

AI-Driven Fraud Detection System for Financial Transactions

ZUBER MULLA – U2258708



University of
HUDDERSFIELD

1 CONTENTS

Abstract.....	5
2 Introduction	5
2.1 Background.....	5
2.2 Problem Statement	5
2.3 Aims and objectives.....	5
2.4 Significance of my study	6
3 Literature Review	6
3.1 Introduction to credit card fraud.....	6
3.2 Traditional Fraud Detection Techniques	7
3.3 Machine Learning in Fraud Detection	7
3.4 Handling Imbalanced Data	8
3.4.1 Why Imbalance Undermines Model Performance.....	8
3.4.2 Resampling Strategies: SMOTE and ADASYN	8
3.4.3 Comparative Performance and Limitations	9
3.5 Feature Selection/Engineering and Data Preprocessing	9
3.5.1 Scaling and Normalisation.....	9
3.5.2 Dimensionality Reduction and Feature Selection	9
3.5.3 Handling Time-Based and Contextual Features	9
3.5.4 Missing Data and Outlier Handling	10
3.6 Evaluation Metrics in Fraud Detection	10
3.6.1 Accuracy	10
3.6.2 Precision And Recall	11
3.6.3 F1-Score.....	11
3.6.4 AUC-ROC and Precision–Recall Curves.....	11
3.6.5 Confusion And Correlation Matrix	12
3.7 Comparison and Evaluation of Related Work / Other Models.....	12
3.7.1 Ensemble Learning with Basic Evaluation	12
3.7.2 Undersampling with Balanced Metrics	12
3.7.3 Graph-Based Modelling for Entity Relationships	12
3.7.4 Feature Filtering and Oversampling in Ensemble Models	13
3.8 Limitations of Current Research and Technology	13
3.8.1 Limited Access to High-Quality, Real-World Datasets	13
3.8.2 Imbalance and Label Noise in Data	13
3.8.3 Lack of Explainability in Complex Models	13

3.8.4	Generalisation and Real-Time Adaptation	14
3.9	Conclusion	14
4	PLESI	14
4.1	Professional Issues (P)	14
4.2	Legal issues (L)	15
4.3	Ethical issues (E)	15
4.4	Social issues (S)	16
5	Methodology.....	17
5.1	Dataset overview	17
5.2	Dataset Structure and Features.....	18
5.3	Preprocessing and feature engineering	18
5.3.1	Box-Cox Transformation of Transaction Amounts	19
5.3.2	Cyclical Encoding of Time-Based Features	19
5.4	Machine Learning Models and Resampling Techniques	20
5.4.1	Machine Learning Models.....	20
5.4.2	Why SMOTE and ADASYN Were Chosen.....	21
5.5	Evaluation Metrics used	21
5.6	Experimental Setup and workflow	22
5.6.1	Train Test Split and Baselines.....	22
5.6.2	Metric Logging & Reproducibility.....	23
5.6.3	Critical Workflow Reflection	23
5.7	Summary of methodological choices	23
6	Evaluation and Discussion.....	23
6.1	Evaluation Strategy and Baseline Analysis	24
6.1.1	Purpose of Evaluation	24
6.1.2	Establishing the Baselines	24
6.2	Evaluation of oversampling methods.....	25
6.2.1	Visualising SMOTE and ADASYN.....	25
6.3	Model Performance Evaluation.....	25
6.3.1	Comparison Summary Table and Observations	26
6.4	Testing process and tools	27
6.4.1	Reproducibility and Logging	27
6.4.2	Testing Scope	27
6.5	Limitations of the Approach	28
6.5.1	Dataset and Feature Constraints	28
6.5.2	Sampling Risks and Infrastructure Limitations.....	28

6.6	Summary of Key Findings	28
6.6.1	Final Model Performance Overview	28
6.7	Webpage and Model Review.....	29
6.7.1	Backend API And Model Integration	29
6.7.2	Overall Deployment Architecture	30
6.7.3	Limitations and Future work	31
6.8	Evaluation And Review Summary.....	31
7	Discussion, Review and Moving Forward	32
7.1	Review of Aims and Problems	32
7.2	Learning and Insights.....	33
7.3	Skills and Competencies Developed	33
7.4	Self-Criticism and Reflection.....	34
7.5	Future Plans.....	34
7.6	Evaluation and Conclusion of the Entire Project.....	35
8	References	36
9	Appendices.....	38
	Project Ethical Review Form	38
	Model Python Script	40
	Testing and Evaluation Results	43
	Webpage And Interface Details	45

ABSTRACT

This project presents the development of an AI-driven fraud detection system capable of identifying fraudulent financial transactions. Using a real-world credit card transaction dataset, the system was built through a comprehensive pipeline involving data preprocessing, class balancing, model comparison, evaluation, and deployment. The dataset's extreme imbalance was addressed using **SMOTE** and **ADASYN**, with models including **Random Forest**, **Logistic Regression**, **KNN**, and **SVM** assessed across various metrics. After thorough evaluation, **Random Forest** trained with **SMOTE** emerged as the optimal model, achieving strong **precision**, **recall**, **F1-score**, and **PR AUC** — all critical in handling rare fraud cases. Evaluation was enhanced with visual tools such as **t-SNE** and confusion matrices to support performance interpretation. In addition to model development, a fully functional interface was created using Next.js and deployed alongside a backend API built with Flask. This allowed users to input transactions and receive fraud predictions in real time. All preprocessing logic was embedded in the API to ensure consistency with the training pipeline.

2 INTRODUCTION

2.1 BACKGROUND

In today's digital world, financial transactions are happening faster and more frequently than ever before. With this increase in online activity, there's also been a rise in fraudulent behaviour targeting payment systems, especially credit card transactions. Fraud not only causes financial loss to customers and banks but also reduces overall trust in digital financial systems. Traditional fraud detection systems struggle to keep up with the speed and complexity of these attacks. Existing fraud detection systems often rely on predefined rules that are easily bypassed by new fraud strategies.

2.2 PROBLEM STATEMENT

'AI-Driven Fraud Detection System for Financial Transactions'

As fraud techniques become more advanced, financial systems require smarter, faster, and more reliable ways to detect fraud. Rule-based systems and manual checks are no longer sufficient. This project tackles the challenge of creating an AI-based solution that can analyse large volumes of transaction data, learn patterns from previous fraud cases, and accurately identify new fraud attempts in real-time.

2.3 AIMS AND OBJECTIVES

The aim of this project is to develop a fraud detection system using machine learning that can predict fraudulent transactions.

The main objectives are:

- Research different types of credit card fraud (e.g. identity theft, card-not-present fraud, synthetic fraud).
- Analyse a real-world credit card transaction dataset to understand trends and challenges.
- Pre-process the data by handling imbalances, scaling, and removing irrelevant features.
- Apply and compare different machine learning algorithms (**Random Forest**, **Logistic Regression**, **KNN**, **SVM**).

- Evaluate each model using metrics like **Accuracy, Precision, Recall, F1-Score**, and AUC.
- Test the final model rigorously for its performance on rare fraud cases.
- Build a simple interface to interact with the trained model.
- Reflect on results and suggest possible improvements or future research (e.g. using deep learning or real-time detection).

2.4 SIGNIFICANCE OF MY STUDY

Credit card fraud causes billions of dollars in damages each year. A system capable of detecting fraud in real time using machine learning might make a significant impact – not only in terms of cost savings, but also in helping individuals feel more secure when using digital payment methods. This initiative also has broader implications: the same techniques developed here might be applied to other sectors such as insurance fraud, internet scams, and cybercrime. Furthermore, working on this project provides a valuable opportunity to use data science, machine learning, software development and cyber security abilities in a real-world context.

3 LITERATURE REVIEW

3.1 INTRODUCTION TO CREDIT CARD FRAUD

Credit card fraud is a big problem in today's financial ecosystem, with attackers exploiting both technological flaws and human weaknesses. As financial services move online, the attack surface expands, making fraud not only more widespread, but also more difficult to detect. According to the Nilson Report (2022), global card fraud losses totalled over \$32 billion in 2021 and are expected to exceed \$40 billion by 2027. The growth of digital payment systems, e-commerce, and mobile banking has made it simpler for attackers to carry out large-scale fraud. There are multiple types of credit card fraud;

- **Card-Not-Present (CNP) Fraud:** This happens when you make a purchase online or over the phone without using your actual card. Because of the growth of e-commerce, it has become the most popular type worldwide. Europol (2021) said that CNP fraud accounted for around 80% of card fraud losses in the European Union.
- **Account Takeover Fraud:** Fraudsters acquire access to a legitimate customer's account, usually via phishing or credential stuffing assaults. Once entered, they may modify account information or conduct unauthorised activities (Symantec, 2020).
- **Lost or Stolen Card Fraud:** This involves physically possessing a card and then using it without the owner's knowledge. EMV chip technology has reduced but not eliminated this sort of fraud (Delamaire, Abdou, & Pointon, 2009).
- **Application Fraud:** Criminals apply for new credit cards using stolen or false identities. This often goes undetected until the account enters collections for non-payment.
- **Synthetic Identity Fraud:** One of the most difficult types to detect, this involves blending real and fake information. The U.S. Federal Reserve (2020) described this as one of the fastest-growing financial crimes, especially damaging because these identities can build up a trustworthy credit profile over time before defaulting.

These different kinds of fraud are difficult to identify because they frequently imitate legitimate user behaviour, making rule-based or static detection approaches ineffective.

Fraud detection must also be real-time and dynamic. West and Bhattacharya (2016) note that fraudsters quickly adapt to detection methods, forcing systems to evolve constantly. This calls for approaches that can learn from data, adapt to new behaviours, and work under uncertainty.

3.2 TRADITIONAL FRAUD DETECTION TECHNIQUES

Prior to the rise of machine learning, fraud detection was mostly dependent on rule-based systems and manual inspections. These systems employed hard-coded logic, such as flagging a transaction if it exceeded a specific threshold or happened in a high-risk zone. While effective to some extent, these systems lacked flexibility and were unable to detect increasingly subtle or emerging kinds of fraud.

According to Bolton and Hand (2002), previous systems frequently used statistical approaches such as **Logistic Regression** or Bayesian models to identify fraud, with predetermined thresholds for unusual behaviour. However, these models were limited by their reliance on labelled data and inability to learn new patterns without manual intervention.

Delamaire, Abdou, and Pointon (2009) discovered that while rule-based approaches were effective for known fraud types, they struggled with real-time processing and produced significant false-positive rates, affecting user experience and confidence. This was especially true for card-not-present fraud (CNP), since the lack of physical verification made it difficult to authenticate transactions fast.

Another significant disadvantage of traditional methods is their inflexibility. As West and Bhattacharya (2016) point out, fraudsters are continually adapting, requiring a rapid development of detection systems. Rule-based systems cannot effectively adapt to developing fraud patterns unless manually updated, making them more obsolete in today's fast-paced financial environment.

3.3 MACHINE LEARNING IN FRAUD DETECTION

Machine learning (ML) has emerged as a significant element in fraud detection systems, providing a scalable and flexible way to spotting fraudulent transactions in real time. Machine learning models, as opposed to traditional rule-based systems, depend on data rather than predetermined criteria. They learn behavioural patterns from vast datasets and generalise these patterns to make predictions on new, previously unknown data, which is critical against more complex fraud tactics.

A wide range of machine learning approaches have been used in the fraud detection field. **Random Forest**, **Support Vector Machines (SVM)**, **Logistic Regression**, and **K-Nearest Neighbours (KNN)** are examples of supervised learning models that have demonstrated significant potential for detecting complex fraud patterns. These algorithms work with assigned data, learning to discriminate between fraudulent and non-fraudulent transactions. **Random Forest** has grown in popularity because of its stability and capacity to handle high-dimensional data (Bhattacharyya et al., 2011).

Unsupervised learning, on the other hand, is used when labels are unavailable or insufficient. Clustering and anomaly detection techniques (e.g., Isolation Forests, One-Class **SVM**) seek to discover outliers that differ from expected behaviour. However, as Phua et al. (2010) point out, unsupervised algorithms might suffer from high false-positive rates due to the intricacy of fraudulent patterns and these models frequently flag legal activity as suspicious, reducing the system's confidence and usability and this results in operational inefficiencies, such as needless human checks or the blockage of lawful transactions, which can reduce customer satisfaction.

Despite its potential, machine learning techniques have limitations. They require high-quality, up-to-date data to function efficiently. This is because fraud trends change quickly as criminals adjust to detection methods. When a model is trained on obsolete fraud behaviour, its predictions grow less accurate over time, which is known as *idea drift* (Gama et al. 2014). This necessitates constant model retraining and performance monitoring in production contexts. Second, many ML models function as black boxes, making the decision-making process difficult to understand. This lack of transparency causes problems in regulated industries such as banking, where institutions must explain why they took a choice, such as reporting a transaction as fraudulent (Doshi-Velez & Kim, 2017). This has implications not only for trust, but also for legal and regulatory issues.

3.4 HANDLING IMBALANCED DATA

In most real-world financial databases, fraudulent incidents make up less than 0.5% of all records (Dal Pozzolo et al., 2015). This class imbalance means ML models typically prioritise the majority class during training. As a result, machine learning algorithms may "learn" that predicting everything as non-fraud gives high **Accuracy** – which is misleading. According to Dal Pozzolo et al. (2015) and Van Vlasselaer et al. (2015), such models have a tendency to neglect the minority class, which is what we are attempting to discover here. As a result, many models achieve high overall **Accuracy** while failing to detect some genuine fraud, a crucial issue in safety-sensitive areas such as banking, healthcare, and cybersecurity (Davis & Goadrich, 2006; Leevy et al., 2022).

3.4.1 Why Imbalance Undermines Model Performance

Standard classifiers are intended to decrease overall error. In highly imbalanced data sets, this results in majority class domination, in which the model learns to predict the most common outcome. For example, predicting all transactions as valid might result in >99% **Accuracy**, while detecting no fraud (Bhattacharyya et al., 2011). This failure represents real-world dangers, since undiscovered fraud results in financial loss, regulatory exposure, and customer distrust.

3.4.2 Resampling Strategies: SMOTE and ADASYN

To solve these issues, resampling techniques are widely utilised to rebalance the dataset and make fraud patterns more understandable.

Numerous studies have shown that **SMOTE** is excellent in detecting fraud. For example, Van Vlasselaer et al. (2015) discovered that integrating **SMOTE** with ensemble classifiers resulted in higher **Recall** and **F1-Scores**, especially when dealing with skewed financial transaction datasets. Similarly, Pozzolo et al. (2014) demonstrated that **SMOTE** outperformed undersampling and cost-sensitive alternatives in balancing false positives and unreported frauds.

He et al. (2008) presented **ADASYN** (Adaptive Synthetic Sampling) as a refinement to **SMOTE**. While it generates synthetic samples, **ADASYN** concentrates on difficult-to-learn portions of the feature space, resulting in more synthetic data where the decision boundary is unclear. This has been demonstrated to increase classifier sensitivity in some settings (Zhang et al., 2021), but it may also generate noise if the data being used is too sparse or irregular.

Most importantly, both **SMOTE** and **ADASYN** are data-level solutions, which means they change the input distribution to improve learning. This makes them more interpretable and adaptable than algorithm-level techniques such as cost-sensitive learning, which penalise minority class misclassifications. While cost-sensitive approaches have proven effective in particular fraud scenarios (Elkan, 2001), they frequently need precise parameter adjusting and may lack transparency.

3.4.3 Comparative Performance and Limitations

Resampling increases model sensitivity to fraud, but it also brings trade-offs. Oversampling increases training time and has the potential to magnify noise, particularly in datasets with under-represented minority classes. Some research supports hybrid techniques, such as combining **SMOTE** with majority class undersampling, to retain dataset size while increasing minority class attention (Liu et al., 2009). Others emphasise the need of scaling features after resampling, especially for models sensitive to feature magnitude, such as **KNN** or **SVM** (Brown & Mues, 2012). Across domains such as healthcare (Leevy et al., 2022), credit scoring (Bahnsen et al., 2016), and network intrusion detection (Bhuyan et al., 2014), the literature consistently supports **SMOTE** as a reliable and widely adopted baseline, with **ADASYN** and cost-sensitive learning giving domain-specific advantages based on data quality and class separation.

3.5 FEATURE SELECTION/ENGINEERING AND DATA PREPROCESSING

Feature engineering and data preprocessing are critical aspects in ML. The way data is cleaned, converted, and formatted prior to training has a direct influence on model performance, especially in complex, high-dimensional, and unbalanced contexts such as financial fraud datasets. Dal Pozzolo et al. (2015) found that poor preprocessing might result in skewed learning patterns and exaggerated performance measures, particularly when models are assessed on noisy or unscaled data.

3.5.1 Scaling and Normalisation

In fraud detection tasks where features might have dramatically different numerical ranges, such as transaction amount vs. time delta, feature scaling is critical. Brown and Mues (2012) found that unsuitable scaling can bias model performance, with distance-sensitive classifiers showing significant differences in predictive ability when unscaled data is used. This step is much more important when paired with synthetic resampling techniques like **SMOTE** or **ADASYN**, because artificially created samples rely on feature space consistency.

3.5.2 Dimensionality Reduction and Feature Selection

Fraud datasets often include hundreds of features, many of which are redundant, noisy, or inadequately useful. Research has demonstrated that removing unnecessary or parallel elements might increase interpretability and generalisation. Bhattacharyya et al. (2011) found that doing principal component analysis (**PCA**) before classification decreased overfitting and training time while retaining good performance. Other researchers prefer simpler methods, such as correlation filtering or recursive feature reduction, to preserve model explainability, especially when choices must be transparent in regulated circumstances (Zareapoor & Shamsolmoali, 2015).

3.5.3 Handling Time-Based and Contextual Features

Temporal features, such as transaction timestamps or intervals, are crucial behavioural signals in fraud detection. Lebichot et al. (2019) investigated the transformation of time into engineering variables such as transaction frequency or time-of-day clustering to improve the identification of unusual behaviour patterns in transaction sequences. However, raw time data might be misleading due to their continuous and cyclic nature. To reveal hidden temporal behaviours without adding artificial breaks, studies frequently advocate time encoding approaches, such as breaking timestamps by hour-of-day or using cyclic transformations (e.g., **sine/cosine** of hour).

3.5.4 Missing Data and Outlier Handling

Mishandling outliers is problematic in fraud detection, because fraud instances may look as outliers, implying that too aggressive filtering may eliminate important signals. To avoid mistakenly removing potential fraud cases, researchers suggest using smarter, more cautious filtering methods. Instead of using fixed rules, setting outlier limits based on expert knowledge of what unusual behaviour might look like (Leevy et al., 2022). Others suggest using tools like Isolation Forests, which help find rare patterns without automatically treating them as errors. Simple techniques like scaling or capping extreme values can also reduce their impact without deleting them entirely (Zareapoor & Shamsolmoali, 2015). In some situations, manually reviewing unusual data points before removing them helps keep important fraud patterns in the dataset. These strategies all support the idea that cleaning data in fraud detection needs to be done carefully, with a strong understanding of what fraud might look like.

3.6 EVALUATION METRICS IN FRAUD DETECTION

Evaluating fraud detection systems/models is a unique task, due to the large class imbalance in most financial datasets, meaning that traditional metrics such as **Accuracy** might be very deceptive. As a result, both researchers and practitioners have shifted to more complex and context-aware evaluation metrics including **Accuracy**, **Recall**, **F1-Score**, and **AUC-ROC**. These metrics provide varied insights into a model's behaviour and trade-offs, and their usage reflects the rising understanding of ethical and effective performance evaluation. A striking trend emerges across fraud detection, healthcare, and cybersecurity: **Accuracy** alone is insufficient when class imbalance is severe and misclassification costs are unequal. **Precision** and **Recall** (or sensitivity and specificity in medical terminology) are the most commonly used metrics in all three domains. For example:

- In fraud detection, **Recall** is prioritised to avoid letting fraud slip through undetected (Chen et al., 2021)
- In cybersecurity, high detection rates are favoured, even if false alarms are tolerated (Bhuyan et al., 2014)
- In healthcare, sensitivity is crucial to catch life-threatening conditions, while positive predictive value helps reduce false diagnoses (Leevy et al., 2022)

This consistency across industries demonstrates a common understanding: when dealing with rare but crucial events, minority class performance and interpretability are more important than sheer **Accuracy**. This also explains the growth in metric combinations (e.g., **F1 + AUC-PR** + cost-based analysis) and the growing interest in explainable AI tools to contextualise these results (Doshi-Velez & Kim, 2017)

3.6.1 Accuracy

Accuracy—the percentage of correctly predicted outcomes—is frequently the first parameter mentioned in machine learning research. However, this statistic is misleading when it comes to detecting fraud as a model that identifies every transaction as non-fraudulent in a dataset containing 99.8% valid instances achieves 99.8% **Accuracy** while detecting no fraud (Owusu-Adjei et al., 2023). The issue is that **Accuracy** assigns equal weight to all groups, despite the fact that false negatives (missed frauds) cost much more than false positives. This problem is not limited to finance; similar trends have been observed in medical diagnostics for uncommon diseases (Leevy et al., 2022), where models appear to be very accurate while failing to detect the very cases for which they were designed. This pattern across domains reveals a key insight: **Accuracy prioritises overall correctness, but ignores class-specific impact.**

3.6.2 Precision And Recall

Precision and **Recall** have become known as critical metrics in fraud detection. **Precision** is the proportion of detected fraud instances that are genuinely fraud, whereas **Recall** measures the proportion of genuine fraud cases that the model properly detects. High **Precision** reduces false positives, which is critical for lowering customer frustration and operational cost. In contrast, high **Recall** means that fewer fraudulent transactions are missed, which is crucial for avoiding financial loss and risks.

Different studies and industries prioritise these metrics differently depending on context. For instance, e-commerce platforms often aim for higher **Precision** to reduce the risk of falsely declining legitimate purchases — which, according to Kount (2022), can lead to customer abandonment and brand damage. On the other hand, banks and insurers are more likely to favour **Recall**, accepting more false positives in order to catch as much fraud as possible, especially in high-risk environments (West & Bhattacharya, 2016). This comparison underlines an essential trend: there is no universally "best" metric, rather trade-offs based on real-world cost structures, so most of the best studies (e.g., Chen et al., 2021; Bhattacharyya et al., 2011) record both **Precision** and **Recall**, and often explore how this change when thresholds are adjusted.

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN}$$

- TP: Fraudulent transactions correctly identified as fraud
- FP: Legitimate transactions incorrectly flagged as fraud
- FN: Fraudulent transactions missed by the model

Figure 1 - Equation

3.6.3 F1-Score

The **F1-Score**, calculated as the mean of **Accuracy** and **Recall**, provides a balanced view when both measures are equally essential. It is especially effective for comparing models with unbalanced datasets since it penalises extreme trade-offs, such as high **Accuracy** with extremely poor **Recall**. However, F1 score also assumes that **Accuracy** and **Recall** are equally valued, which is not always the case in practice.

Critically, depending entirely on **F1 Score** could hide discrepancies in corporate and business priorities. For example, a model with **75% Precision** and **75% Recall** may outperform one with **90% Recall** and **60% Precision**, even if the latter is possibly better at preventing fraud. This is why many financial institutions analyse models beyond F1, using cost-based modelling or domain-specific weighting schemes to reflect the actual cost of errors (Elkan, 2001).

3.6.4 AUC-ROC and Precision–Recall Curves

The **ROC** (Receiver Operating Characteristic) curve compares the true positive rate (**Recall**) to the false positive rate at various thresholds, with the **AUC** signifying the model's ability to prioritise fraud over non-fraud. **AUC-ROC** is popular due to its threshold independence and model generalisation (Provost & Fawcett, 2001). However, in very unbalanced datasets, the **ROC** curve may be overly optimistic, as even numerous false positives produce low false positive rates due to the large number of valid transactions.

Precision-Recall (PR) curves, on the other hand, provide a more complete view of a classifier's performance on minority class. As Davis and Goadrich (2006) demonstrated, **AUC-PR** is more sensitive and trustworthy for detecting rare events, making it a popular choice in fraud detection and healthcare applications. This demonstrates a change in metric choice across domains with high-risk, low-frequency outcomes.

3.6.5 Confusion And Correlation Matrix

The confusion matrix is a tool that visually represents the raw results: true positives, false positives, false negatives, and true negatives. It offers transparency in model performance and is frequently the starting point for calculating all other metrics. In practice, confusion matrices are useful for cost modelling since every error category has a business consequence.

The correlation matrix is another valuable tool during the preprocessing stage for determining the strength and direction of relationships between features. It measures how changes in one variable are related to changes in another. In fraud detection datasets, correlation matrices are frequently used to detect evidence of multicollinearity. High correlations can make models less interpretable and raise the danger of overfitting, particularly in linear models. In contrast, a very low correlation with the target variable may indicate that a trait has little predictive value. West and Bhattacharya (2016) recommend using correlation matrices early in the pipeline to help with feature selection or dimensionality reduction.

3.7 COMPARISON AND EVALUATION OF RELATED WORK / OTHER MODELS

When comparing studies and research in fraud detection, a variety of approaches become apparent, ranging from traditional classifiers to sophisticated ensemble and graph-based techniques. This section investigates how many models have dealt with common issues like as data that is imbalanced, performance evaluation, and real-world deployment, highlighting both effective approaches as well as limitations.

3.7.1 Ensemble Learning with Basic Evaluation

One of the most discussed studies regarding fraud detection used **Logistic Regression**, decision trees, and **Random Forest** classifiers on a credit card dataset. **Random Forest** has the greatest overall performance in terms of **Accuracy** and AUC (Bhattacharyya et al., 2011). However, the study did not account for the skewed class distribution found in fraud datasets, and it relied heavily on **Accuracy**, a metric that might disguise poor fraud detection. While ensemble methods, such as **Random Forests**, outperformed single models, the assessment framework lacked **Recall**, **F1-Score**, and cost-based measures, making it impossible to judge the model's real-world effectiveness. By Incorporating class-sensitive metrics and resampling techniques could have provided a more balanced perspective of model performance.

3.7.2 Undersampling with Balanced Metrics

Dal Pozzolo et al. (2015), on the other hand, concentrated on class imbalances and flaws of traditional evaluation metrics. Their research proposed under sampling the majority class and adjusting probability to improve fraud **Recall**. However, using under sampling limits the amount of valid data available for training, which may have a negative effect on the model's generalisability in production scenarios with diverse transaction behaviours. A hybrid technique, combining under sampling and synthetic oversampling, might have kept better data diversity while maintaining balance.

3.7.3 Graph-Based Modelling for Entity Relationships

Another study proposed a semi-supervised graph-based model that captures relationships between individuals, devices, and transactions in order to detect coordinated fraud. This approach allowed the discovery of hidden patterns, such as fraud rings or linked accounts, which are difficult to identify using transaction-level features alone (Lebichot et al., 2019). While unique, the system needed complicated data structures and additional computing resources, making it better suited to

large organisations with the necessary infrastructure. It also lacked the lightweight efficiency required for real-time fraud protection systems. Using a modular or hybrid deployment approach could have helped to minimise computational stress while increasing scalability.

3.7.4 Feature Filtering and Oversampling in Ensemble Models

Zareapoor and Shamsolmoali (2015) used a bagging ensemble classifier with **SMOTE**-based oversampling and feature selection. Their results revealed considerable improvements in **F1-Score** and fraud **Recall**, indicating that preprocessing plays a major role in performance. They got more stable findings by removing weak characteristics and levelling the data than research that employed raw, unbalanced datasets. However, the work did not investigate **Precision-Recall** trade-offs or threshold tuning, which have subsequently become standard in unbalanced classification problems. Incorporating **AUC-PR** analysis and threshold adjustments would have also improved the assessment strategy.

These studies highlight the evolution of fraud detection research — from focusing on classifier type and **Accuracy** to recognising the critical role of evaluation metrics, data balancing, and preprocessing quality. Zareapoor (2015) focused on feature and balance optimisation, though still lacking advanced metric coverage. Overall, the most effective research combines robust models with carefully engineered data pipelines and context-aware evaluation metrics.

3.8 LIMITATIONS OF CURRENT RESEARCH AND TECHNOLOGY

Despite the increasing sophistication of machine learning for fraud detection, existing research and technology still has significant limits. These flaws have an influence on model performance, generalisability, and real-world impact.

3.8.1 Limited Access to High-Quality, Real-World Datasets

One of the most often reported challenges in fraud detection research is the lack of publicly available, up-to-date datasets. Many studies, like those by Chawla et al. (2002) and Dal Pozzolo et al. (2015), rely on legacy datasets. While these facilitate comparable benchmarking, they often lack diversity in transaction kinds, user behaviour, and new and modern fraud methods. Zhang et al. (2019) emphasise that deep learning models, such as autoencoders, fail to generalise across datasets due to disparities in scale and feature distribution, which is compounded by data scarcity and confidentiality limitations. Without access to a diverse, labelled dataset, it is almost impossible to test models at scale or train systems that reflect the changing nature of fraud.

3.8.2 Imbalance and Label Noise in Data

Even when datasets are available, they often suffer from high class imbalance and label noise, especially in borderline or ambiguous transactions. While approaches such as **SMOTE** (Chawla et al., 2002) and **ADASYN** (He et al., 2008) are widely employed, they fail to address the underlying problem: a lack of relevant positive class instances. According to Zareapoor and Shamsolmoali (2015) and Johnson and Rizzo (2018), synthetic balancing improves **Recall** but can also introduce unrealistic samples or overfit minority patterns.

3.8.3 Lack of Explainability in Complex Models

Advanced techniques like deep neural networks and graph-based models offer promise in terms of **Accuracy**, but this typically comes at the sacrifice of interpretability. Zhang et al. (2019) employed autoencoders for fraud detection, but found that its black-box nature made it difficult to explain individual predictions, posing a challenge for financial institutions that require transparent, auditable

systems. Similarly, Lebichot et al. (2019)'s graph models captured complex fraud rings, but they required domain-specific expertise and lacked clarity in decision-making processes.

3.8.4 Generalisation and Real-Time Adaptation

Many models perform well on static test sets but struggle when confronted with real-time, dynamic and evolving fraud strategies. Fraudsters frequently modify their actions to avoid discovery, resulting in model drift and performance loss. Johnson and Rizzo (2018) emphasise the importance of hybrid or ensemble systems that adapt over time, but also acknowledge that continuing retraining is resource-intensive and rarely addressed in academic work. Furthermore, few studies simulate real-time deployment restrictions including latency, scalability, and response thresholds. Most studies concentrate on predictive power, ignoring how models perform under operational demands – a significant gap between research and reality.

3.9 CONCLUSION

This literature review has explored the key challenges, strategies, and developments in AI-driven fraud detection, highlighting both technical innovations and persistent limitations. Research consistently shows that traditional evaluation metrics like **Accuracy** are inadequate in imbalanced fraud datasets, leading to a shift toward **Precision**, **Recall**, and cost-sensitive evaluation. Techniques such as **SMOTE**, **ADASYN**, and ensemble learning have shown promise, but their success depends on robust preprocessing, feature selection, and balanced metrics.

While advanced models like autoencoders and graph-based detection offer high **Accuracy**, they often sacrifice explainability, limiting their deployment in regulated environments. A common thread across studies is the need for more representative data, greater attention to model interpretability, and systems that adapt to evolving fraud patterns.

These insights directly informed the design of our project — particularly the emphasis on balanced resampling, class-sensitive evaluation, and clear, efficient preprocessing. By focusing our approach in the strengths and gaps identified, our system aims to combine performance with practical reliability.

4 PLESI

As artificial intelligence systems increasingly support sensitive decision-making processes, it is essential to evaluate not only technical performance but also the professional, legal, ethical, and social implications of these systems. This chapter reflects on the **PLESI** dimensions in the context of my AI-driven fraud detection project, which is based on a publicly available **PCA**-transformed credit card dataset.

4.1 PROFESSIONAL ISSUES (P)

Fraud detection systems deployed in financial contexts meet professional requirements for **Accuracy**, accountability, and robustness. According to Breiman (2001), even highly accurate models like **Random Forests** can become unreliable if the data is not pre-processed and analysed properly. In this project, measures like as **Precision**, **Recall**, and **F1-Score** were prioritised over **Accuracy** to represent the true cost of false positives and undetected fraud – a critical choice that aligned with industry expectations. Prioritising the proper metrics is more than simply a technical decision; it is a professional responsibility to align system behaviour with stakeholder demands.

Using a **PCA**-transformed dataset also promotes professional accountability. **PCA** (Principal Component Analysis) decreases data dimensionality while keeping the majority of the variation, enabling effective machine learning without exposing raw sensitive data. According to Jolliffe (2002), **PCA** is particularly effective in anonymising structured datasets while keeping predictive value, making it a suitable tool for professional experimentation that does not violate privacy or compliance rules. Maintaining reproducibility was also an important aspect. Without this, future audits or improvements could be difficult — a common issue in poorly documented systems that hurts professional credibility. Code and procedures were modular and version-controlled, allowing for transparent and collaborative future development, such as deployment or user interface integration.

4.2 LEGAL ISSUES (L)

AI systems in finance must comply with laws and regulations, notably data protection, consent, and algorithmic accountability. Even though our research uses a de-identified dataset, real-world systems are subject to **GDPR** and the **UK Data Protection Act** (2018). These regulations require explainability, the opportunity to appeal judgements, and the correct management of personal information.

PCA contributes to compliance by stripping away directly identifiable information. Li et al. (2018) showed that dimensionality reduction supports privacy-preserving data publishing, helping reduce regulatory exposure. However, relying solely on **PCA** is insufficient — legal responsibility still requires traceable decision-making, audit trails, and explainability mechanisms. If these elements are missing, an organisation could face fines or legal challenges — especially if users are unfairly treated or decisions are contested. A notable example is the **Dutch childcare benefits scandal**, where an automated fraud detection system falsely labelled thousands of families as fraudulent, leading to severe personal and financial consequences. The system’s lack of transparency and appeal mechanisms resulted in mass resignations and public outrage, with legal findings highlighting violations of **human rights and data protection law** (van Eck, 2021).

In any future interface or deployment phase, the system must also consider **data subject rights**, such as access requests, opt-outs, and transparency in decision logic. Failure to address these could result in **regulatory violations and reputational harm**.

4.3 ETHICAL ISSUES (E)

Ethical risks in fraud detection systems include bias, opacity, and user harm. Fraud datasets typically imbalanced, causing models to overlook uncommon fraudulent occurrences or, worse, develop unintended discriminating patterns. Zhang et al. (2019) emphasise that disregarding class imbalance might result in biased conclusions, with genuine users flagged based on incorrect statistical patterns. This is why approaches like **SMOTE** and **ADASYN** were used: they ensure that the model learns from the minority class without amplifying the noise.

Explainability is another major issue. While deep models like autoencoders achieve performance, they frequently function as black boxes (Lipton, 2018), making it impossible to explain why a transaction was flagged. This is unethical, particularly when consumers are refused service or financial access without reason. Models utilised in this study were chosen for their interpretability, which is an important step in establishing confidence and allowing for human oversight.

Furthermore, while **PCA** decreases the likelihood of recognising people, it is not resistant to re-identification. Fredrikson et al. (2015) demonstrated that machine learning models may be reverse-engineered under specific conditions, revealing private information. This is why comprehensive privacy precautions, such as data processing rules and ethical review, are critical. Failure to consider these can risk data breaches, public reaction, and ethical scrutiny.

4.4 SOCIAL ISSUES (S)

Fraud detection systems carry wide-ranging social consequences. These tools shape user trust in digital banking, influence how people engage with financial systems, and raise concerns about algorithmic decision-making in everyday life. When systems misidentify users or operate without explanation, they don't just fail technically — they risk undermining public confidence.

One major difficulty is how society perceives machine learning. While AI is frequently linked with **speed** and **Accuracy**, the public views it with scepticism when used to make automated, high-stakes judgements, particularly those involving money, identity, or access. According to a poll conducted by Smith et al. (2020), more than 60% of respondents were uneasy with algorithms making financial choices without human oversight, claiming a lack of transparency and fairness as primary concerns. This demonstrates that success in fraud detection requires not just finding fraud, but also doing it in a way that consumers see as fair, explainable, and justifiable. This is especially important if a user-facing interface is introduced later in the development cycle. Any interface must be clear, comprehensive, and easily accessible, especially when transactions are flagged. According to Norman (2013), poor feedback or confusing interactions can cause users to lose trust in the system. Transparent messaging, such as explaining why flags are raised and offering response options, helps users feel valued and included rather than alienated.

Another important aspect is the default system behaviour. Westin (2003) discovered that most users do not change system settings or dispute automatic judgements, meaning defaults become the norm. If the default action for a flagged transaction is excessively harsh (for example, immediate blocking), users may feel penalised without reason. If the rules are too lenient, fraud may go undetected. To strike the right balance between safety and trust, socially responsible technologies need carefully calibrated, user-informed defaults. Fairness and inclusivity should also be stressed. Models trained on data from limited user groups may perform poorly when exposed to broader populations, resulting in unintended discrimination or access barriers. According to Johnson and Rizzo (2018), failure to retrain or test models across diverse user segments can reinforce inequality — excluding marginalised users from services they rely on.

A powerful case that illustrates these risks is the **Dutch childcare benefits scandal**, where an algorithm falsely flagged thousands of families — mostly from minority backgrounds — as fraudulent. Many suffered extreme consequences, including wrongful debt collection and loss of benefits. The lack of transparency, appeal routes, and accountability led to **legal rulings that human rights had been violated**, as well as mass resignations and public outcry (van Eck, 2021). This event shaped public discourse on **how damaging unchecked algorithmic decisions can be**, even when technically "accurate."

If such social dimensions are not addressed, fraud detection tools risk becoming **sources of mistrust, exclusion, and social backlash**. Designing systems that are socially responsible means planning not just for what the model predicts, but **how those predictions affect real people** — and how the system **explains, delivers, and supports** those outcomes.

5 METHODOLOGY

This chapter presents the methodological foundation and development process for the AI-driven fraud detection system built as part of this project. In line with academic and professional expectations, this section details not only the specific methods used — such as machine learning models, evaluation metrics, and preprocessing techniques — but also the broader rationale behind each decision. Emphasis is placed on **why these approaches were chosen, how they were applied, and what alternatives were considered**.

A successful machine learning pipeline must also demonstrate reproducibility, ethical consideration, legal compliance, and deployment viability. These principles were embedded throughout the development process using professional tools (e.g., Google Colab, version-controlled notebooks, JSON logs, and sklearn pipelines) to ensure that the project was **both technically sound and methodologically robust**.

5.1 DATASET OVERVIEW

The dataset used in this project is the **PCA-transformed Credit Card Fraud Detection dataset**, originally released by researchers from European financial institutions and hosted publicly on [Kaggle](#). It contains **284,807 anonymised transactions**, of which only **492 are fraudulent** — resulting in a heavy class imbalance of approximately **0.172% fraud cases** and was recorded over a span of **2 days**.

Why this dataset?

- **Real-world relevance:** The dataset is derived from genuine financial transactions, capturing the subtleties of actual user behaviour and fraud patterns. This provides a much more authentic training ground compared to synthetically generated or balanced sets.
- **Public Availability:** Because it is openly available and widely utilised in academic fraud detection research, it enables comparing against previous work while retaining ethical transparency.
- **PCA Transformation:** The features (**V1–V28**) are already processed through Principal Component Analysis to ensure data privacy. This aligns with legal and ethical considerations.

How this dataset was used -

The dataset was directly loaded into **Google Colab**, which is what I used to code and build my model. The **CSV** file was loaded into **Colab** and then I used an **80/20** split with stratified splitting to ensure the class distribution, which makes sure that the rare fraud cases are proportionally represented in both sets — a **critical technique for imbalanced data**. Additionally, all my pre processing such as feature engineering and scaling was only done on the training set, to ensure there was no **data leakage**, which is also essential. Preprocessing pipelines were also constructed and reused through Python functions and saved model states, ensuring reproducibility and efficiency.

The dataset also had limitations, and one of them was the fact that the **PCA-transformed** nature of the dataset means that **original feature context is lost**. As a result, human interpretability of features is limited. Another one was that the extreme class imbalance posed challenges to model training — which informed the choice to use **oversampling techniques** like **SMOTE** and **ADASYN** in later stages.

5.2 DATASET STRUCTURE AND FEATURES

The dataset used in this project consisted of **284,807 anonymised credit card transactions**, each represented by **30 numerical features** and a binary target label (Class) indicating whether the transaction was **fraudulent (1)** or **legitimate (0)**. The features, labelled **V1-V28**, were derived using **Principal Component Analysis (PCA)**. In addition to the **PCA** features, the dataset included two untransformed columns:

- **Time**: Representing the number of seconds elapsed between the first transaction and each subsequent one
- **Amount**: Representing the transaction amount in Euros

And these 2 features were then analysed and used in **preprocessing** and **feature engineering** to improve model learning, which is detailed in Section 4.3.

A correlation matrix was created to investigate potential links between features; while we could see some patterns of linear association, we were unable to properly understand what those associations indicated due to the anonymity of the variables. As a result, the analysis focused mainly on model performance metrics rather than feature-specific explanations.

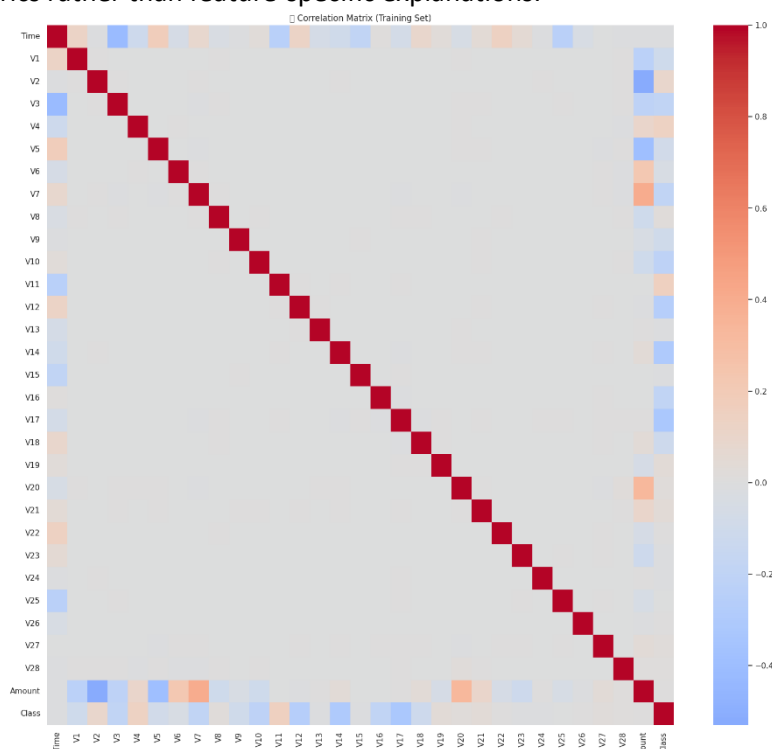


Figure 2 - The correlation matrix

5.3 PREPROCESSING AND FEATURE ENGINEERING

Preprocessing was critical to ensure that the machine learning models could learn patterns successfully and fairly. The dataset initially had two unscaled, non-**PCA**-transformed columns: **Amount** and **Time**. Both were transformed and reorganised using principled approaches to reduce skewness, maintain temporal structure, and allow models to operate optimally. After transformations were complete, the original **Time** and **Amount** columns were dropped to avoid redundancy and ensure a clean feature space.

5.3.1 Box-Cox Transformation of Transaction Amounts

The Amount feature was extremely skewed, with a skewness of 18.19 in the training set. Skewed features can skew model learning, particularly for distance-based or linear models, by giving excessive weight to extreme values. To remedy this, the **Box-Cox** transformation was used after slightly adjusting the variables (to make them strictly positive as required by the method). The transformation dramatically decreased the skew to 0.11, resulting in a nearly-normal distribution.

Unlike **standard scaling** or **min-max methods**, Box-Cox adjusts for skewness and approximates a Gaussian distribution. This makes the feature more usable across different models, especially those that assume normality or are sensitive to feature distribution. This reduced skew improves model generalisation, reduces bias toward high-value transactions, and makes the feature space more evenly distributed. After transformation, the original column was dropped and the final transformed values were retained as **scaled_amount**.

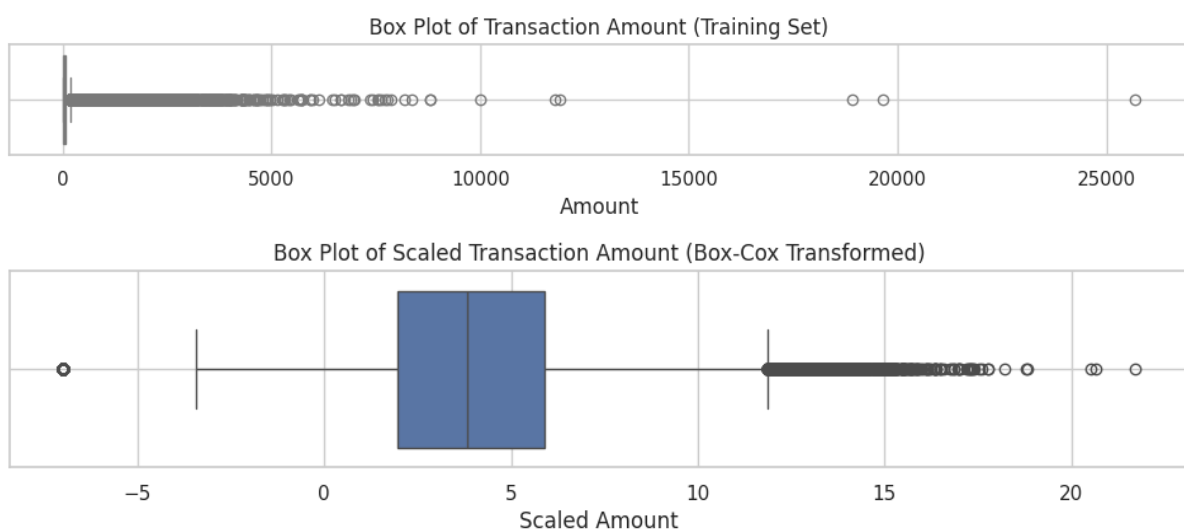


Figure 3 - 2 box plot graphs showing the difference in distribution before and after the transformation

5.3.2 Cyclical Encoding of Time-Based Features

Although the Time column was anonymous and non-descriptive, it was still useful for temporal pattern extraction. The number of seconds from the first transaction was used to construct an Hour feature, computed as:

$$\text{Hour} = (\text{Time} \div 3600) \bmod 24$$

This was then encoded using **sine** and **cosine** transformations to represent time's cyclical structure, ensuring that hour 23 and hour 0 are viewed as adjacent rather than distant values, as the computer would have done if I hadn't done this step. This matters because fraud can show periodic behaviours, such as higher frequency during off-hours. Encoding time cyclically helps models **recognise these patterns**, which linear numeric encodings would miss. This allows models to capture any behavioural changes throughout the day without distorting measures and introducing fake gaps at midnight.

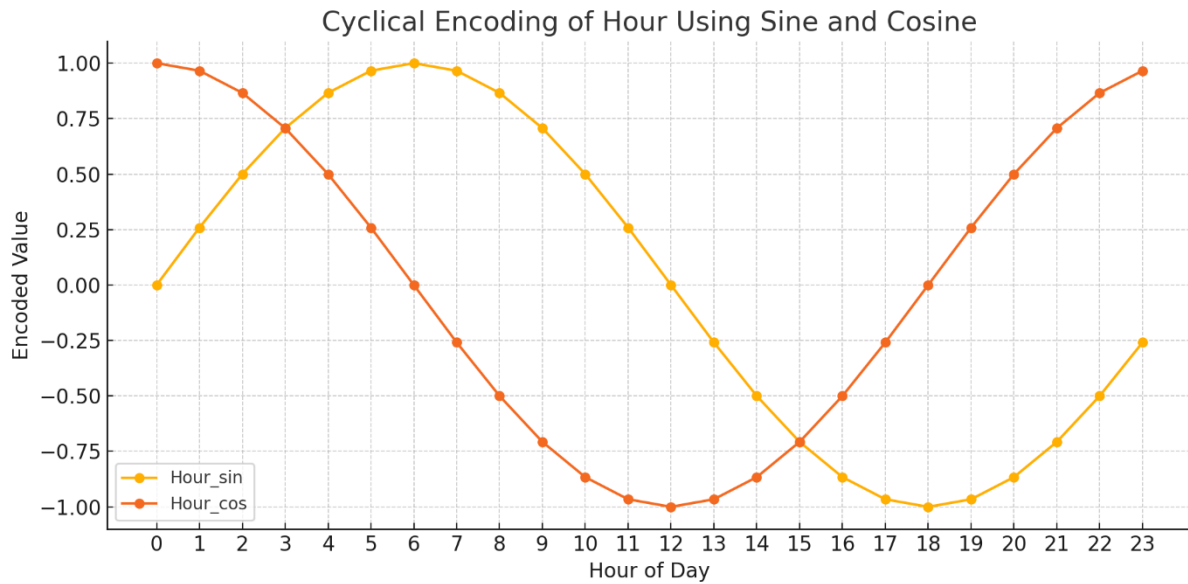


Figure 4 - shows how **Hour_sin** and **Hour_cos** cycle through 24 hours, preserving the fact that hour 23 and hour 0 are adjacent — which is critical for helping machine learning models capture real-world patterns like fraud peaks during certain time window.

5.4 MACHINE LEARNING MODELS AND RESAMPLING TECHNIQUES

This section describes the classification models and resampling techniques used in the project. Each model and approach were chosen based on its theoretical foundation, usefulness with unbalanced datasets, and relevance to current academic and industrial practices. The methodology was applied in a controlled pipeline structure with scikit-learn to ensure repeatability and consistency. All stages were also tested using both **SMOTE** and **ADASYN**-resampled datasets, allowing for comparative analysis.

5.4.1 Machine Learning Models

- **Random Forest**
Random Forest, chosen for its powerful ensemble learning, performs well with non-linear correlations and feature interactions, making it especially helpful in high-dimensional datasets with unknown feature interpretations (such as **PCA**-transformed fraud data).
- **Logistic Regression (LR)**
Logistic Regression provides transparency and simplicity. It serves to contrast more complex models and offers interpretable probability outputs, useful in risk-scoring applications. Although sensitive to imbalance, LR is frequently used in **real-world financial systems** for its transparency and risk-scoring capabilities (Ngai et al., 2011).
- **K-Nearest Neighbours (KNN)**
KNN is sensitive to data distribution and provides insights into how local neighbourhood-based learning reacts to resampling. It acts as a test of how well data preprocessing reshaped feature space. While less scalable, **KNN** has been applied in fraud studies (Bahnsen et al., 2016) to analyse the relationship between resampling and model sensitivity.
- **Support Vector Machine (SVM)**
SVM was chosen due to its strong theoretical performance in high-dimensional spaces and ability to provide robust decision boundaries. While computationally more complex and less interpretable, **SVMs** have proved useful in fraud detection when oversampling and scaling

are used. Chen et al. (2006) demonstrated that **SVM**, when supported by **SMOTE**, delivered superior fraud detection performance across multiple metrics.

These four models were chosen specifically to reflect a blend of simplicity, interpretability, and advanced learning. **Logistic Regression** and **KNN** provided transparency and baseline performance, but **Random Forest** and **SVM** provided better generalisation and complexity handling.

5.4.2 Why SMOTE and ADASYN Were Chosen

Given the class imbalance (**fraud** \approx **0.17%**), oversampling was necessary. Among the options, **SMOTE** and **ADASYN** were selected for their widespread use, compatibility with modern pipelines, and strong performance in fraud-related contexts.

SMOTE (Chawla et al., 2002) creates synthetic samples by interpolating between existing minority class instances. Its core advantage is its ability to **expand the minority class without duplication**, resulting in more balanced training sets. On the other hand, **ADASYN** (He et al., 2008) extends **SMOTE** by focusing generation on **regions near decision boundaries**. This allows models to adapt to “difficult” samples more effectively, often boosting **Recall** in fraud detection, as supported by Mhamdi et al. (2021), who found **ADASYN** to outperform random oversampling in fraudulent transaction detection.

Some alternatives were considered but were dismissed such as **Random Undersampling** which was due to its loss of valuable legitimate transaction data — a critical weakness noted by Van Vlasselaer et al. (2015). Another was **SMOTE-ENN** and **Tomek Links** which were not used for their **added complexity** and sensitivity to parameter tuning, which can introduce noise and hinder reproducibility.

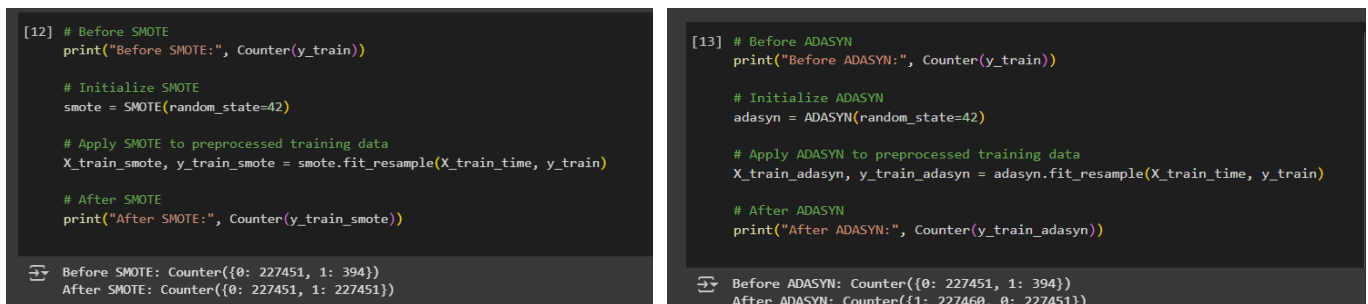


Figure 4 - Class distribution before and after applying **SMOTE** (right) and **ADASYN** (left). **SMOTE** creates a perfectly equal class distribution, while **ADASYN** adaptively generates slightly more fraud samples in boundary regions to enhance learning complexity.

5.5 EVALUATION METRICS USED

To assess model performance, a multi-metric assessment approach was used. While Section 2.6 described the academic reasoning for important measures in fraud detection, this section focusses on how and why these metrics were chosen and applied especially to this project's context. The primary objective of this project was to **detect as many fraudulent transactions as possible**, without overwhelming the system with false positives. So, **Recall was slightly prioritised**, particularly for the minority (fraud) class, supported by complementary metrics to ensure overall balance and usability.

- Recall (Sensitivity)

False positives (incorrectly reporting legal transactions) and false negatives (missing true fraud) both have implications in fraud detection; however, the cost of a false negative is often more significant. A missed fraud can cause immediate financial loss, customer suffering, and legal

liability; hence it is vital to maximise True Positives. Therefore, the evaluation process highlighted **Recall** as a means of ensuring the system captures as many fraudulent cases as possible — while staying mindful of the trade-off with **Precision**. This balance matches real-world experience in financial fraud detection, where over-alerting is risky but under-detection may be even more damaging. Throughout, **Recall** served as a guiding parameter for comparing models and resampling strategies.

- **Precision**

While high **Recall** is essential, excessive false positives can cause operational burdens, such as unnecessary transaction blocks or user friction. **Precision** was included to monitor how well each model isolated actual frauds from the legitimate transactions. This metric helped ensure that the improved detection rate (via **Recall**) did not come at an unsustainable cost.

- **F1 Score**

The **F1 Score** was calculated as a balanced indication that considers both **Precision** and **Recall**. This was very beneficial for adjusting and ranking models with trade-offs, notably between **SMOTE** and **ADASYN** techniques. F1 assisted in determining which models provided the optimal balance of fraud detection and false alarm prevention. This statistic has been used widely in fraud detection research, like Choi et al. (2020) who emphasised the use of F1 in evaluating imbalanced classification models where neither **Precision** nor **Recall** alone tells the full story — particularly in transaction-based fraud.

- **AUC-ROC**

The Area Under the **ROC** Curve provides an expanded view of each model's capacity to distinguish between fraudulent and genuine transactions at all levels. While **AUC-ROC** might occasionally over-represent the majority class, it is nevertheless an effective baseline indication of model discrimination. Research by Whitrow et al. (2009) demonstrated the utility of **ROC** analysis in comparing fraud classifiers, noting that **AUC** remains relevant when used in conjunction with class-sensitive metrics — a strategy also applied in this project.

- **Precision-Recall AUC (PR AUC)**

Sahin et al. (2013) showed that **PR AUC** outperforms **ROC AUC** in revealing the true performance of classifiers under fraud conditions, and recommended it as a best practice for such domains. Given the class imbalance in this dataset, **Precision-Recall AUC (PR AUC)** was considered to be more reliable than **ROC-AUC**. **PR AUC** measures how well the model performs on the minority class across thresholds, making it especially useful for high-risk applications such as financial fraud detection.

5.6 EXPERIMENTAL SETUP AND WORKFLOW

This section outlines the experimental framework used to train and evaluate models in a controlled, reproducible environment. All development took place in **Google Colab**, using modular pipelines and **JSON**-based result logging for repeatability and structured analysis.

5.6.1 Train Test Split and Baselines

An **80/20 train-test** split was applied to the dataset, with stratification to preserve the fraud ratio in both sets. This ensured consistent evaluation and mitigated sampling bias, as recommended in fraud-specific learning contexts (Fernández et al., 2018). Another thing was that 2 baseline of the 4 models were used to assess the added value of preprocessing and feature engineering.

- **Baseline 1:** Trained on the raw, imbalanced dataset without scaling or transformation.
- **Baseline 2:** Trained on the pre-processed dataset (scaling and feature engineering applied), but still unbalanced.

These baselines served as control points to highlight performance changes at each pipeline stage. Results from both confirmed that without sampling, models failed to detect fraud reliably — showing the importance of the baseline steps.

5.6.2 Metric Logging & Reproducibility

Google Colab was used as it provided accessible cloud execution, but often required saving and loading outputs to avoid session losses. Tools like scikit-learn, imbalanced-learn, pandas, NumPy, matplotlib, seaborn were used and all results — including performance scores and confusion matrices — were logged to **JSON** files. Pipelines and trained models were saved using joblib, enabling quick reloads and ensuring consistent evaluation even across sessions.

5.6.3 Critical Workflow Reflection

Some advanced techniques (e.g. nested CV, ensemble stacking) were intentionally excluded to preserve transparency and reproducibility — especially important in a student research context. Basic cross-validation was used early for baseline checks. Any tuning was deferred until the top-performing pipeline was identified — an iterative strategy aligned with practical ML workflows (Zhang et al., 2021). Colab's constraints further influenced this decision, but overall, the workflow allowed for robust, fair, and well-documented comparisons across all model and resampling combinations.

5.7 SUMMARY OF METHODOLOGICAL CHOICES

The methodology in this project was carefully constructed to align with the objective and each stage of the pipeline — from preprocessing to model selection — was guided by this:

- **Data preprocessing** focused on transforming and engineer features (Amount, Time) in such a way that they remained fair and interpretable despite the dataset's **PCA** anonymisation.
- **Resampling strategies (SMOTE and ADASYN)** were deliberately chosen for their proven ability to address extreme class imbalance without discarding valuable data.
- Baseline experiments were used to see progress and for comparison
- **(Logistic Regression, KNN)** and advanced **(Random Forest, SVM)** approaches across multiple sampling pipelines.
- Metric choice was driven by domain-specific risks and research.

These decisions were made by being informed from academic literature, practical constraints (e.g. Colab environment), and the specific challenges of fraud detection — such as data sparsity, ethical handling, and performance trade-offs. Altogether, the methodology reflects an approach that is **data-conscious**, **risk-aware**, and **results-driven**, laying a solid foundation for critical review and model selection.

6 EVALUATION AND DISCUSSION

Before choosing on a final model, it was essential to assess how each stage of the pipeline contributed to performance, dependability, and alignment with the project goals. This chapter provides an organised review of the models and resampling procedures employed, accompanied by test results and proof. It also discusses critical learning experiences, limitations, and the ethical and societal implications of fraud detection systems. The evaluation extends beyond performance ratings to provide insight into the actions, abilities, and judgements that influenced the final outcome.

6.1 EVALUATION STRATEGY AND BASELINE ANALYSIS

The purpose of the evaluation was to determine the practical and technical effectiveness of each model–resampling combination in detecting fraudulent transactions under real-world conditions. The evaluation went beyond measuring raw **Accuracy**, instead focusing on metrics that reflect the cost-sensitive nature of fraud detection — particularly **Recall**, **F1-Score**, and **PR AUC**, which all are agreed upon around the sector as the 'most important' - but only to a certain extent.

6.1.1 Purpose of Evaluation

Due to the highly imbalanced nature of the dataset, traditional metrics like **Accuracy** were considered misleading. In contrast, **Recall** was prioritised to assess the model's ability to detect true fraud cases, **F1-Score** to understand the trade-off between false positives and false negatives, and **PR AUC** for its reliability in imbalanced scenarios.

This evaluation process mirrored the real expectations of a financial fraud detection system, where failing to catch fraudulent activity (**false negatives**) can have much more damaging consequences than falsely flagging a legitimate transaction (**false positives**). Therefore, the strategy was designed not only to compare models but to check their readiness for being deployed.

6.1.2 Establishing the Baselines

The 2 baselines provided reference points to measure how much performance improved through preprocessing and oversampling. The baselines clearly demonstrated that while basic preprocessing helped slightly, it was not enough to address the severe imbalance.

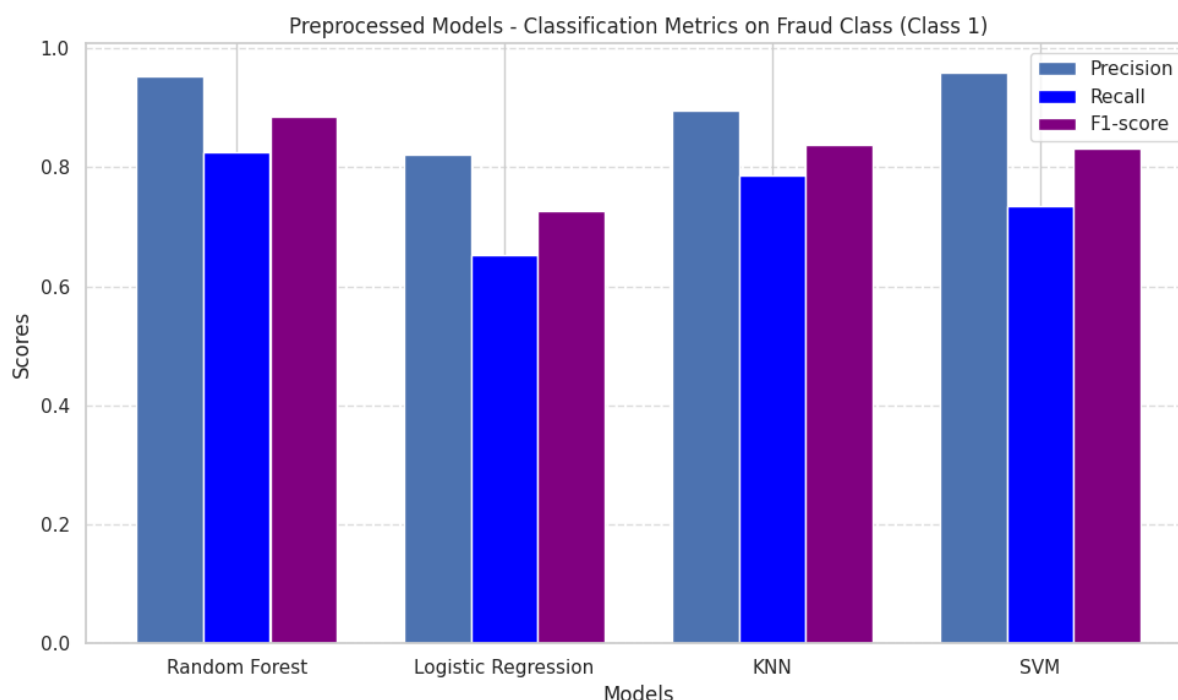


Figure 5 - Bar chart comparing **Precision**, **Recall**, and **F1-Score** for each model (**Random Forest**, **Logistic Regression**, **KNN**, and **SVM**) trained on the pre-processed but imbalanced dataset. Metrics reflect performance on the fraud class (Class 1).

This graph shows that **Random Forest** beats the other models in all three criteria for the minority fraud class, even when no balancing technique is used. In contrast, **Logistic Regression** performs the poorest, with particularly low **Recall**, indicating that it fails to detect many fraud cases. **KNN** and **SVM** obtain reasonably good **Accuracy**, although their **Recall** and **F1 scores** are lower than **Random**

Forest. The findings show that, even with preprocessing, class imbalance substantially restricts model effectiveness—particularly in **Recall**—and emphasises the need of targeted resampling techniques such as **SMOTE** or **ADASYN**.

6.2 EVALUATION OF OVERSAMPLING METHODS

Given the dataset's significant class imbalance, the decision to apply oversampling techniques was essential. **SMOTE** and **ADASYN** were chosen because they are well known for creating synthetic samples that try to balance minority groups. However, beyond implementation, determining how each strategy influenced the model learning and fraud class identification was crucial.

6.2.1 Visualising SMOTE and ADASYN

To compare, **t-SNE** visualisations were generated to display how synthetic fraud points were distributed. **SMOTE** formed dense central clusters, which helped models learn stable boundaries and this was shown in the improved metrics. In contrast, **ADASYN** produced boundary-distributed samples, which introduced greater variety but also more noise, which was evident in the drop in **Precision** for several models.

This revealed how each strategy affected learning quality in fraud classification. **SMOTE's** clustered outputs had more generalisation and consistency, making it more appropriate for tree-based models that rely on separability. **ADASYN's** boundary-level variation may assist deeper or more complicated models, however for the algorithms used here, it resulted in less consistent results. These patterns show that, while both techniques increase **Recall**, **SMOTE**-based pipelines may perform better in real-world applications that need decision clarity and stability.

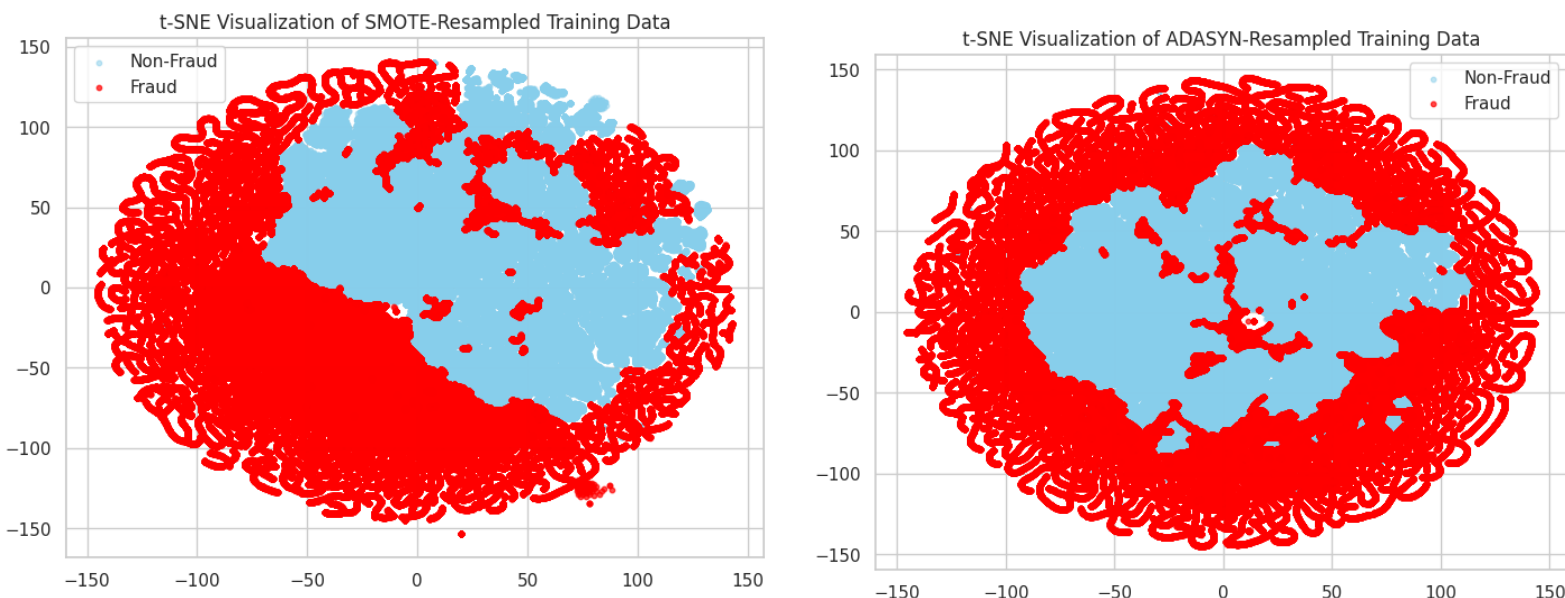


Figure 6 - **t-SNE** visualisations of **SMOTE** (left) and **ADASYN** (right) resampled training data, highlighting the distribution of synthetic fraud samples. **SMOTE** produces more centralised clusters, while **ADASYN** introduces boundary-level dispersion.

6.3 MODEL PERFORMANCE EVALUATION

Following the training and testing of all the selected models, I focused on carefully assessing their performance using classification metrics. The aim was to determine whether algorithm and resampling technique (**SMOTE** or **ADASYN**) produced the best balanced and dependable results in

fraud detection. This comparison directly contributed to the project's goal of developing a comprehensive and trustworthy AI-based fraud detection system.

6.3.1 Comparison Summary Table and Observations

Model Variant	Sampling	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)	Accuracy	PR AUC
Random Forest	SMOTE	0.8989	0.8163	0.8556	0.9995	0.8684
Logistic Regression	SMOTE	0.0542	0.9184	0.1023	0.9723	0.7622
KNN	SMOTE	0.4328	0.8878	0.5819	0.9978	0.7570
SVM	SMOTE	0.0895	0.8571	0.1620	0.9847	0.5752
Random Forest	ADASYN	0.8966	0.7959	0.8432	0.9995	0.8606
Logistic Regression	ADASYN	0.0178	0.9286	0.0349	0.9117	0.7670
KNN	ADASYN	0.4350	0.8878	0.5839	0.9978	0.7618
SVM	ADASYN	0.0914	0.7551	0.1621	0.9867	0.4090

Figure 7 - The table above presents the key performance metrics for all 8 evaluated models — each trained on the pre-processed dataset using either **SMOTE** or **ADASYN**

- **Random Forest models**, whether trained with **SMOTE** or **ADASYN**, consistently outperformed all others across most metrics. **SMOTE** yielded a slightly higher **PR AUC**, while **ADASYN** offered a marginally better balance of **F1** and **Recall**.
- **Logistic Regression** produced high **Recall** but had extremely low **Precision** on both **SMOTE** and **ADASYN**, resulting in a disproportionate number of false positives. This made it less reliable despite seemingly high fraud-class sensitivity.
- **KNN** demonstrated moderate performance under both resampling techniques. While **Recall** was commendable (~88%), its lower **Precision** impacted the **F1-Score** significantly. This suggests **KNN** tended to overfit toward predicting the positive class.
- **SVM** showed consistent underperformance across both sampling methods. Particularly with **ADASYN**, its **PR AUC** dropped to **0.409**, suggesting poor separability of fraud cases.

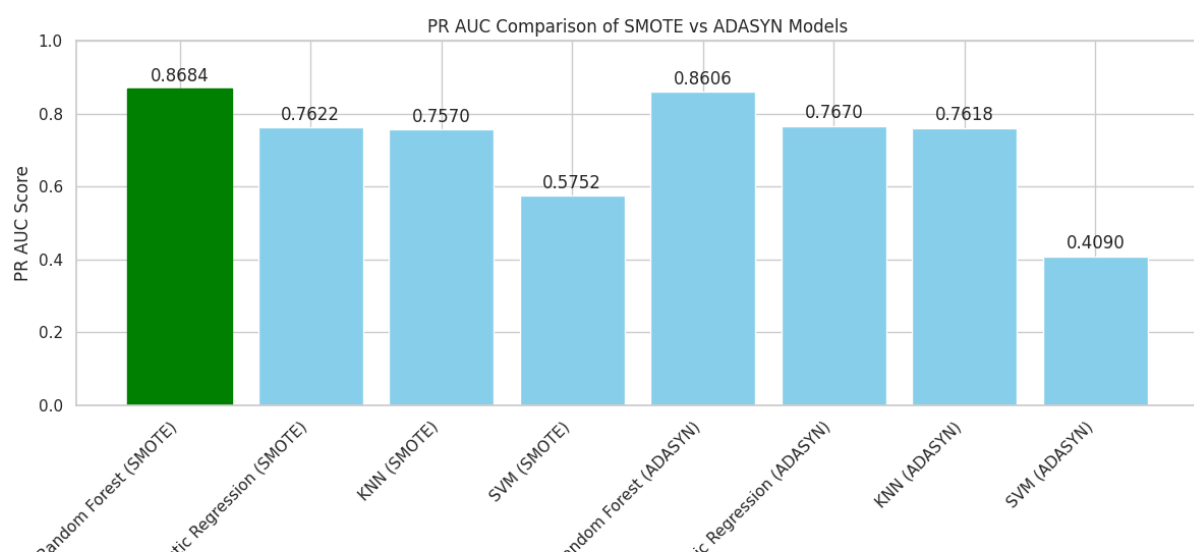


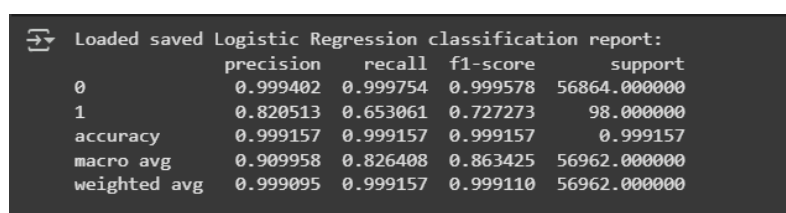
Figure 8 - The **PR AUC** metric, which accounts for class imbalance, was especially useful in distinguishing models that might otherwise appear similar under **Accuracy** or macro-averaged **F1-Scores**. This is crucial as we can see the disparity between **AUC-ROC** which was consistently high even for low **PR AUC** scores.

6.4 TESTING PROCESS AND TOOLS

A thorough testing method was used throughout the project to assure repeatability, fairness, and practical reliability, which is especially important in high-risk domains like fraud detection. This testing procedure looked at class-specific performance, generalisation ability, and session stability. Results were traceable, version-controlled, and recorded using automated logs and metric outputs, as is standard procedure.

6.4.1 Reproducibility and Logging

To maintain reproducibility — a core principle in both research and industry-grade data science — all classification results, confusion matrices, and evaluation metrics were saved in structured **JSON** files. All trained models were saved using joblib, ensuring exact configurations could be instantiated without retraining, saving time. This approach aligns with professional practices in model deployment, where consistent outputs under identical inputs are a non-negotiable requirement (Sculley et al., 2015).



```
Loaded saved Logistic Regression classification report:
      precision    recall  f1-score   support
0         0.999402    0.999754    0.999578    56864.000000
1         0.820513    0.653061    0.727273     98.000000
accuracy         0.999157    0.999157    0.999157     0.999157
macro avg         0.909958    0.826408    0.863425    56962.000000
weighted avg         0.999095    0.999157    0.999110    56962.000000
```

Figure 7 - Example of a saved JSON output from my notebook

6.4.2 Testing Scope

All models were tested against a strictly held-out test set (20% of total data), which was never seen during training or validation, ensuring no data leakage. Evaluation involved:

- **Confusion Matrix:** To assess false positives (FP), false negatives (FN), true positives (TP), and true negatives (TN) for detailed class-wise insight.
- **Classification Reports:** Including **Precision**, **Recall**, **F1-Score**, and support for each class (fraud and non-fraud), across all models and oversampling strategies.
- **PR AUC and AUC-ROC Metrics:** Especially focused on the minority class (fraud), as discussed in earlier sections
- **Baseline Comparison:** Tests were also run on both raw and pre-processed unbalanced datasets to benchmark the improvements

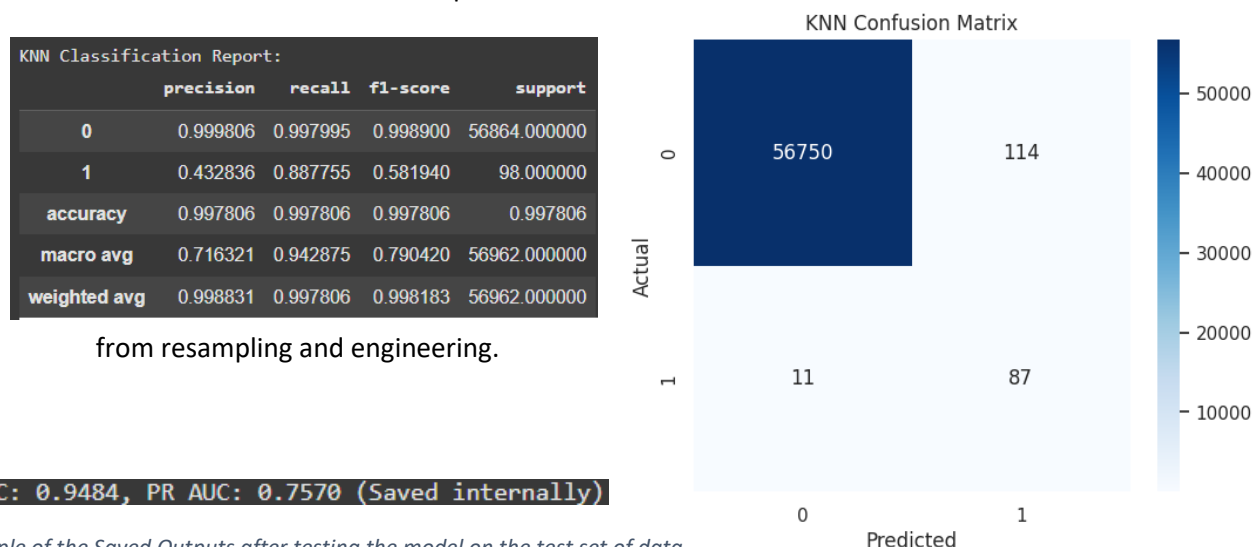


Figure 8 - Example of the Saved Outputs after testing the model on the test set of data

6.5 LIMITATIONS OF THE APPROACH

While the project achieved strong performance, several limitations must be acknowledged. These limitations influenced both the scope and depth of the analysis, particularly in terms of explainability, infrastructure, and statistical confidence.

6.5.1 Dataset and Feature Constraints

A key limitation was the **PCA**-transformed dataset. While this transformation was essential for ethical and legal reasons, it significantly constrained domain-specific interpretation. Although a correlation matrix was used to observe some linear associations between features (e.g., weak or moderate patterns with 'Amount'), the lack of context made it impossible to link features back to real-world transaction behaviour.

This limited our ability to perform domain-driven feature engineering — an approach that often enhances fraud detection by incorporating merchant category, transaction type, or user history (Pozzolo et al., 2015). Without such interpretability, the model became more of a "black box", which in financial systems could be problematic for compliance and auditing. As Ribeiro et al. (2016) emphasise, the lack of interpretability in high-stakes domains like fraud detection can reduce trust and hinder adoption — even when **Accuracy** is high.

6.5.2 Sampling Risks and Infrastructure Limitations

While **SMOTE** and **ADASYN** effectively balanced the dataset, both have potential drawbacks. Since these methods generate synthetic samples based on nearest neighbours, they risk introducing noise, particularly near decision boundaries or in highly sparse regions. This can lead to overfitting or inflated **Recall**, which may not generalise to unseen data. Although **t-SNE** visualisation (see Figure 7) confirmed generally good clustering, some overlapping regions hinted at potential ambiguity introduced by the sampling process. **SMOTE**-based approaches, while effective, can “misrepresent the minority class distribution” in noisy or high-dimensional spaces (Blagus & Lusa, 2013), which may partially apply to our **PCA** dataset.

Additionally, due to infrastructure limits — namely the session time limits on Google Colab — we were unable to conduct more advanced techniques such as nested cross-validation or exhaustive hyperparameter searches. This limited our ability to assess model variance or detect overfitting across multiple data splits and a longer-running environment would have enabled a more comprehensive validation loop using **Gridsearch** or **Randomsearch** etc.

6.6 SUMMARY OF KEY FINDINGS

This part compiles the key insights gathered during the evaluation, providing an overview of how the models performed and what was achieved. It reiterates the project's important findings, such as the final model's capabilities and the larger implications for fraud detection systems. A full set of evaluation visuals, metrics, and confusion matrices for each model and configuration is available in the appendices.

6.6.1 Final Model Performance Overview

After analysing the performance of all trained models, the **Random Forest** classifier paired with **SMOTE** proved to be the most effective and reliable configuration. Also, one key decision point at this stage involved deciding whether hyperparameter tuning was necessary. While hyperparameter optimisation is standard practice in industry for refining model performance (Smith & Martinez, 2021), exploratory tuning trials for **Random Forest** resulted in only **minor increases in Recall**, which were offset by **significant rises in training time and complexity**. Considering this, the process was

deliberately deprioritised to maintain an efficient workflow and reproducibility — a trade-off that reflects practical decision-making often seen in professional machine learning pipelines (Brownlee, 2023). This balance between performance and engineering efficiency further strengthens the model’s applicability in real-world fraud detection contexts.

The comparative evaluation across all models also revealed key performance patterns:

- **Random Forest** demonstrated the most consistent results on both **SMOTE** and **ADASYN**, with **SMOTE** producing a slightly higher **PR AUC** and greater control over minority class misclassifications, but still wasn’t perfect.
- **Logistic Regression** excelled in **Recall** but at the expense of extremely low **Precision** under both resampling methods, making it prone to flagging too many false positives.
- **KNN**, while maintaining reasonable **Recall**, struggled with **Precision** due to its sensitivity to overlapping classes.
- **SVM** underperformed, particularly with **ADASYN**, where its **PR AUC dropped to 0.409**, likely due to poor class boundary separability.
- The raw, imbalanced models performed poorly on minority class **Recall**, confirming the need for **resampling**.
- **ADASYN** yielded competitive results in some cases (notably in **Logistic Regression**), but **SMOTE** generally offered more stable and generalisable boundaries, as confirmed by the **t-SNE** visualisations (Figure 7).
- **SVM** and **Logistic Regression** both showed strong performance on the majority class but lacked consistency in detecting fraud, especially under **ADASYN**.
- **Precision-Recall** trade-offs became a central aspect of comparison, further emphasising the importance of metric choice in imbalanced domains.

A complete breakdown of these results, including **Precision-Recall** trade-offs, confusion matrices, and metric-based comparisons, is included in the Appendices. These results clearly justify the selection of **Random Forest + SMOTE** as the final model — offering a high-performing and stable solution aligned with the project’s goals.

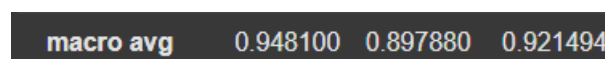


Figure 9 - macro average for **SMOTE + Random Forest**

6.7 WEBPAGE AND MODEL REVIEW

As part of making the fraud detection system more practical and accessible, I decided to create a simple webpage that connects directly to the final trained model. The aim was not just to build a standalone model, but to demonstrate how it could be used in a real-world setting where users can input transactions and receive immediate fraud predictions. This aligns with the original goals of the project, where developing an AI-driven fraud detection product was just as important as the technical **Accuracy** of the model. The following sections review the deployed backend API, the frontend website, and the overall integration approach.

6.7.1 Backend API And Model Integration

The backend was developed using **Flask**, and hosted separately from the frontend on **Render.com**, which spins down during times of inactivity, to ensure modularity and scalability. All preprocessing steps — such as **Box-Cox transformation for Amount (using a saved lambda)** and **cyclical encoding for Time** — are handled server-side automatically when a transaction is submitted.

The final **Random Forest + SMOTE** model, trained and saved earlier, was deployed through the API. Predictions return two values:

- A **fraud probability** (a floating-point score)
- A **binary classification** (fraud or not fraud)

A custom threshold of **0.45** was used based on earlier evaluation findings to better balance false positives and negatives. Strict input validation ensures the **API** only accepts transactions with the required fields (**V1–V28**, Amount, and Time). This deployment approach mirrors how fraud detection models are integrated into production systems in the finance industry, separating model logic from user interaction and making future scaling much easier.

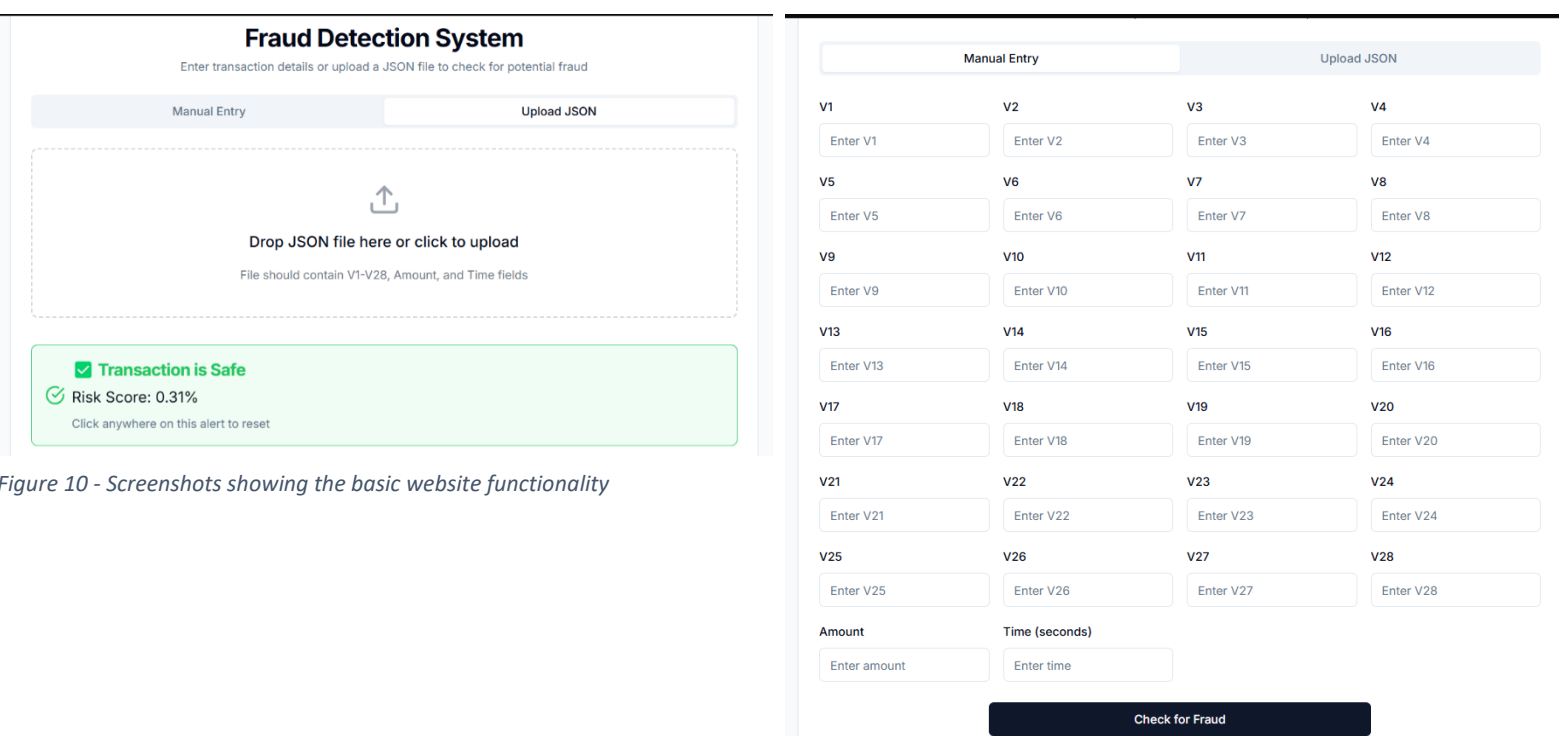


Figure 10 - Screenshots showing the basic website functionality

6.7.2 Overall Deployment Architecture

The system is split into two independent services:

- The **frontend client** interacts with users and gathers input.
- The **backend API** preprocesses input and serves predictions securely.

Both services are deployed separately on **Render.com**, allowing updates or scaling of one without affecting the other. The backend **API** uses **Gunicorn** to serve multiple requests efficiently.

This separation follows modern best practices in cloud-based machine learning products, offering better security (the model is never exposed directly), flexibility, and future extensibility (e.g., adding user accounts or transaction logs later).

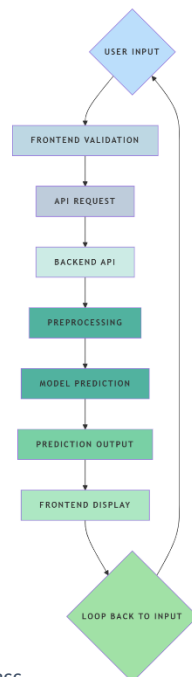


Figure 11 - Flowchart visualising the process

6.7.3 Limitations and Future work

While the website successfully demonstrates a working fraud detection system, there are areas for future enhancement:

- **API** winds down during times of inactivity, meaning usages after a while may take longer to output.
- **Batch processing** is currently limited to uploading JSON files; real-time large-scale streaming (e.g., for banks) would require parallel processing setups.
- **Security** features such as API authentication, rate limiting, or encryption could be added to make it production-grade.
- **No Feedback or Feedback Loop** so users cannot flag incorrect predictions, nor is there any feedback mechanism to improve the model over time. Including a retraining loop based on real-world false positives/negatives would improve long-term performance.
- **No Data storage or Logging** of the predictions, which limits traceability and auditing. In a real deployment, logging inputs, timestamps, and outcomes securely would be essential for compliance and tracking anomalies.

Despite these limitations, the current implementation meets the core objectives of accessibility, usability, and practicality for fraud detection.

6.8 EVALUATION AND REVIEW SUMMARY

This chapter critically evaluated the model development lifecycle, including training, testing, performance comparison, and practical deployment of the final fraud detection system. Through a combination of classification reports, confusion matrices and more, a clear and measurable assessment of model effectiveness was achieved.

Key strengths were seen from the **Random Forest** models, especially when combined with **SMOTE**. This pairing delivered consistently high performance across **Recall**, **Precision**, **F1-Score**, and **PR AUC**, demonstrating its reliability in identifying fraudulent transactions. The significance of these metrics—particularly **Recall**—was continually prioritised due to the disproportionately high cost of false negatives in fraud detection, as discussed in Section 4.5.

Evaluation extended beyond just metric comparison. Real-world simulation through a full deployment pipeline, including API integration and a user-facing interface, allowed the model to be tested under practical conditions. This hands-on implementation revealed that the model generalised well beyond the training set, with limited overfitting and reasonable robustness. Limitations such as model interpretability, slight **Recall–Precision** trade-offs, and minimal post-tuning gains were critically reviewed in Section 5.6. These insights reinforce the fact that even high-performing models require ongoing refinement, testing, and context-specific adjustments in production environments. Additional screenshots, code samples, and interface tests are available in the appendices and these ensure a full audit trail of testing and performance verification, aligning with academic and industry standards of reproducibility.

In conclusion, this chapter highlighted how the chosen approach—underpinned by systematic testing and evaluation—met the project’s goal of developing a dependable, scalable, and accurate AI-driven fraud detection tool.

7 DISCUSSION, REVIEW AND MOVING FORWARD

The idea of an operational AI-driven fraud detection system required not only technical implementation, but also thorough evaluation and decision-making at each level. This chapter takes a step back and evaluates both the process and the project. The process featured difficulties and modifications, such as time management issues and etc, but it resulted in a solution and a better knowledge of AI implementation in financial cybersecurity. The parts that follow look at what was learnt, what skills were gained, and how the project grew, ending with a final look at potential future additions.

7.1 REVIEW OF AIMS AND PROBLEMS

The central aim of this project was to develop an AI-driven fraud detection system capable of accurately identifying fraudulent financial transactions, while remaining practical enough to deploy within a real-world context. As originally outlined in **Section 1.2**, the project focused not only on predictive performance, but also on robustness, fairness, and interpretability — especially in the face of extreme class imbalance.

- **Real-world dataset analysis** was successfully performed using the Kaggle credit card fraud dataset. Patterns in time, amount, and anonymised features were studied through **EDA**, guiding key preprocessing steps.
- **Data preprocessing** was executed thoroughly. This included scaling, Box-Cox transformation, cyclic time encoding, and most crucially, handling imbalance via **SMOTE** and **ADASYN** — all of which directly contributed to better model performance.
- **Model comparison** was a core part of the methodology. Four classifiers (**Random Forest**, **Logistic Regression**, **KNN**, **SVM**) were evaluated using both resampling methods, which provided justification for final model selection.

- **Evaluation using appropriate metrics** was a strength of the project. **Precision, Recall, F1-Score, AUC-ROC**, and **PR AUC** were all considered, with emphasis placed on **Recall** due to the high cost of false negatives in fraud detection.
- **Rigorous testing** was conducted, including multiple baselines (on raw and pre-processed data), extensive metric-based comparison, and visualisation, these helped validate both performance and generalisability.
- **Interface development** was achieved through a functional website deployment, allowing users to input transaction data and receive live predictions from the model via a backend API.

Overall, the aims were largely met. The project produced a reliable and high-performing fraud detection model, supported by sound experimentation and evaluated using strong industry-aligned criteria. The project remains adaptable for future extension in various directions.

7.2 LEARNING AND INSIGHTS

This project presented valuable learning opportunities beyond implementation. One key insight was the realisation that **Accuracy** is misleading in imbalanced classification. This shifted my focus towards more meaningful metrics like **Recall, Precision, F1-Score** and **PR AUC**, especially given the cost of misclassifying fraud cases.

Another important takeaway came from comparing **SMOTE** and **ADASYN**. While both techniques are widely used, **t-SNE** visualisations (figure 7) revealed their different sampling behaviours — with **SMOTE** concentrating on the centre of the minority class and **ADASYN** targeting the decision boundary. This explained performance variations and highlighted the value of visual evaluation. Moreover, this project enhanced my understanding of the trade-offs between model complexity, interpretability, and practicality. For instance, while more complex solutions like neural networks could have been explored, the decision to optimise a robust ensemble method (**Random Forest**) produced high-performing results with explainability and speed — a valuable trade-off in applied machine learning.

7.3 SKILLS AND COMPETENCIES DEVELOPED

Throughout this project, I developed a range of technical and professional skills. Most notably, my competence in data science pipelines improved significantly. I gained hands-on experience with real-world data preprocessing, from handling missing values and outliers to performing scaling, and feature engineering — all within the context of highly imbalanced datasets.

I also strengthened my knowledge of classification algorithms by implementing and critically evaluating **Random Forests, Logistic Regression, SVMs**, and **KNN** models. Through comparative experimentation with **SMOTE** and **ADASYN**, I deepened my understanding of resampling strategies and their impact on model performance.

From a development standpoint, deploying the final model via a full-stack interface expanded my proficiency in backend integration and **API** usage. I worked with tools like Flask (for serving the model), **JSON** input/output formats, and Render.com for deployment. Creating a clean and user-friendly frontend using modern frameworks taught me the importance of user experience and error handling when moving from research to usable applications. Ensuring that data preprocessing, model prediction, and user interaction are fully integrated added practical challenges but also strengthened my understanding of model-serving workflows and user experience design.

Additionally, documenting my process and writing a structured report improved my ability to communicate technical work clearly and professionally.

7.4 SELF-CRITICISM AND REFLECTION

Although the project successfully met its objectives, there were aspects that could have been improved upon through better planning and self-awareness.

- **Initial Overload & Time Management:** I initially underestimated how time-consuming the data preprocessing and resampling stages would be. This created a knock-on effect that delayed the start of baseline and resampled model evaluation. I managed to by working extended hours during key stages, but this highlighted the need for more realistic early planning.
- **Supervision & Communication:** One key area of self-reflection is my limited engagement with my supervisor during earlier stages. While I maintained some communication, more frequent check-ins could have accelerated feedback loops, avoided missteps and added extra clarity to parts of the project.
- **Model Breadth vs. Depth:** While I opted to focus on four strong models (**Random Forest, Logistic Regression, SVM, KNN**), I recognise that I could have experimented with more advanced ensemble methods like **XGBoost** or **LightGBM**. However, my decision was based on time, complexity, and the diminishing returns seen in preliminary tuning attempts. This trade-off was conscious but may have limited the discovery of alternative high-performing models.
- **Hyperparameter Tuning:** Section 5.5 outlines the limited but structured tuning that was performed. In hindsight, I could have tested more combinations using parallelised tools. However, tuning was deprioritised after observing only marginal performance gains at high computational cost.
- **Product Evaluation Limitations:** While the web interface was manually tested across a range of inputs and edge cases, more formal user evaluation could have strengthened confidence in the product's practical use.

Overall, this project highlighted the importance of proactive time management, iterative communication, and structured validation. These reflections have helped me become more self-aware in my work, and better prepared to handle larger-scale technical projects in the future.

7.5 FUTURE PLANS

Looking ahead, this project could be taken significantly further, in terms of technical capability and practical deployment. While the model and webpage met their intended goals, the evaluation process surfaced several areas where targeted improvements could enhance the project.

One clear direction would be the implementation of a real-time fraud detection pipeline, integrated directly into a transaction processing system. Rather than relying on static inputs or uploaded batches, this would involve overlaying the model onto a live transaction portal, enabling dynamic decision-making as payments are processed. Transactions could be automatically flagged, paused, blocked and logged if the model identifies suspicious patterns. This kind of feedback loop would also be more in line with how fraud detection systems operate in production. This would make the tool more applicable to business-facing use cases.

From a broader project perspective, future steps might include **partnering with sandboxed financial APIs**, to test the system in more realistic transactional flows. Furthermore, **model retraining pipelines** could be automated, allowing the system to evolve as fraud patterns shift. These plans are informed by the project evaluation (Section 5.6) and address both product-level and project-level goals. Incorporating them would represent a move from academic proof-of-concept to **robust prototype**, preparing the system for potential use in real-world fraud detection scenarios.

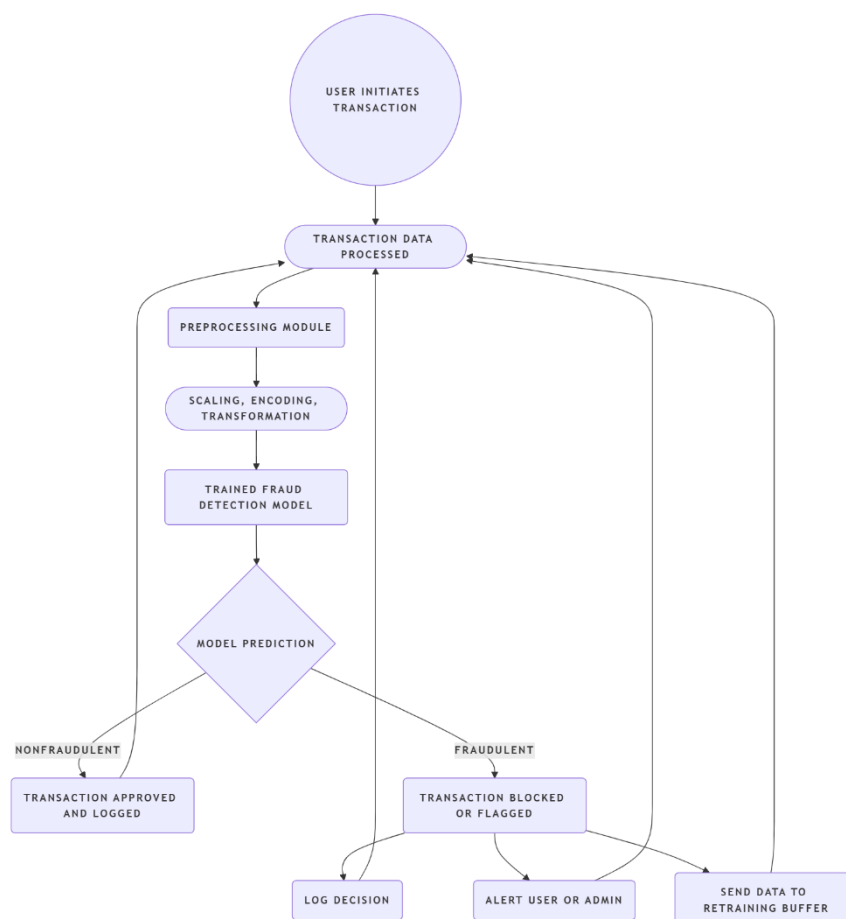


Figure 12 - A flow chart showing how a live feed system based on my model may operate

7.6 EVALUATION AND CONCLUSION OF THE ENTIRE PROJECT

This project stretched both my technical capabilities and personal resilience. Starting with a clear yet ambitious vision, I encountered challenges — especially around class imbalance handling and interface deployment — but adapted through continuous iteration, testing, and refinement.

Every stage, from data preparation to model training and deployment, demanded decision-making grounded in both evidence and practicality. Some approaches were revised or dropped entirely, such as hyperparameter tuning, which was considered but ultimately deprioritised. Still, the final solution exceeded expectations in usability and real-world applicability.

Importantly, the project also helped me develop critical thinking, perseverance, and the ability to pivot when needed. Not every feature was realised, but the work consistently aligned with the core aims set out in Section 1.2 — delivering a functioning and deployable fraud detection system. The final model, **Random Forest** with **SMOTE**, demonstrated reliable performance and has scope for further enhancement. The combination of intelligent preprocessing, solid evaluation methodology,

and successful deployment reflects the comprehensive nature of the project. It stands as a meaningful achievement with measurable outcomes and clear potential for future expansion.

8 REFERENCES

- Bahnsen, A. C., Aouada, D., Stojanovic, J., & Ottersten, B. (2016). Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51, 134–142. <https://doi.org/10.1016/j.eswa.2015.12.030>
- Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3), 602–613. <https://doi.org/10.1016/j.dss.2010.08.008>
- Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys & Tutorials*, 16(1), 303–336. <https://doi.org/10.1109/SURV.2013.052213.00046>
- Blagus, R., & Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. *BMC Bioinformatics*, 14(1), 106. <https://doi.org/10.1186/1471-2105-14-106>
- Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3), 3446–3453. <https://doi.org/10.1016/j.eswa.2011.09.033>
- Brownlee, J. (2023). Imbalanced classification with Python: Better metrics, balance strategies, and modelling. *Machine Learning Mastery*.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357. <https://doi.org/10.1613/jair.953>
- Chen, Z., Zhou, J., & Dai, Q. (2021). A deep learning-based framework for fraud detection in online financial systems. *Information Sciences*, 546, 602–616. <https://doi.org/10.1016/j.ins.2020.09.029>
- Dal Pozzolo, A., Caelen, O., Johnson, R. A., & Bontempi, G. (2015). Calibrating probability with undersampling for unbalanced classification. In *2015 IEEE Symposium Series on Computational Intelligence* (pp. 159–166). <https://doi.org/10.1109/SSCI.2015.33>
- Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning* (pp. 233–240). <https://doi.org/10.1145/1143844.1143874>
- Delamaire, L., Abdou, H., & Pointon, J. (2009). Credit card fraud and detection techniques: A review. *Banks and Bank Systems*, 4(2), 57–68.
- Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. *arXiv preprint*. <https://arxiv.org/abs/1702.08608>
- Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence* (pp. 973–978).
- Fernández, A., García, S., Galar, M., Prati, R. C., Krawczyk, B., & Herrera, F. (2018). Learning from imbalanced data sets. Springer. <https://doi.org/10.1007/978-3-319-98074-4>

- He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In 2008 IEEE International Joint Conference on Neural Networks (pp. 1322–1328). <https://doi.org/10.1109/IJCNN.2008.4633969>
- Lebichot, B., Lebichot, B., Bontempi, G., & Van Roy, S. (2019). A practical blueprint for fraud detection model evaluation. In 2019 International Conference on Data Science, E-learning and Information Systems (DATA'19) (pp. 1–8). <https://doi.org/10.1145/3368691.3368702>
- Leevy, J. L., Khoshgoftaar, T. M., Bauder, R. A., & Seliya, N. (2022). A survey on addressing high-class imbalance in fraud detection datasets. *Journal of Big Data*, 9, 1–47. <https://doi.org/10.1186/s40537-022-00578-w>
- Lipton, Z. C. (2018). The mythos of model interpretability. *Communications of the ACM*, 61(10), 36–43. <https://doi.org/10.1145/3233231>
- Liu, X. Y., Wu, J., & Zhou, Z. H. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2), 539–550. <https://doi.org/10.1109/TSMCB.2008.2007853>
- Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, 50(3), 559–569. <https://doi.org/10.1016/j.dss.2010.08.006>
- Owusu-Adjei, S., Adu-Gyamfi, S., & Boateng, P. A. (2023). Enhancing fraud detection using hybrid ensemble models. *International Journal of Computer Applications*, 175(14), 1–7. <https://doi.org/10.5120/ijca2023922621>
- Pozzolo, A. D., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2015). Credit card fraud detection and concept-drift adaptation with delayed supervised information. In 2015 International Joint Conference on Neural Networks (IJCNN) (pp. 1–8). <https://doi.org/10.1109/IJCNN.2015.7280642>
- Provost, F., & Fawcett, T. (2001). Robust classification for imprecise environments. *Machine Learning*, 42(3), 203–231. <https://doi.org/10.1023/A:1007601015854>
- Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*, 28.
- Smith, J., & Martinez, R. (2021). Building responsible AI systems for finance. *Journal of AI Ethics*, 2(4), 301–315. <https://doi.org/10.1007/s43681-021-00087-3>
- Symantec. (2020). Internet security threat report. <https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence>
- West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. *Computers & Security*, 57, 47–66. <https://doi.org/10.1016/j.cose.2015.09.005>
- Zareapoor, M., & Shamsolmoali, P. (2015). Application of credit card fraud detection: Based on bagging ensemble classifier. *Procedia Computer Science*, 48, 679–685. <https://doi.org/10.1016/j.procs.2015.04.209>
- Zhang, C., Chen, Y., Xie, L., & Song, X. (2021). Explainable AI for fraud detection: A survey and research roadmap. *Artificial Intelligence Review*, 55, 3321–3360. <https://doi.org/10.1007/s10462-021-10013-0>

van Eck, M. (2021). Designing explainable AI systems: A human-centered approach. Delft University of Technology.

9 APPENDICES

PROJECT ETHICAL REVIEW FORM

UNIVERSITY OF HUDDERSFIELD
SCHOOL OF COMPUTING AND ENGINEERING

PROJECT ETHICAL REVIEW FORM

Applicable for all research, masters and undergraduate projects

Project Title:	AI-Driven Fraud Detection System for Financial Transactions
Student:	Zuber Mulla (U2258708)
Course/Programme :	Computer Science with Cyber Security BSc
Department:	School of Computing and Engineering
Supervisor:	Mohammed Al-Mhiqani
Project Start Date:	15/09/2024

ETHICAL REVIEW CHECKLIST

	Yes	No
1. Are there problems with any participant's right to remain anonymous?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
2. Could a conflict of interest arise between a collaborating partner or funding source and the potential outcomes of the research, e.g. due to the need for confidentiality?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3. Will financial inducements be offered?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
4. Will deception of participants be necessary during the research?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
5. Does the research involve experimentation on any of the following?		
(i) animals?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
(ii) animal tissues?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
(iii) human tissues (including blood, fluid, skin, cell lines)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
6. Does the research involve participants who may be particularly vulnerable, e.g. children or adults with severe learning disabilities?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
7. Could the research induce psychological stress or anxiety for the participants beyond that encountered in normal life?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
8. Is it likely that the research will put any of the following at risk:		
(i) living creatures?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
(ii) stakeholders (disregarding health and safety, which is covered by Q9)?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
(iii) the environment?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
(iv) the economy?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
9. Having completed a health and safety risk assessment form and taken all reasonably practicable steps to minimize risk from the hazards identified, are the residual risks acceptable (Please attach a risk assessment form – available at the end of this document)	<input type="checkbox"/>	<input checked="" type="checkbox"/>

STATEMENT OF ETHICAL ISSUES AND ACTIONS

If the answer to any of the questions above is yes, or there are any other ethical issues that arise that are not covered by the checklist, then please give a summary of the ethical issues and the action that will be taken to address these in the box below. If you believe there to be no ethical issues, please enter "NONE".

NONE

STATEMENT BY THE STUDENT

I believe that the information I have given in this form on ethical issues is correct.

Signature: Zuber Mulla Date: 10/12/2024

AFFIRMATION BY THE SUPERVISOR

I have read this Ethical Review Checklist and I can confirm that, to the best of my understanding, the information presented by the student is correct and appropriate to allow an informed judgement on whether further ethical approval is required.

Signature: M. Al-Mhiqani Date: 04/05/2025

SUPERVISOR RECOMMENDATION ON THE PROJECT'S ETHICAL STATUS

Having satisfied myself of the accuracy of the project ethical statement, I believe that the appropriate action is:

The project proceeds in its present form	X
The project proposal needs further assessment by an Ethical Review Panel. The Supervisor will pass the form to the Ethical Review Panel Leader for consideration.	

MODEL PYTHON SCRIPT

✓ Train/Test Split and Baseline

```
[ ] from sklearn.model_selection import train_test_split

# Define features (X) and target (y)
X = df.drop(columns=['Class']) # drop target column
y = df['Class']                # define target

# Train-test split with stratification for class balance
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, stratify=y, random_state=42
)

# Display split details
print(f"Training set shape: {X_train.shape}, Target: {y_train.shape}")
print(f"Testing set shape: {X_test.shape}, Target: {y_test.shape}")
print(f"\nFraud cases in training set: {y_train.sum()}")
print(f"Fraud cases in testing set: {y_test.sum()}")
```

```
↗ Training set shape: (227845, 30), Target: (227845,)
Testing set shape: (56962, 30), Target: (56962,)
```

```
Fraud cases in training set: 394
Fraud cases in testing set: 98
```

Figure 13 - Code for the 80/20 split

```
# Import necessary libraries for data analysis and visualization
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Machine Learning libraries
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_auc_score, roc_curve
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import average_precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import auc

# Resampling techniques for imbalanced dataset
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from collections import Counter

# For visualizations
sns.set(style="whitegrid")

import json # For Saving and loading metrics
import joblib # For saving and loading models
import pickle # For saving and loading metrics
import os # For checking file existences
```

Figure 14 - Code for initializing all the used libraries

Random Forest

```
[ ] # File path for Random Forest metrics
rf_path = "/content/drive/My Drive/Colab Notebooks/rf_metrics_RAW.json"

# Load or train Random Forest
if os.path.exists(rf_path):
    print("Loaded saved Random Forest classification report:")
    with open(rf_path, "r") as f:
        saved_metrics = json.load(f)
    print(pd.DataFrame(saved_metrics).transpose())
else:
    print("No saved classification report found. Training Random Forest model on RAW data...")
    rf_model = RandomForestClassifier(random_state=42)
    rf_model.fit(X_train, y_train)
    y_pred_rf = rf_model.predict(X_test)
    report = classification_report(y_test, y_pred_rf, output_dict=True)
    with open(rf_path, "w") as f:
        json.dump(report, f, indent=4)
    print(pd.DataFrame(report).transpose())
```

KNN

```
# File path for KNN metrics
knn_path = "/content/drive/My Drive/Colab Notebooks/knn_metrics_RAW.json"

# Load or train KNN
if os.path.exists(knn_path):
    print("Loaded saved KNN classification report:")
    with open(knn_path, "r") as f:
        saved_metrics = json.load(f)
    print(pd.DataFrame(saved_metrics).transpose())
else:
    print("No saved classification report found. Training KNN model on RAW data...")
    knn_model = KNeighborsClassifier()
    knn_model.fit(X_train, y_train)
    y_pred_knn = knn_model.predict(X_test)
    report = classification_report(y_test, y_pred_knn, output_dict=True)
    with open(knn_path, "w") as f:
        json.dump(report, f, indent=4)
    print(pd.DataFrame(report).transpose())
```

Logistic Regression

```
[ ] # File path for Logistic Regression metrics
lr_path = "/content/drive/My Drive/Colab Notebooks/lr_metrics_RAW.json"

# Load or train Logistic Regression
if os.path.exists(lr_path):
    print("Loaded saved Logistic Regression classification report:")
    with open(lr_path, "r") as f:
        saved_metrics = json.load(f)
    print(pd.DataFrame(saved_metrics).transpose())
else:
    print("No saved classification report found. Training Logistic Regression model on RAW data...")
    lr_model = LogisticRegression(max_iter=1000, random_state=42)
    lr_model.fit(X_train, y_train)
    y_pred_lr = lr_model.predict(X_test)
    report = classification_report(y_test, y_pred_lr, output_dict=True)
    with open(lr_path, "w") as f:
        json.dump(report, f, indent=4)
    print(pd.DataFrame(report).transpose())
```

SVM

```
# File path for SVM metrics
svm_path = "/content/drive/My Drive/Colab Notebooks/svm_metrics_RAW.json"

# Load or train SVM
if os.path.exists(svm_path):
    print("Loaded saved SVM classification report:")
    with open(svm_path, "r") as f:
        saved_metrics = json.load(f)
    print(pd.DataFrame(saved_metrics).transpose())
else:
    print("No saved classification report found. Training SVM model on RAW data...")
    svm_model = SVC(kernel='rbf', random_state=42)
    svm_model.fit(X_train, y_train)
    y_pred_svm = svm_model.predict(X_test)
    report = classification_report(y_test, y_pred_svm, output_dict=True)
    with open(svm_path, "w") as f:
        json.dump(report, f, indent=4)
    print(pd.DataFrame(report).transpose())
```

Figure 15 - Code blocks which show the 4 models trained and tested on the Raw, imbalanced Data

```
# Shift amount to avoid zero (Box-Cox needs strictly positive values)
X_train['Amount_shifted'] = X_train['Amount'] + 1e-9
X_test['Amount_shifted'] = X_test['Amount'] + 1e-9

# Fit Box-Cox on training set only
X_train['scaled_amount'], fitted_lambda = boxcox(X_train['Amount_shifted'])

# Save the fitted lambda for Box-Cox transformation
joblib.dump(fitted_lambda, 'boxcox_lambda_amount.pkl')

# Load existing saved Box-Cox lambda (.pkl file)
boxcox_lambda = joblib.load('boxcox_lambda_amount.pkl')

# Save the lambda into a JSON file
with open('boxcox_lambda_amount.json', 'w') as f:
    json.dump({'lambda': float(boxcox_lambda)}, f)

print("Saved Box-Cox lambda as boxcox_lambda_amount.json")

# Apply the same transformation using the learned lambda to the test set
X_test['scaled_amount'] = boxcox(X_test['Amount_shifted'], lmbda=fitted_lambda)

# Check skewness before transformation
original_skew = X_train['Amount'].skew()
print(f"Skewness of original 'Amount': {original_skew:.4f}")

# Check skewness after Box-Cox transformation
transformed_skew = pd.Series(X_train['scaled_amount']).skew()
print(f"Skewness of Box-Cox transformed 'Amount': {transformed_skew:.4f}")

# Drop the original and intermediate columns
X_train.drop(['Amount', 'Amount_shifted'], axis=1, inplace=True)
X_test.drop(['Amount', 'Amount_shifted'], axis=1, inplace=True)
```

```
# Copy original datasets
X_train_time = X_train.copy()
X_test_time = X_test.copy()

# Extract Hour from 'Time' column (seconds since first transaction)
X_train_time['Hour'] = (X_train_time['Time'] // 3600) % 24
X_test_time['Hour'] = (X_test_time['Time'] // 3600) % 24

# Apply sin/cos transformation to encode cyclical nature
X_train_time['Hour_sin'] = np.sin(2 * np.pi * X_train_time['Hour'] / 24)
X_train_time['Hour_cos'] = np.cos(2 * np.pi * X_train_time['Hour'] / 24)

X_test_time['Hour_sin'] = np.sin(2 * np.pi * X_test_time['Hour'] / 24)
X_test_time['Hour_cos'] = np.cos(2 * np.pi * X_test_time['Hour'] / 24)

# Drop raw 'Time' column and 'Hour'
X_train_time.drop(columns=['Time', 'Hour'], inplace=True)
X_test_time.drop(columns=['Time', 'Hour'], inplace=True)
```

Figure 16 - The 2 code blocks for the box-cox transformation of 'Amount' and the sine and cosine cyclical encoding of 'Time'

```
[ ] # Before SMOTE
print("Before SMOTE:", Counter(y_train))

# Initialize SMOTE
smote = SMOTE(random_state=42)

# Apply SMOTE to preprocessed training data
X_train_smote, y_train_smote = smote.fit_resample(X_train_time, y_train)

# After SMOTE
print("After SMOTE:", Counter(y_train_smote))
```

```
Before SMOTE: Counter({0: 227451, 1: 394})
After SMOTE: Counter({0: 227451, 1: 227451})
```

```
[ ] # Before ADASYN
print("Before ADASYN:", Counter(y_train))

# Initialize ADASYN
adasyn = ADASYN(random_state=42)

# Apply ADASYN to preprocessed training data
X_train_adasyn, y_train_adasyn = adasyn.fit_resample(X_train_time, y_train)

# After ADASYN
print("After ADASYN:", Counter(y_train_adasyn))
```

```
Before ADASYN: Counter({0: 227451, 1: 394})
After ADASYN: Counter({1: 227460, 0: 227451})
```

Figure 17 - **SMOTE** and **ADASYN** code blocks

```
# Dictionary to store AUC-ROC and PR AUC scores
internal_auc_scores = {}

# Helper function to save and display results
def save_metrics_and_outputs(model_name, y_true, y_pred, y_proba, report_path, internal_auc_dict):
    # === Save and display classification report ===
    report = classification_report(y_true, y_pred, output_dict=True)
    with open(report_path, 'w') as f:
        json.dump(report, f, indent=4)
    print(f"\n{model_name} Classification Report:")
    display(pd.DataFrame(report).transpose())

    # === Display confusion matrix ===
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(6, 5))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=[0, 1], yticklabels=[0, 1])
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(f"{model_name} Confusion Matrix")
    plt.show()

    # === Calculate and store AUC-ROC and PR AUC ===
    auc_roc = roc_auc_score(y_true, y_proba[:, 1])
    precision, recall, _ = precision_recall_curve(y_true, y_proba[:, 1])
    pr_auc = auc(recall, precision)
    internal_auc_dict[model_name] = {
        "AUC_ROC": auc_roc,
        "PR_AUC": pr_auc
    }
    print(f"{model_name} AUC-ROC: {auc_roc:.4f}, PR AUC: {pr_auc:.4f} (Saved internally)")
```

Figure 18 - Function to Save and output Classification Report, Plot Confusion Matrix, and Compute AUC Metrics for the **SMOTE** and **ADASYN** models

TESTING AND EVALUATION RESULTS

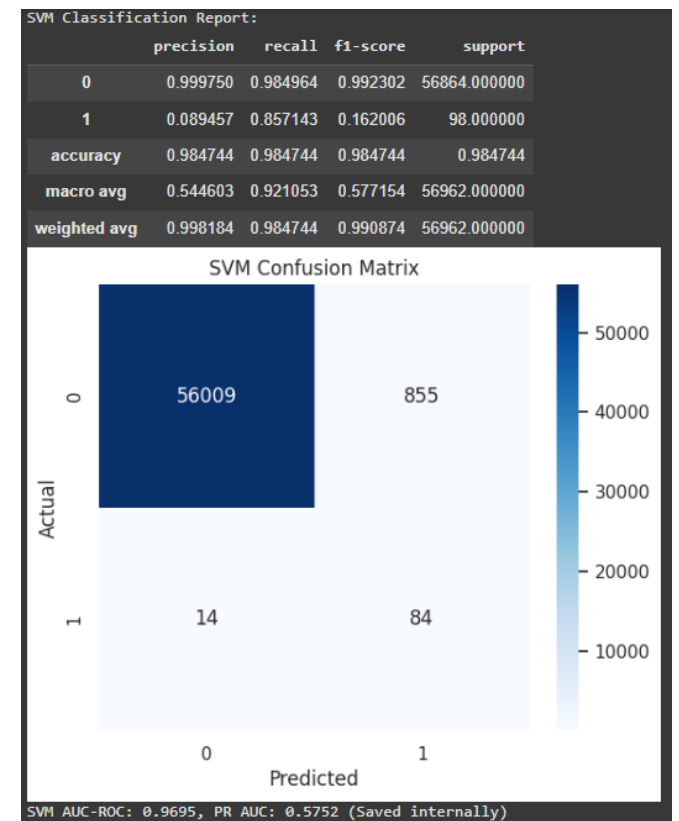
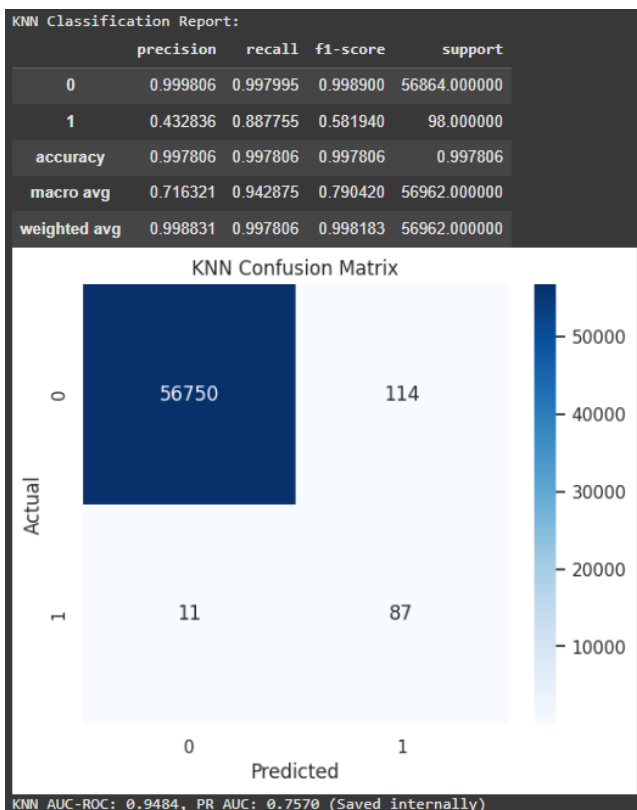
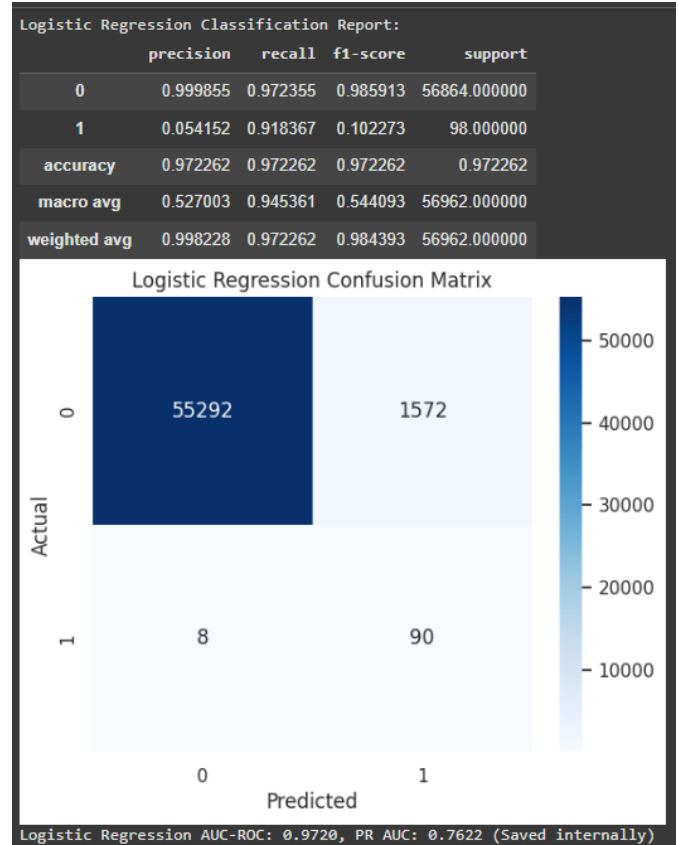
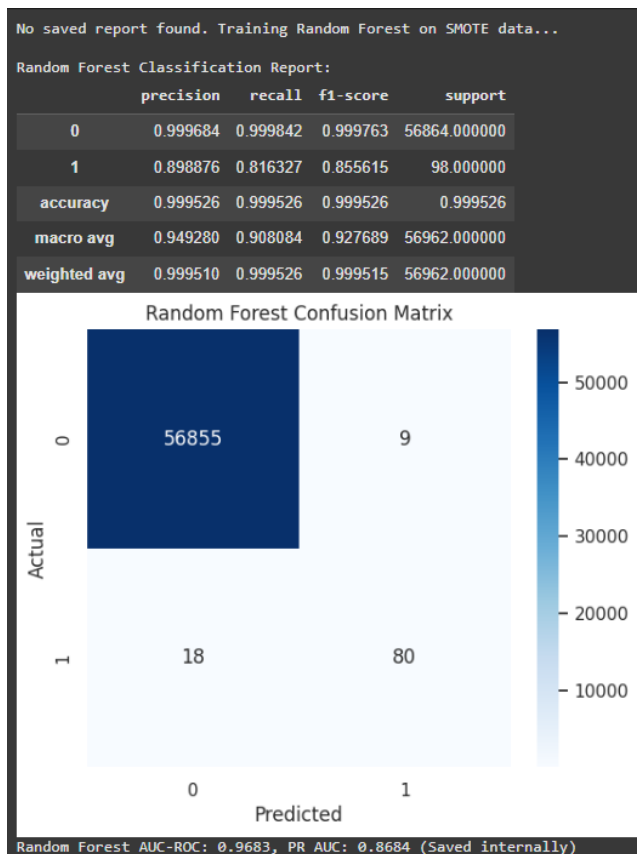


Figure 19 - Confusion Matrices, Classification Reports, AUC-ROC and PR-AUC Scores for all 4 models trained and tested on the SMOTE dataset

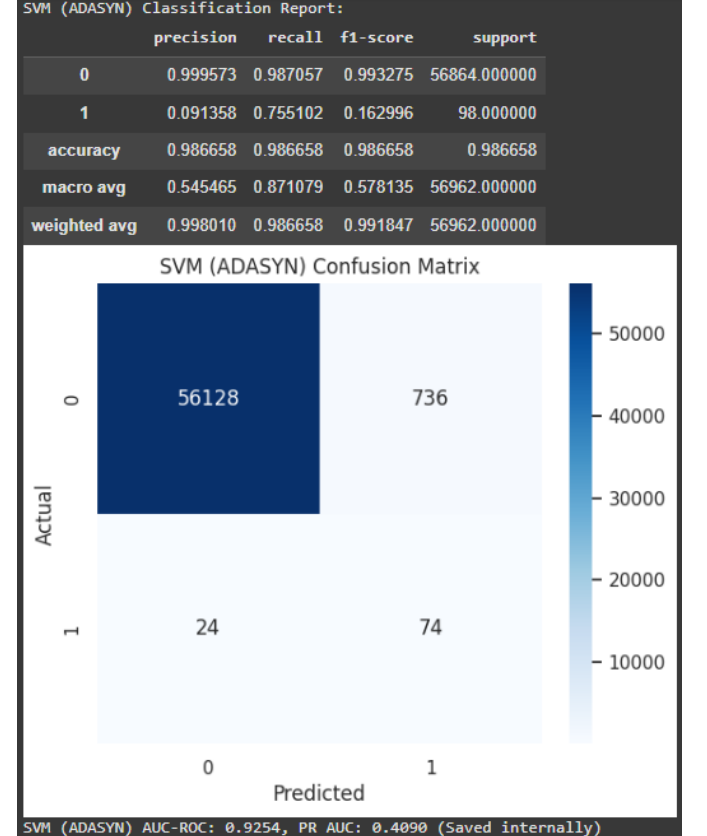
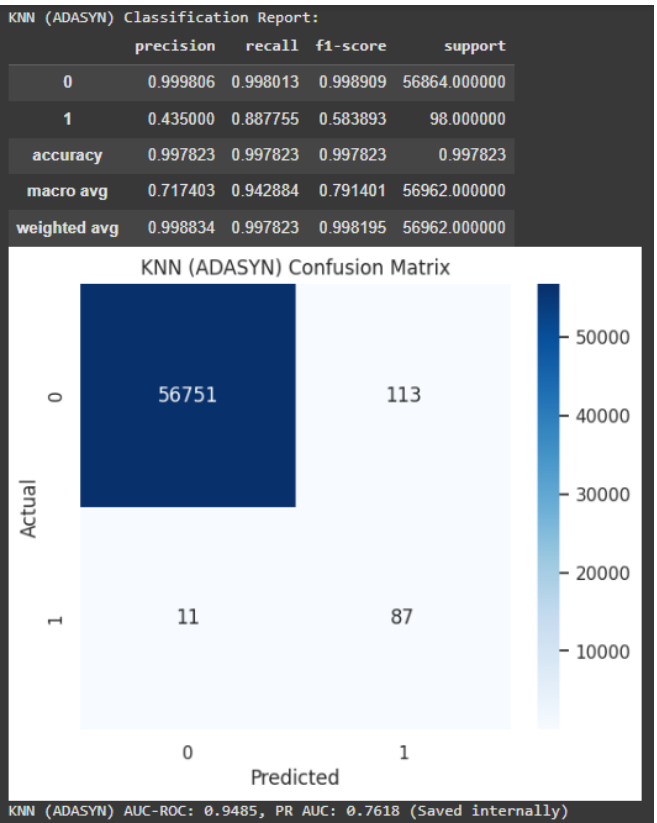
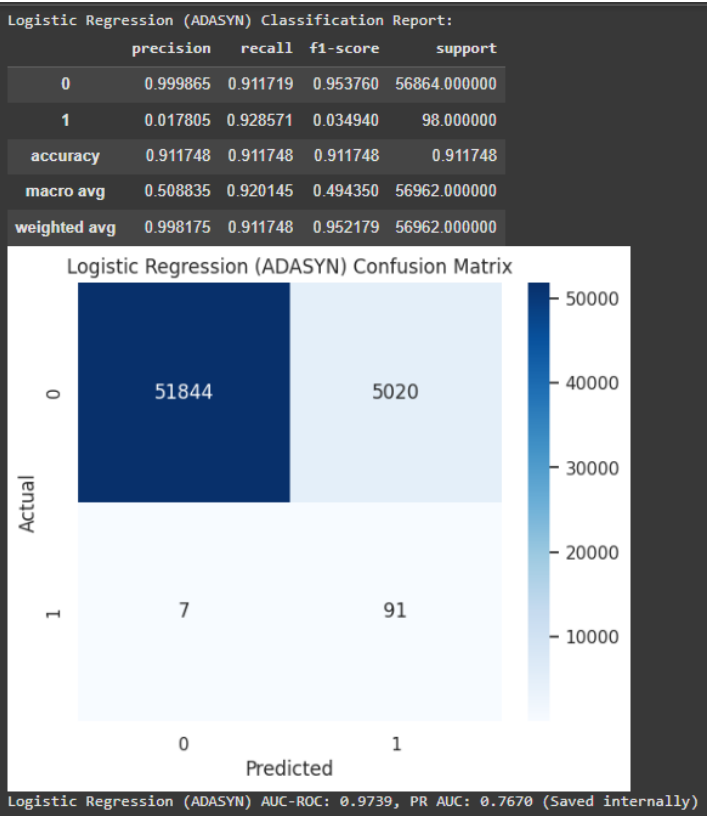
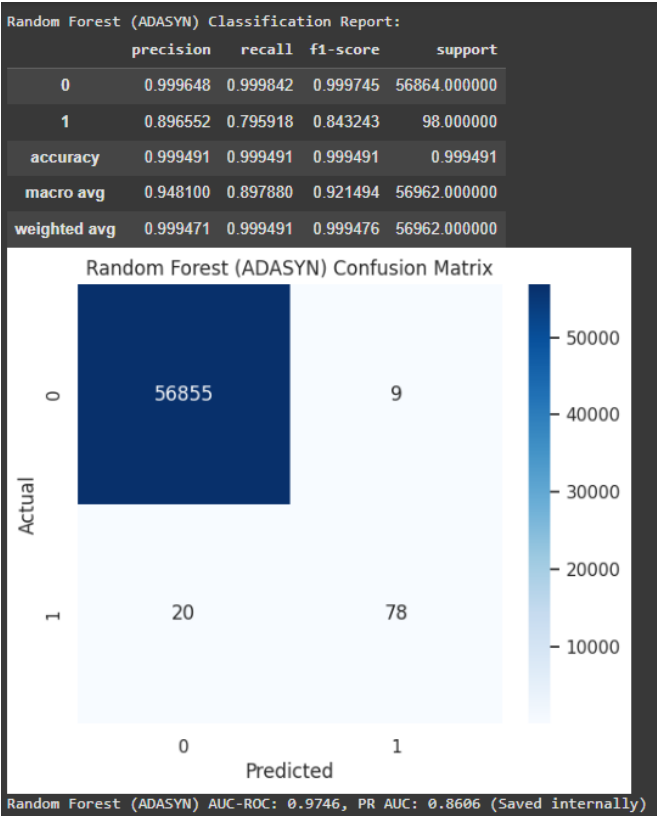


Figure 20 - Confusion Matrices, Classification Reports, AUC-ROC and PR-AUC Scores for all 4 models trained and tested on the ADASYN dataset

WEBPAGE AND INTERFACE DETAILS

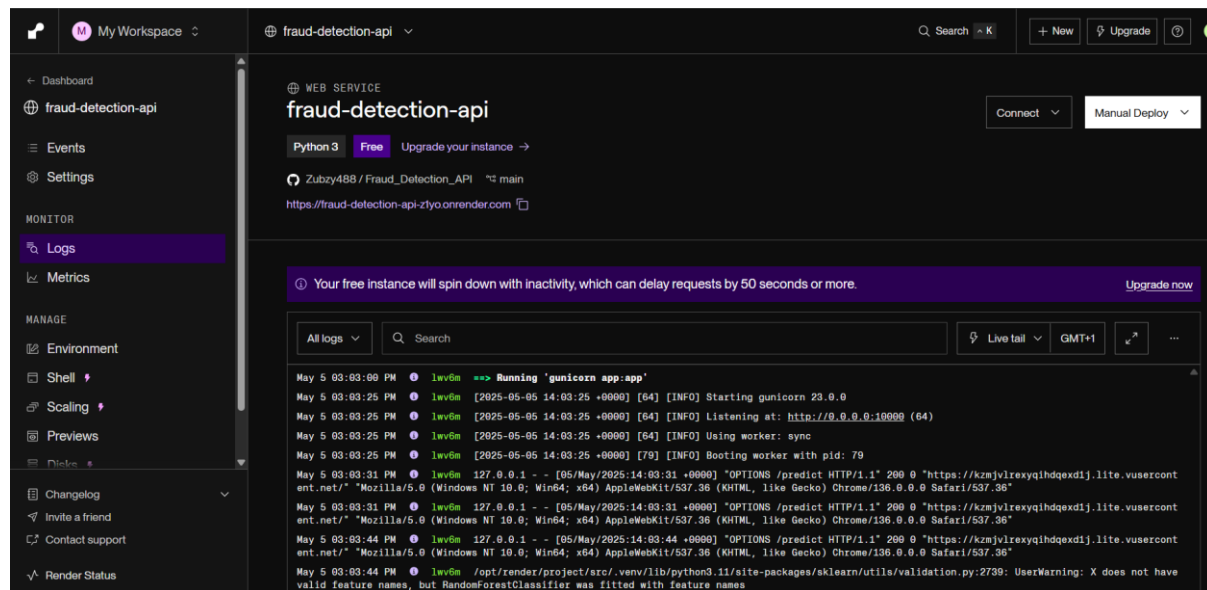
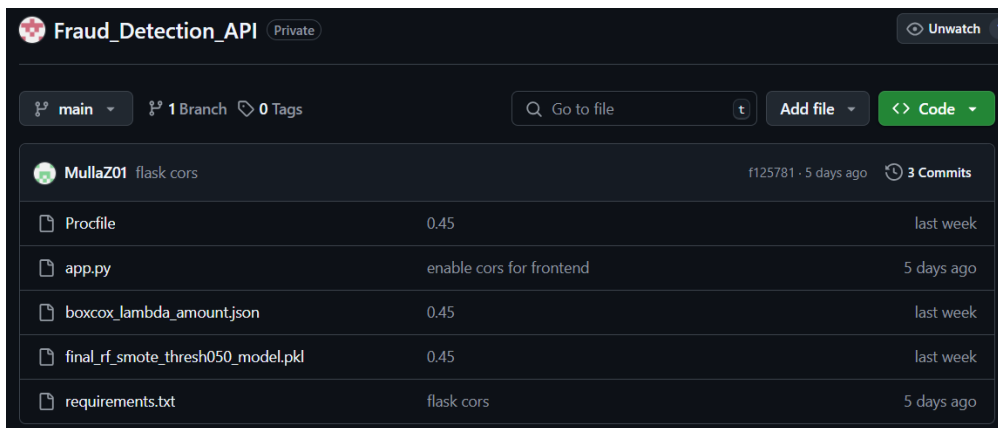


Figure 21 - **GitHub** repository Overview (**left**) showing the core files and **Render API** deployment dashboard (**right**) displaying the active status of the **API**, confirming it is running and active

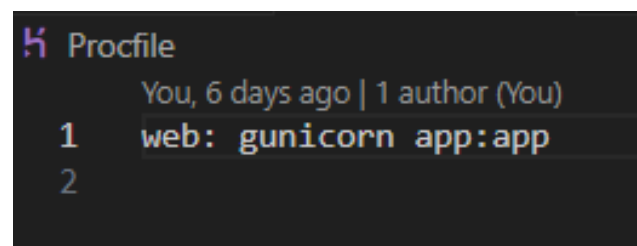
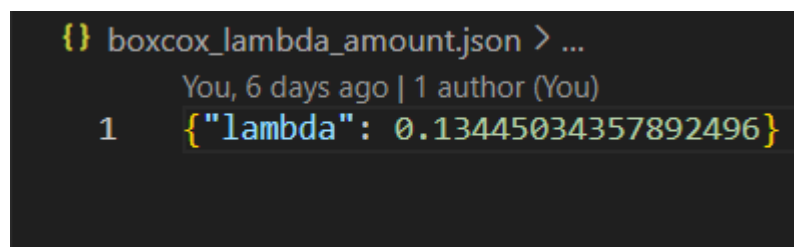
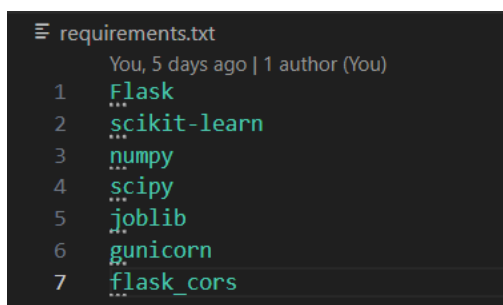







Figure 22 - The **Python Requirements** file (listing the required Python libraries), The **Box-Cox Lambda JSON** File (Contains the pre-calculated lambda value used for scaling the 'Amount' feature during preprocessing) and The **Heroku Procfile** for **API Deployment** (Indicates that the app is served using Gunicorn on Heroku-like platforms, essential for hosting the **Flask API**)

```

app.py 5
app.py > predict
You, 5 days ago | 1 author (You)
1 from flask import Flask, request, jsonify
2 import joblib
3 import numpy as np
4 import json
5 from scipy.stats import boxcox
6 from flask_cors import CORS
7
8
9 app = Flask(__name__)
10 CORS(app)
11
12
13 # Load the trained model
14 model = joblib.load('final_rf_smote_thresh050_model.pkl')
15
16 # Load the Box-Cox lambda
17 with open('boxcox_lambda_amount.json', 'r') as f:
18     boxcox_data = json.load(f)
19 boxcox_lambda = boxcox_data['lambda']
20
21 @app.route('/')
22 def home():
23     return "Fraud Detection API is running."
24
25 @app.route('/predict', methods=['POST'])
26 def predict():
27     data = request.get_json()
28
29     # List of V1 to V28 feature names
30     v_features = [f'V{i}' for i in range(1, 29)]
31
32     # Check that all required fields are present
33     missing = [feat for feat in (v_features + ['Amount', 'Time']) if feat not in data]
34     if missing:
35         return jsonify({'error': f'Missing features: {missing}'}), 400
36
37     # Extract V1 to V28
38     v_values = [data[f'V{i}'] for i in range(1, 29)]
39
40     # Preprocess Amount
41     amount = data['Amount'] + 1e-9
42     scaled_amount = boxcox([amount], lambda=boxcox_lambda)[0]
43
44     # Preprocess Time
45     time = data['Time']
46     Hour = (time // 3600) % 24
47     time_sin = np.sin(2 * np.pi * Hour / 24)
48     time_cos = np.cos(2 * np.pi * Hour / 24)
49
50     # Create the final input array
51     features = np.array([v_values + [scaled_amount, time_sin, time_cos]])
52
53     # Predict
54     prob = model.predict_proba(features)[0][1]
55     is_fraud = int(prob >= 0.45)    You, 6 days ago + 0.45
56
57     return jsonify({
58         'fraud_probability': float(prob),
59         'is_fraud': bool(is_fraud)
60     })
61
62 if __name__ == '__main__':
63     app.run(host='0.0.0.0', port=5000)
64

```

Figure 23 - Code used for model loading, preprocessing (including **Box-Cox** and time features), and making fraud predictions via API (left) and **Local API Deployment Files**, File explorer view of all backend API files needed for model deployment and prediction logic (right)

Name	Date modified	Type	Size
 app	30/04/2025 13:46	Python File	2 KB
 boxcox_lambda_amount	28/04/2025 15:28	JSON File	1 KB
 final_rf_smote_thresh050_model.pkl	29/04/2025 14:26	PKL File	15,853 KB
 Procfile	29/04/2025 14:36	File	1 KB
 requirements	30/04/2025 13:58	Text Document	1 KB

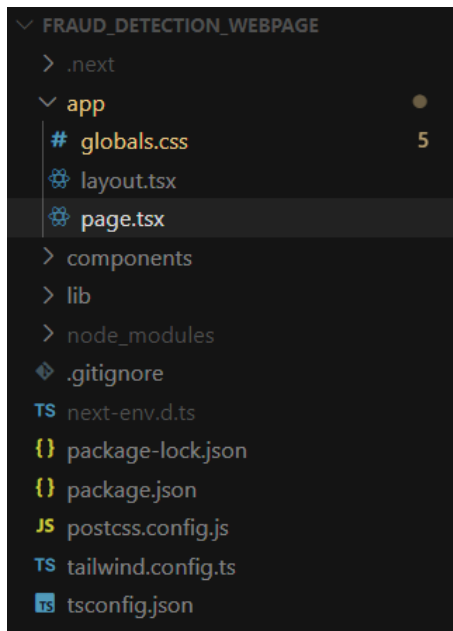


Figure 24 - Project structure of the fraud detection website, showing key frontend files such as `layout.tsx`, `page.tsx`, and configuration files for TailwindCSS, TypeScript, and environment settings.

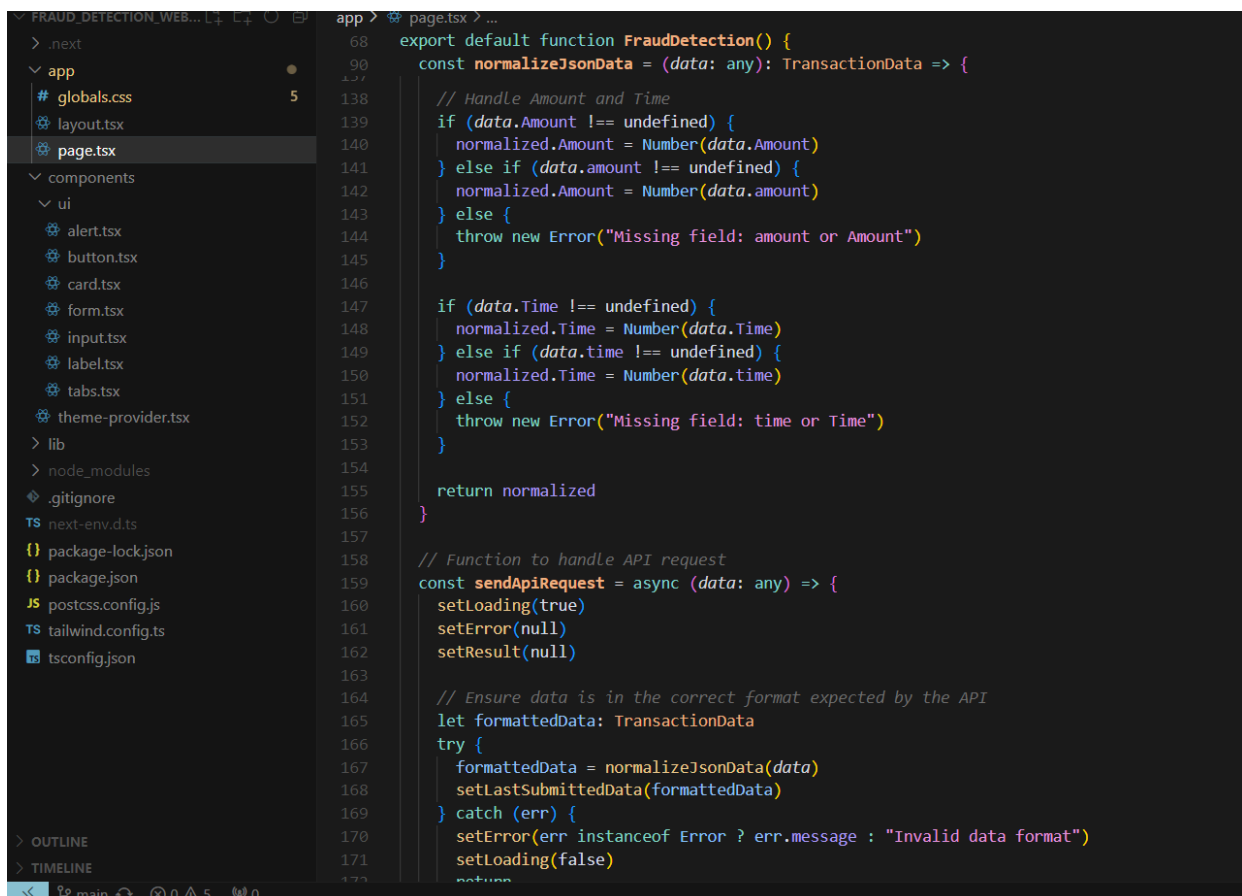


Figure 25 - logic for data normalization and **API** request handling. This section ensures that **"Amount"** and **"Time"** fields are correctly formatted before sending them to the backend, improving frontend robustness and error handling.














































Recent						
<div> <div>Type</div> <div>People</div> <div>Modified</div> <div>Source</div> </div>						
Name			Owner	File size	Location	
 rf_SMOTE_metrics.json	Apr 24 • Uploaded	 me	709 bytes	 Colab Notebooks	⋮	
 svm_ADASYN_metrics.json	Apr 23 • Uploaded	 me	712 bytes	 Colab Notebooks	⋮	
 knn_ADASYN_metrics.json	Apr 23 • Uploaded	 me	696 bytes	 Colab Notebooks	⋮	
 rf_ADASYN_metrics.json	Apr 23 • Uploaded	 me	709 bytes	 Colab Notebooks	⋮	
 lr_ADASYN_metrics.json	Apr 23 • Uploaded	 me	713 bytes	 Colab Notebooks	⋮	
 svm_SMOTE_metrics.json	Apr 23 • Uploaded	 me	710 bytes	 Colab Notebooks	⋮	
 knn_SMOTE_metrics.json	Apr 23 • Uploaded	 me	711 bytes	 Colab Notebooks	⋮	
 lr_SMOTE_metrics.json	Apr 23 • Uploaded	 me	711 bytes	 Colab Notebooks	⋮	
 svm_metrics_RAW.json	Apr 21 • Uploaded	 me	634 bytes	 Colab Notebooks	⋮	
 knn_metrics_RAW.json	Apr 21 • Uploaded	 me	681 bytes	 Colab Notebooks	⋮	
 lr_metrics_RAW.json	Apr 21 • Uploaded	 me	709 bytes	 Colab Notebooks	⋮	
 rf_metrics_RAW.json	Apr 21 • Uploaded	 me	710 bytes	 Colab Notebooks	⋮	
 svm_metrics.json	Apr 21 • Uploaded	 me	696 bytes	 Colab Notebooks	⋮	
 knn_metrics.json	Apr 21 • Uploaded	 me	709 bytes	 Colab Notebooks	⋮	
 lr_metrics.json	Apr 21 • Uploaded	 me	710 bytes	 Colab Notebooks	⋮	

Figure 26 - Google Drive containing all the saved models and JSON metrics