaptable to real-world challenges, and the project's modular design and documented processes make it scalable and extensible, paving the way for future enhancements such as real-time data integration or advanced feature engineering. By combining cutting-edge machine learning with a user-centric application, the project delivers a comprehensive fraud detection solution that aligns with industry needs and demonstrates the power of AI in combating financial fraud.

DATA PREPROCESSING

Effective data preprocessing was a cornerstone of the "Fraud Transaction Detection Using Transactions Dataset" project, ensuring that the synthetic dataset was optimally prepared for machine learning. Given the complexity of fraud detection and the imbalanced nature of the dataset, meticulous preprocessing was essential to create a clean, structured, and model-ready dataset. The preprocessing pipeline addressed feature selection, data cleaning, data splitting, class imbalance, and feature scaling, each tailored to enhance the performance of the AdaBoostClassifier model. Below is a detailed breakdown of the preprocessing steps undertaken to ensure the dataset was suitable for modeling:

1. Feature Selection and Retention:
    - Objective: To retain all relevant features that could contribute to detecting fraudulent patterns while ensuring compatibility with the chosen machine learning algorithm.
    - Process: The dataset included eight features: CUSTOMER_ID, TERMINAL_ID, TX_AMOUNT, DAY, MONTH, YEAR, HOUR, and MINUTE. Each feature was evaluated for its relevance to fraud detection:
        - CUSTOMER_ID: A unique identifier for each customer, retained to capture customer-specific fraud patterns, such as those simulated in the customer fraud scenario (where certain customers' transactions are flagged as fraudulent).
        - TERMINAL_ID: A unique identifier for each terminal, kept to identify terminal-based fraud patterns, as outlined in the terminal fraud scenario.
        - TX_AMOUNT: The transaction amount, critical for detecting high-amount fraud (transactions exceeding $220) and customer fraud (transactions with amounts multiplied by five).
        - DAY, MONTH, YEAR, HOUR, MINUTE: Temporal features, essential for capturing time-based patterns, such as transactions occurring at unusual hours (e.g., late at night) or during specific periods (e.g., 28-day

terminal fraud windows or 14-day customer fraud windows).

- Rationale for Retention: All features were retained due to their direct relevance to the fraud scenarios defined in *NoteBook.txt*. No features were dropped, as each provided unique information for distinguishing fraudulent from safe transactions.
- Categorical Feature Handling: CUSTOMER_ID and TERMINAL_ID were kept as integers rather than one-hot encoded, as tree-based models (e.g., DecisionTreeClassifier used as the base estimator in AdaBoost) can effectively handle numerical representations of categorical variables without requiring additional encoding.
- Numerical Feature Handling: Numerical features (TX_AMOUNT, DAY, MONTH, YEAR, HOUR, MINUTE) were initially kept as-is to preserve their raw values, which are meaningful for tree-based models. However, these features were later standardized (see Feature Scaling below) to ensure consistency and improve model performance.

2. Data Cleaning:
   - Objective: To ensure the dataset was free of errors, missing values, duplicates, or inconsistencies that could negatively impact model training.
   - Process:
     - Missing Values: The dataset was thoroughly inspected for missing values across all features using Python's pandas library (e.g., df.isnull().sum()). As a synthetic dataset, *processed_fraud_data.csv* was pre-cleaned, and no missing values were found, eliminating the need for imputation or removal.
     - Duplicates: The dataset was checked for duplicate rows using df.duplicated().sum(). No duplicates were identified, confirming the dataset's integrity.
     - Inconsistencies: Feature-specific checks were performed to ensure data consistency:
       - CUSTOMER_ID and TERMINAL_ID were validated to ensure they were positive integers within reasonable ranges (1 to 9999 for CUSTOMER_ID, 1 to 999 for TERMINAL_ID).

        - TX_AMOUNT was verified to be positive and within a realistic range for financial transactions (e.g., no negative or unrealistically large values).
        - Temporal features (DAY, MONTH, YEAR, HOUR, MINUTE) were checked for valid ranges (e.g., DAY: 1–31, MONTH: 1–12, YEAR: 2018–2024, HOUR: 0–23, MINUTE: 0–59). No anomalies were detected.
    - Outlier Detection: TX_AMOUNT was inspected for outliers using descriptive statistics (e.g., df['TX_AMOUNT'].describe()). While high-amount transactions (> $220) were intentionally included as part of the fraud simulation, no erroneous outliers were found.
  - Outcome: The synthetic dataset was confirmed to be clean and consistent, requiring no additional cleaning steps. This ensured that subsequent preprocessing and modeling steps could proceed without data quality issues.
3. Data Splitting:
  - Objective: To divide the dataset into training and testing sets to enable model training and unbiased evaluation while preserving the distribution of the target variable (TX_FRAUD).
  - Process:
    - The dataset of 67,000 transactions was split into training (70%, 46,900 transactions) and testing (30%, 20,100 transactions) sets using scikit-learn's train_test_split function.
    - Stratified Sampling: Given the imbalanced nature of the dataset (approximately 1–2% fraudulent transactions), stratified sampling was employed by setting stratify=y in train_test_split. This ensured that the proportion of fraudulent transactions (TX_FRAUD = 1) was consistent between the training and test sets, preventing biased evaluation.
    - Random Seed: A random seed of 42 was used (random_state=42) to ensure reproducibility of the split.
  - Outcome: The training set contained 46,900 transactions, and the test set contained 20,100 transactions, with both sets maintaining the original fraud proportion (approximately 1–2%). This allowed for robust model training and

fair evaluation of performance.

4. Handling Class Imbalance:
    - Objective: To address the severe class imbalance in the dataset, where fraudulent transactions (TX_FRAUD = 1) constituted only 1–2% of the data, to ensure the model could effectively learn fraud patterns.
    - Challenge: Imbalanced datasets can lead to models that prioritize the majority class (safe transactions), resulting in poor detection of the minority class (fraudulent transactions), which is critical for fraud detection.
    - Process:
        - SMOTE Application: The Synthetic Minority Oversampling Technique (SMOTE) from the imblearn library was used to oversample the minority class (fraudulent transactions) during model training. SMOTE generates synthetic samples for the minority class by interpolating between existing samples, preserving the underlying distribution while increasing representation.
        - Sampling Strategy: A sampling strategy of 0.12 was chosen (sampling_strategy=0.12), meaning the number of fraudulent transactions was increased to approximately 12% of the training set. This was a deliberate choice to balance the dataset without overwhelming it with synthetic data, which could lead to overfitting.
        - Integration in Pipeline: SMOTE was incorporated into the machine learning pipeline (imblearn.pipeline.Pipeline) to ensure it was applied only to the training data during cross-validation and final training, avoiding data leakage into the test set.
        - Tuning Considerations: The sampling strategy was tuned through experimentation to optimize recall (sensitivity to fraudulent transactions) while maintaining overall accuracy and avoiding overfitting. A value of 0.12 was found to strike an effective balance.
    - Outcome: SMOTE increased the representation of fraudulent transactions in the training set, enabling the model to learn fraud patterns effectively without compromising performance on the majority class. This was critical for achieving high recall (97.22% on the test set), ensuring minimal missed frauds.

5. Feature Scaling:
    - Objective: To normalize numerical features to a consistent scale, improving model performance and convergence, particularly for features with varying ranges like TX_AMOUNT.
    - Process:
        - StandardScaler: Scikit-learn's StandardScaler was applied to numerical features (TX_AMOUNT, DAY, MONTH, YEAR, HOUR, MINUTE) to standardize them to a mean of 0 and a standard deviation of 1. This transformation was performed as part of the machine learning pipeline to ensure consistency between training and test sets.
        - Rationale: While tree-based models like AdaBoost with DecisionTreeClassifier as the base estimator are generally insensitive to feature scales, standardizing numerical features was beneficial for:
            - Ensuring compatibility with potential alternative algorithms (e.g., Logistic Regression) during model selection.
            - Improving the numerical stability of SMOTE, which relies on distance-based calculations to generate synthetic samples.
            - Facilitating future extensions of the pipeline to include non-tree-based models.
        - Categorical Features: CUSTOMER_ID and TERMINAL_ID were not scaled, as they were treated as categorical integers suitable for tree-based models.
        - Pipeline Integration: The StandardScaler was included in the pipeline (imblearn.pipeline.Pipeline) to ensure that scaling was applied consistently during training and prediction, avoiding data leakage.
    - Outcome: Feature scaling normalized the distributions of numerical features, particularly TX_AMOUNT, which ranged from small values (e.g., $1) to large values (e.g., >$220 for fraudulent transactions). This ensured the model could effectively process features with different scales, contributing to its high performance.
6. Pipeline Integration:
    - To streamline preprocessing and ensure reproducibility, all preprocessing steps

(SMOTE and StandardScaler) were integrated into a single pipeline using imblearn.pipeline.Pipeline. The pipeline consisted of:

- ('smote', SMOTE(sampling_strategy=0.12, random_state=42)): Oversampling the minority class.
- ('scaler', StandardScaler()): Standardizing numerical features.
- ('model', AdaBoostClassifier(...)): The machine learning model (detailed in the Model Development section).

- This pipeline ensured that preprocessing steps were applied consistently during training, cross-validation, and prediction, preventing data leakage and simplifying the workflow.

7. Validation of Preprocessing:
   - Exploratory Data Analysis (EDA): Before preprocessing, EDA was conducted using pandas and visualization libraries (e.g., matplotlib, seaborn) to understand feature distributions and relationships. For example:
     - Histograms of TX_AMOUNT revealed a right-skewed distribution, with a small number of high-value transactions corresponding to the high-amount fraud scenario.
     - Temporal features (HOUR, DAY) showed patterns, such as higher fraud risk during late-night hours, aligning with the fraud scenarios.
   - Post-Preprocessing Checks: After applying SMOTE and scaling, the training set was inspected to confirm the increased proportion of fraudulent transactions (approximately 12%) and the standardized distributions of numerical features (mean ≈ 0, std ≈ 1).
   - Reproducibility: A random seed of 42 was used for SMOTE and data splitting to ensure consistent results across runs.

Outcome: The preprocessing steps transformed the raw dataset into a clean, structured, and model-ready format. By retaining all relevant features, cleaning the data, splitting it with stratification, addressing class imbalance with SMOTE, and standardizing numerical features, the dataset was optimally prepared for training a robust fraud detection model. These steps directly contributed to the model's high performance, as evidenced by the test set metrics (accuracy: 99.83%, recall: 97.22%, ROC AUC: 98.57%). The preprocessing pipeline's integration ensured efficiency, reproducibility, and scalability, making it suitable

for both development and deployment in the Streamlit application.

MODEL DEVELOPMENT

The development of the fraud detection model for the "Fraud Transaction Detection Using Transactions Dataset" project was a critical component of the system, aimed at achieving high-accuracy detection of fraudulent transactions within a synthetic dataset of 67,000 records. The model was built using the AdaBoostClassifier from scikit-learn, selected for its robustness in handling imbalanced datasets and its ability to combine weak learners into a strong ensemble model through boosting. The model development process was meticulously structured to ensure optimal performance, involving algorithm selection, pipeline construction, hyperparameter tuning, model training and evaluation, and model saving. Each stage was carefully executed to address the challenges of fraud detection, particularly the imbalanced nature of the dataset, where fraudulent transactions constituted only 1–2% of the data. The resulting model achieved exceptional performance metrics, making it a reliable component of the fraud detection system. Below is a detailed description of each stage in the model development process.

The first stage was algorithm selection, where multiple machine learning algorithms were evaluated to identify the most suitable one for the task. The evaluated algorithms included Logistic Regression, Decision Tree Classifier, Random Forest Classifier, AdaBoostClassifier, and GradientBoostingClassifier, all implemented using scikit-learn. Each algorithm was assessed for its ability to handle the imbalanced dataset and detect the minority class (fraudulent transactions) effectively. Logistic Regression, a linear model, was tested but struggled with the non-linear patterns in the fraud scenarios. The Decision Tree Classifier provided interpretability but was prone to overfitting on the imbalanced data. Random Forest Classifier, an ensemble method, performed well but was computationally intensive. GradientBoostingClassifier showed strong performance but was sensitive to hyperparameter settings. The AdaBoostClassifier was ultimately selected due to its superior performance on the imbalanced dataset, leveraging the boosting technique to iteratively focus on misclassified instances, which are often fraudulent transactions in this context. The base estimator for AdaBoost was a DecisionTreeClassifier, chosen for its flexibility and interpretability as a weak learner. The combination of AdaBoost’s boosting mechanism and the DecisionTreeClassifier’s ability to capture non-linear relationships made it an ideal

choice for this project. To ensure a fair comparison, each algorithm was trained on the preprocessed dataset (after splitting into training and test sets), and preliminary performance metrics (accuracy, F1 score, recall, and ROC AUC) were calculated, confirming AdaBoost's edge, particularly in recall and ROC AUC, which are critical for fraud detection.

The second stage involved constructing a machine learning pipeline to streamline preprocessing and modeling, ensuring reproducibility and efficiency. The pipeline was built using the imblearn.pipeline.Pipeline class from the imbalanced-learn library, which seamlessly integrated preprocessing steps with the model. The pipeline consisted of three components: the Synthetic Minority Oversampling Technique (SMOTE) to address class imbalance by generating synthetic samples for the minority class (fraudulent transactions) with a sampling strategy of 0.12, increasing the proportion of fraudulent transactions to approximately 12% in the training set; the StandardScaler to standardize numerical features (TX_AMOUNT, DAY, MONTH, YEAR, HOUR, MINUTE) to a mean of 0 and a standard deviation of 1, ensuring consistent scales and improving the numerical stability of SMOTE and potential non-tree-based models; and the AdaBoostClassifier, configured with a DecisionTreeClassifier as the base estimator. This pipeline ensured that preprocessing steps (SMOTE and scaling) were applied only to the training data during cross-validation and final training, preventing data leakage into the test set. The use of a pipeline simplified the workflow, reduced the risk of errors, and made the model development process more reproducible. The pipeline was designed to handle the unique challenges of the dataset, such as the severe class imbalance and the varying scales of numerical features, while maintaining compatibility with the tree-based AdaBoost model.

Hyperparameter tuning was the third stage, aimed at optimizing the AdaBoostClassifier's performance. Given the large number of hyperparameters and the computational cost of exhaustive search, RandomizedSearchCV with 3-fold stratified cross-validation was employed to explore the parameter space efficiently. Stratified cross-validation ensured that each fold maintained the proportion of fraudulent transactions, providing reliable estimates of model performance. The search space included key hyperparameters for both the AdaBoostClassifier and its base DecisionTreeClassifier: n_estimators (number of boosting iterations: [50, 100, 150, 200, 300]), learning_rate (weight applied to each classifier: [0.01, 0.05, 0.1, 0.2]), algorithm (boosting algorithm: ['SAMME', 'SAMME.R']), estimator__max_depth (maximum depth of the base decision tree: [1, 2, 3, 4]),

estimator__min_samples_split (minimum samples required to split a node: [2, 5, 10]), and estimator__min_samples_leaf (minimum samples required at a leaf node: [1, 2, 4]). RandomizedSearchCV performed 30 iterations, evaluating combinations of these parameters using the ROC AUC score as the scoring metric, given its suitability for imbalanced datasets. The best parameters identified were: n_estimators: 300, learning_rate: 0.2, algorithm: SAMME, estimator__max_depth: 3, estimator__min_samples_split: 10, and estimator__min_samples_leaf: 10. These parameters balanced model complexity and performance, with a deeper base tree (max_depth=3) allowing the model to capture complex patterns while regularization parameters (min_samples_split and min_samples_leaf) prevented overfitting. The best cross-validation ROC AUC score was 0.9877, indicating strong model performance and robustness across different data subsets. The tuning process was computationally intensive but critical for achieving optimal performance, particularly in maximizing recall to minimize missed fraudulent transactions.

The fourth stage was model training and evaluation, where the final model was trained using the optimized pipeline and the best hyperparameters. The training set (46,900 transactions) was used to fit the model, with SMOTE and StandardScaler applied as part of the pipeline. Performance was evaluated on both the training and test sets (20,100 transactions) using multiple metrics to ensure a comprehensive assessment of the model's effectiveness. For the training set, the metrics were: Accuracy Score: 0.9985, F1 Score: 0.9985, Recall Score: 0.9711, and ROC AUC Score: 0.9853. For the test set, the metrics were: Accuracy Score: 0.9983, F1 Score: 0.9983, Recall Score: 0.9722, and ROC AUC Score: 0.9857. These metrics demonstrate the model's ability to accurately classify transactions while maintaining a high recall, ensuring that 97.22% of fraudulent transactions were correctly identified on the test set, which is critical for minimizing false negatives (missed frauds) in fraud detection. The high ROC AUC score (0.9857) indicates excellent discrimination between fraudulent and safe transactions. To further validate the model's performance, a confusion matrix was generated, showing a low number of false negatives and false positives, confirming the model's balanced performance across both classes. A classification report provided additional metrics, such as precision and F1 score per class, further validating the model's effectiveness. The close alignment of training and test set metrics suggested that the model was well-generalized, with minimal overfitting. The high recall was particularly important, as missing fraudulent transactions could have significant financial implications

in a real-world setting.

The final stage was model saving, where the trained pipeline (including SMOTE, StandardScaler, and the optimized AdaBoostClassifier) was serialized using joblib and saved as *ada_model.pkl*. This file was designed to be seamlessly integrated into the Streamlit web application (*app.py*), enabling real-time and batch predictions. The saved model encapsulated all preprocessing and modeling steps, ensuring that new data could be processed consistently without requiring manual preprocessing. The use of joblib ensured efficient storage and loading of the model, which was critical for the application's performance, especially when processing large batches of transactions.

The model development process resulted in a highly accurate and reliable fraud detection system, capable of handling the complexities of the synthetic dataset. The AdaBoostClassifier's ability to focus on misclassified instances, combined with the optimized pipeline and hyperparameters, enabled the model to effectively detect fraudulent transactions across the three simulated fraud scenarios (high-amount fraud, terminal fraud, and customer fraud). The exceptional performance metrics (test accuracy: 99.83%, recall: 97.22%, ROC AUC: 98.57%) demonstrate the model's suitability for real-world fraud detection, where high recall is critical to minimize missed frauds. The integration of the model into a reproducible pipeline and its serialization as *ada_model.pkl* ensured seamless deployment in the Streamlit application, making it a practical and scalable solution for financial fraud detection.

APPLICATION DEVELOPMENT

The development of the web application for the "Fraud Transaction Detection Using Transactions Dataset" project was a pivotal component in making the fraud detection system accessible and practical for end-users. Built using the Streamlit framework and implemented in the file *app.py*, the application provides a robust, interactive, and visually appealing interface that enables users to analyze financial transactions for potential fraud in real time or in batches. The application integrates the trained machine learning model (*ada_model.pkl*), leverages Plotly for interactive visualizations, and incorporates a custom-designed user interface to ensure usability across a wide range of users, including financial analysts, business stakeholders, and non-technical personnel. The application's design prioritizes accessibility, performance, and clarity, delivering actionable insights through predictions, confidence scores, risk level classifications, and dynamic visualizations. The development process encompassed designing the user interface, implementing two operational modes (single transaction analysis and batch processing), and ensuring robust technical integration and performance optimization. Below is a detailed exploration of each aspect of the application development process.

The design and user interface of the Streamlit application were carefully crafted to provide an intuitive and visually engaging experience. A custom dark theme was implemented to enhance readability and aesthetic appeal, featuring a gradient background that transitions smoothly across dark shades (from #0c0c0c to #1a1a2e to #16213e), creating a modern and professional look suitable for financial applications. Gradient animations were applied to headers and buttons, using CSS animations to cycle through vibrant colors (e.g., #ff6b6b, #4ecdc4, #45b7d1), adding a dynamic and engaging effect without overwhelming the user. Key metrics, such as fraud count, fraud rate, safe transaction count, and average fraud amount, are displayed in visually distinct metric cards styled with a glassmorphism effect, achieved through semi-transparent backgrounds (rgba(255, 255, 255, 0.05)), subtle borders, and blur effects (backdrop-filter: blur(10px)). These cards use bold typography and highlighted numbers (styled with a glowing effect using #4ecdc4 and text-shadow) to draw attention to critical insights. The interface leverages Streamlit's column-based layout to organize content efficiently, ensuring that input fields, visualizations, and results are

presented in a clear and logical manner. For example, in single transaction analysis, input fields are arranged in a two-column layout for transaction details and risk visualizations, while batch processing results are displayed in a four-column grid for summary metrics. The sidebar, styled with a semi-transparent dark background, serves as a control panel, providing options to switch between operational modes and display model information (e.g., algorithm: AdaBoostClassifier, accuracy: 99.83%, ROC AUC: 98.57%). This responsive and aesthetically consistent design ensures that users can interact with the application effortlessly, regardless of their technical expertise.

The application supports two operational modes: single transaction analysis and batch processing, each tailored to different use cases. In single transaction analysis, users can manually input transaction details through an intuitive interface featuring interactive widgets, such as number inputs for CUSTOMER_ID (1–9999), TERMINAL_ID (1–999), and TX_AMOUNT (0.01–10,000), and select boxes for DAY (1–31), MONTH (1–12), YEAR (2022–2024), HOUR (0–23), and MINUTE (0–59). Default values (e.g., CUSTOMER_ID: 124, TX_AMOUNT: 30.0, DAY: 15) are provided to streamline testing. Upon clicking the "Analyze Transaction" button, styled with a gradient background and hover effects, the system loads the *ada_model.pkl* using joblib and predicts the fraud status (Safe or Fraud) along with a confidence score (fraud probability). The prediction is displayed in an animated alert box—red for fraud (with a pulsing animation) or green for safe—enhancing visual feedback. A risk factor analysis is presented as a horizontal bar chart using Plotly, showing four simulated risk factors: Amount Risk (calculated as TX_AMOUNT / 1000 * 100, capped at 100%), Time Risk (elevated to 20% for transactions between 10 PM and 6 AM, otherwise 10%), Customer Risk (simulated with random values between 5–25%), and Terminal Risk (simulated with random values between 5–30%). The chart uses a RdYlBu_r colorscale to indicate risk severity, with a transparent background to match the dark theme. The risk level is classified as LOW (<30% fraud probability), MEDIUM (30–70%), or HIGH (>70%) and displayed in a metric widget alongside the fraud probability, with a delta indicator showing the deviation from a 50% threshold. This mode is ideal for real-time analysis, enabling users to quickly assess individual transactions.

In batch processing mode, users can upload a CSV file containing transactions with the required columns (CUSTOMER_ID, TERMINAL_ID, TX_AMOUNT, DAY, MONTH, YEAR, HOUR, MINUTE). The application validates the file's structure, displaying an error

if the required columns are missing, and processes the entire dataset using the *ada_model.pkl*. Predictions include fraud status, fraud probability, and risk level (LOW, MEDIUM, HIGH) for each transaction, with risk levels assigned based on probability bins (0–0.3, 0.3–0.7, 0.7–1.0). Summary metrics are displayed in four metric cards: number of frauds detected, number of safe transactions, fraud rate (percentage of fraudulent transactions), and average transaction amount for fraudulent transactions. These metrics are calculated dynamically using pandas (e.g., sum(predictions) for fraud count, len(predictions) - fraud_count for safe transactions). Visualizations include a Plotly histogram of transaction amounts, color-coded by fraud status (red for fraud, green for safe), and a pie chart showing the distribution of risk levels (LOW: green, MEDIUM: orange, HIGH: red), both styled with a transparent background to align with the dark theme. Users can download the processed results as a CSV file (*fraud_analysis_results.csv*) using Streamlit's download_button, facilitating further analysis or reporting. A sample dataframe is displayed if no file is uploaded, illustrating the expected format with example transactions. This mode is designed for analyzing large datasets, such as daily transaction logs, making it suitable for financial institutions processing high volumes of transactions.

The technical implementation of the application focused on seamless model integration, performance optimization, and robust error handling. The *ada_model.pkl* is loaded using joblib's joblib.load function, wrapped in a @st.cache_resource decorator to cache the model in memory, reducing load times for subsequent predictions. The model is used to predict fraud status (model.predict) and probabilities (model.predict_proba) for both single and batch inputs, with predictions processed efficiently using pandas DataFrames. Plotly is used for all visualizations, ensuring interactivity (e.g., hover details, zooming) and consistency with the dark theme through custom layout settings (e.g., plot_bgcolor='rgba(0,0,0,0)', font=dict(color='white')). Performance optimization is achieved through Streamlit's caching mechanisms: @st.cache_resource for the model and @st.cache_data for sample data generation, minimizing redundant computations. Error handling includes checks for the presence of *ada_model.pkl* (displaying an error with st.error if missing) and validation of uploaded CSV files to ensure they contain the required columns, with a sample dataframe provided to guide users. A simulated processing delay (time.sleep(2)) is added for single transaction predictions to enhance the user experience by mimicking real-time analysis. The application also includes a footer with attribution and branding, styled with centered text and

reduced opacity for subtlety.

The Streamlit application is fully functional, providing a seamless and intuitive user experience for both real-time and batch fraud detection. Its custom-designed interface, robust operational modes, and efficient technical implementation make it a powerful tool for end-users. The application's ability to deliver clear predictions, actionable insights through visualizations, and downloadable results ensures its applicability in real-world financial settings, while its scalability and performance optimization support deployment in high-volume environments. The successful integration of the *ada_model.pkl* with the Streamlit framework demonstrates the project's end-to-end capability, from model development to user-facing deployment, making it a comprehensive solution for combating financial fraud.

RESULTS AND ACHIEVEMENTS

The "Fraud Transaction Detection Using Transactions Dataset" project delivered a robust and high-performing fraud detection system that successfully met its objectives of providing an accurate, scalable, and user-friendly solution for identifying fraudulent financial transactions. The system's effectiveness is demonstrated through several key achievements that highlight its technical excellence and practical applicability. The AdaBoostClassifier model, trained on a synthetic dataset of 67,000 transactions, achieved exceptional performance metrics on the test set, with an accuracy of 99.83%, an F1 score of 99.83%, a recall of 97.22%, and a ROC AUC score of 98.57%. These metrics underscore the model's ability to accurately classify transactions as fraudulent or safe while maintaining a high recall, ensuring that 97.22% of fraudulent transactions were correctly identified, thereby minimizing missed frauds, which is critical for real-world financial applications where false negatives can lead to significant losses. The synthetic dataset incorporated three distinct fraud scenarios—high-amount fraud (transactions exceeding $220), terminal-based fraud (randomly selected terminals processing fraudulent transactions for 28 days), and customer-based fraud (randomly selected customers with one-third of their transactions modified to high amounts for 14 days)—ensuring that the model was trained and validated on diverse and realistic fraud patterns reflective of real-world challenges, such as phishing attacks, compromised terminals, and card-not-present fraud. The Streamlit web application, implemented in *app.py*, provides a visually appealing and intuitive interface that supports both single transaction analysis and batch processing, making the system accessible to a wide range of users, from financial analysts to business stakeholders. Interactive visualizations, powered by Plotly, include histograms of transaction amounts, pie charts of risk level distributions, and risk factor bar charts, enhancing usability by presenting complex model outputs in an easily interpretable format. Users can download batch processing results as a CSV file, facilitating further analysis and reporting. The system is designed for scalability, capable of handling large datasets, and is adaptable to new data sources, with the fraud simulation logic well-documented in *NoteBook.txt* to ensure transparency and reproducibility. The machine learning pipeline, integrating SMOTE, StandardScaler, and the

AdaBoostClassifier, is fully reproducible, allowing for consistent model retraining and deployment. The application provides comprehensive metrics, including fraud probability, risk level classifications (LOW, MEDIUM, HIGH), and summary statistics such as fraud count, fraud rate, and average fraud amount, enabling informed decision-making for end-users. These achievements collectively demonstrate the project's success in delivering a practical, high-performance solution for fraud detection that aligns with industry needs and sets a strong foundation for combating financial fraud effectively.

CHALLENGES AND LEARNINGS

The development of the fraud detection system presented several challenges that required innovative solutions and provided valuable learning opportunities, enhancing technical expertise and problem-solving skills. The highly imbalanced dataset, with fraudulent transactions comprising only 1–2% of the 67,000 records, posed a significant challenge, as it risked the model overfitting to the majority class (safe transactions), potentially leading to poor detection of fraudulent transactions. This was addressed by applying the Synthetic Minority Oversampling Technique (SMOTE) with a carefully tuned sampling strategy of 0.12, which increased the proportion of fraudulent transactions to approximately 12% in the training set without introducing excessive synthetic data that could cause overfitting. Experimentation was required to balance recall and overall model performance, deepening understanding of imbalanced dataset techniques and the trade-offs between oversampling and model generalization. Hyperparameter tuning for the AdaBoostClassifier was another challenge, given its large hyperparameter space and the computational expense of exhaustive search. RandomizedSearchCV with 3-fold stratified cross-validation was employed to efficiently explore parameters such as n_estimators (50–300), learning_rate (0.01–0.2), algorithm (SAMME, SAMME.R), and base estimator parameters (max_depth: 1–4, min_samples_split: 2–10, min_samples_leaf: 1–4), optimizing for ROC AUC score. This process highlighted the importance of balancing computational resources with model optimization and demonstrated the effectiveness of randomized search for large parameter spaces, resulting in a best cross-validation ROC AUC of 0.9877. Developing the Streamlit application required careful planning to design an interactive and visually appealing interface while ensuring seamless integration with the *ada_model.pkl*. Custom CSS was used to create a dark theme with gradient animations, and Plotly was integrated for interactive visualizations, such as histograms and pie charts, styled to match the application's aesthetic. Streamlit's caching mechanisms (@st.cache_resource, @st.cache_data) were implemented to optimize performance, particularly for model loading and batch processing, teaching valuable lessons in web application development, UI/UX design, and performance optimization. Designing realistic fraud scenarios for the synthetic dataset was a complex task, requiring careful consideration to ensure they were both challenging and representative of real-world fraud patterns, such as high-

amount transactions, compromised terminals, and stolen credentials. The three scenarios were implemented based on common fraud patterns and validated through the model's performance on the test set (recall: 97.22%), reinforcing the importance of domain knowledge in dataset design and fraud detection. These challenges collectively enhanced technical skills in machine learning, data preprocessing, web application development, and project management, providing a deeper understanding of practical AI system development and the ability to address complex real-world problems effectively.

CONCLUSION

The "Fraud Transaction Detection Using Transactions Dataset" project, completed during my internship at Unified Mentor, successfully delivered a high-performance AI-based fraud detection system that addresses the critical need for robust fraud mitigation in financial transactions. The AdaBoostClassifier, trained on a synthetic dataset of 67,000 transactions with realistic fraud scenarios (high-amount, terminal-based, and customer-based), achieved exceptional performance metrics, including a test accuracy of 99.83%, a recall of 97.22%, and a ROC AUC score of 98.57%, demonstrating its ability to accurately detect fraudulent transactions while minimizing missed frauds. The Streamlit web application, implemented in *app.py*, provides a user-friendly interface for both single transaction analysis and batch processing, featuring a custom dark theme, interactive Plotly visualizations (e.g., histograms, pie charts, risk factor bar charts), and downloadable results, making complex model outputs accessible to non-technical users. The project addressed significant challenges, including class imbalance through SMOTE, hyperparameter tuning with RandomizedSearchCV, and application development with Streamlit and Plotly, resulting in valuable learnings in machine learning, data preprocessing, web development, UI/UX design, and fraud scenario design. These experiences provided hands-on expertise in building end-to-end AI solutions, from data preprocessing to model deployment and user interface design, equipping me with a strong foundation for future projects in artificial intelligence, financial technology, and fraud detection. The system's scalability, accuracy, and usability make it a valuable tool for financial institutions seeking to combat fraud effectively, while its modular design and documented processes ensure adaptability for future enhancements, such as real-time data integration, advanced feature engineering, or deep learning models. By combining cutting-edge machine

learning with a user-centric application, the project delivers a comprehensive and practical solution that aligns with industry needs and demonstrates the transformative potential of AI in enhancing financial security.