UNIVERSITY OF TARTU

Institute of Computer Science

Data Science Curriculum

Anton Malkovski

Machine learning methods in Anti-Money Laundering

Master's Thesis (15 ECTS)

Supervisor(s): Kadir Aktas, PhD

Tartu 2025

Machine learning methods in Anti-Money Laundering

Abstract:

Anti-Money Laundering (AML) is a critical operation in the financial sector, and with the constant growth in transaction volume, traditional AML methods are no longer sufficient for effectively detecting and preventing money laundering activities. Machine learning (ML) has the potential to discover complex patterns within the vast amounts of transactional data and reduce the false positives (FP) in the AML alerts. This thesis analyzes the applicability of machine learning in AML and proposes a full training pipeline that covers model training, hyperparameter optimization, and synthetic data generation. The work focuses on training a machine learning model with focus on reducing FP noise while being able to classify true positive (TP) alerts by augmenting the highly imbalance training dataset with synthetically generated minority class samples using a Variational autoencoder (VAE) and applying Optuna hyperparameter optimization to tune the model. The results of the experiments demonstrate that this method improves the model's performance while maintaining its ability to generalize to unseen data, finally achieving noise reduction of FP alerts by 40%.

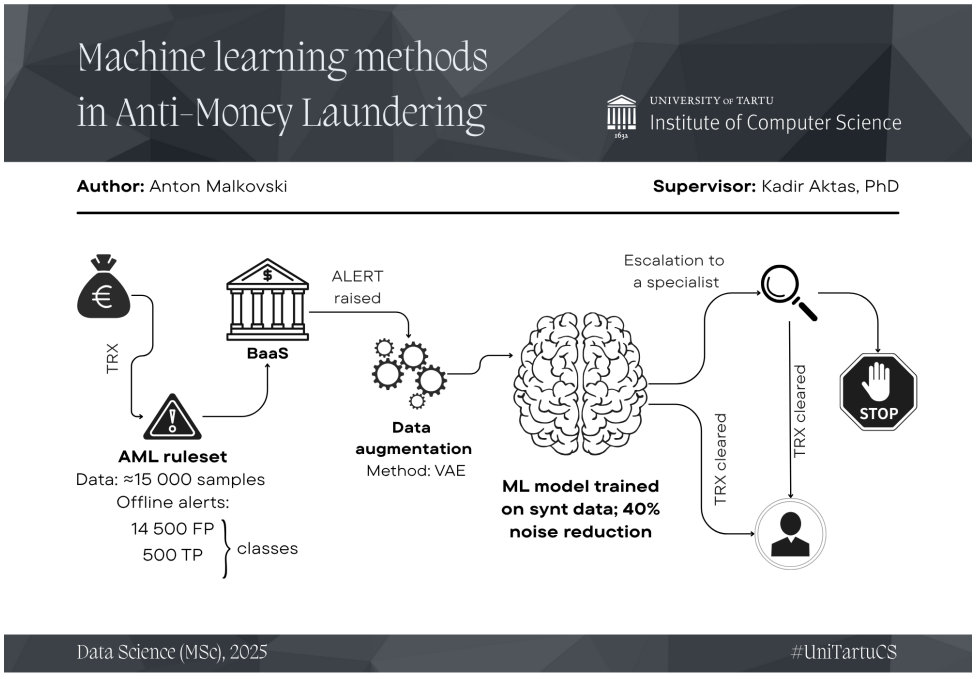


Figure 1: Graphical abstract

Keywords:

Machine Learning, Anti-Money Laundering, Fin-Crime, Artificial Intelligence, Banking, Banking as a service, Deep Learning, Variational autoencoders, Explainable AI

CERCS: P176 Artificial intelligence

Masinõppe meetodid rahapesu tõkestamisel

Abstrakt:

Rahapesu tõkestamine (AML) on finantssektoris kriitilise tähtsusega tegevus ning tehingute mahu pideva kasvuga ei ole traditsioonilised AML meetodid enam piisavad rahapesu efektiivseks tõkestamiseks. Masinõppel (ML) on potentsiaal avastada keerukaid mustreid tohututes tehinguandmete kogustes ning vähendada valepositiivseid (FP) tulemusi AML andmetes. Käesolev lõputöö analüüsib masinõppe rakendatavust rahapesu tõkestamisel ning pakub välja treening protsessi, mis hõlmab mudeli treenimist, hüperparameetrite optimeerimist ja sünteetiliste andmete genereerimist. Töö keskendub masinõppe mudeli treenimisele eesmärgiga vähendada valepositiivset müra, säilitades samal ajal võimet klassifitseerida päriselt olulisi tehinguid potentsiaalselt ohtlikeks, täiendades äärmiselt tasakaalustamata treeningandmestikku sünteetiliselt genereeritud vähemusklassi näidistega, kasutades VAE-d ning rakendades Optuna hüperparameetrite optimeerimist mudeli häälestamiseks. Katsete tulemused näitavad, et see meetod parandab mudeli efektiivsust, saavutades lõpuks valepositiivsete hoiatuste müra vähenemise 40% võrra.

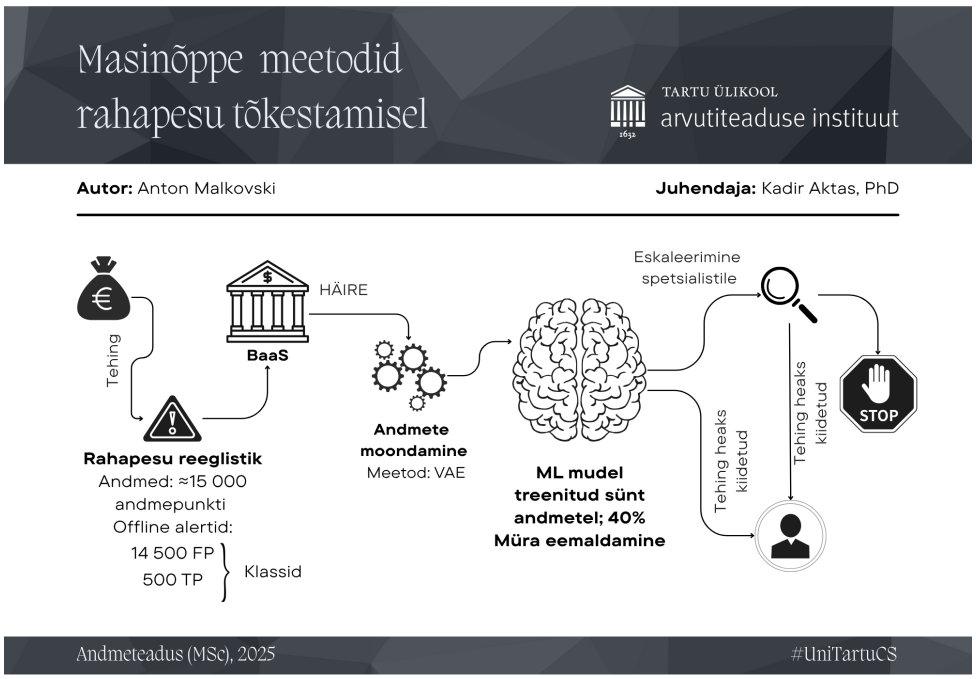


Figure 2: Visuaalne abstrakt

Võtmesõnad:

Masinõpe, rahapesu tõkestamine, finants kuritegevus, Tehisintellekt, AI, Pangandus

CERCS: P176 Tehisintellekt

Contents

List of Figures

List of Tables

List of Abbreviations

AML	Anti-Money Laundering
BaaS	Banking as a Service
TP	True Positive
FP	False Positive
VAE	Variational Autoencoder
SMOTE	Synthetic Minority Oversampling Technique
SHAP	SHapley Additive exPlanations
LIME	Local Interpretable Model-Agnostic Explanations
TPOT	Tree-based Pipeline Optimization Tool
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
ML	Machine Learning
AutoML	Automated Machine Learning
Optuna	Optimization Framework for Hyperparameter Search
API	Application Programming Interface

1 Introduction

Anti-money laundering (AML) is one of the key operational units in every bank structure, because of the regulatory requirements and common sense, all financial institutions providing transactional services must monitor their customer base and incoming transactions for potential money laundering breach. AML refers to the laws, regulations, and practices aimed at preventing criminals from disguising illicit funds as legitimate. Globally, the scale of money laundering is enormous: an estimated 2–5% of world GDP is laundered each year, yet only about 1.1% of these illicit funds is ultimately recovered [1]. Such illicit financial flows fuel organized crime and terrorism, undermining economic stability and public safety [1]. In practice, AML efforts encompass a broad range of crimes and financial activities – from drug trafficking and corruption to tax evasion and terror financing – all of which generate "dirty money" that must be cleaned to enter the legitimate economy [1]. Because of this broad impact, AML has become a global priority, with governments and institutions worldwide implementing measures to detect and deter laundering.

Traditional AML transaction monitoring systems have largely relied on predefined rules and thresholds (e.g., "if transaction exceeds X or if behavior deviates from profile, then flag") [2]. Although these rule-based controls are valued for their transparency and operational simplicity, they have proven to be notably inefficient in practice for example due to the constant need for manual thresholding adjustments. Industry experts observe that such systems typically generate an overwhelming volume of alerts that do not correspond to actual illicit activity, creating a persistent false positive problem [3]. Money laundering schemes can be simple, for instance, smurfing, which involves making multiple deposits below the threshold where banks must report cash transactions to regulatory authorities [4]. On the other hand, money laundering schemes can be very complex, criminals may use multiple institutions, countries and fake documentation in order to hide the origins of illicit funds [4]. It is important that these complex schemes are reviewed by AML specialists who are domain experts and are able to discover money laundering patterns in the vast sea of data [5].

These challenges are particularly pronounced in emerging financial service models such as Banking as a Service (BaaS). This research was conducted at LHV Bank Ltd, a specialized BaaS provider that processes transactions of over 200 financial institutions. In the BaaS model, licensed banks deliver comprehensive banking infrastructure, products, and services to other businesses through application programming interfaces (APIs) [6]. The BaaS provider manages critical backend functions, including regulatory compliance, security protocols, risk management systems and payments flow thereby enabling non-banking companies to offer financial services to their customers [6]. This creates a distinct transaction ecosystem where traditional AML approaches face additional complexities because bank's direct customers are service providers rather than end-users, creating unique challenges in customer profiling and transaction monitoring.

The combination of inherent scheme complexity, enormous transaction volumes, and rudimentary rule-based screening creates an alert burden, posing a significant challenge for human AML analysts to review effectively and efficiently. This operational challenge presents a compelling use case for machine learning algorithms, which can augment AML specialists by improving existing control mechanisms and analyzing large datasets to extract relevant patterns [7].

This thesis specifically focuses on improving transaction monitoring systems using machine learning. While machine learning holds substantial promise for AML detection, its implementation faces significant constraints including limited data availability, stringent regulatory requirements, and technological limitations. Research indicates that while descriptive profiling has proven relatively effective, predictive profiling continues to face considerable challenges requiring human oversight and substantial investment in enhanced AI infrastructure [7, 8]. This study suggests that supervised learning approaches represent a promising direction for future AML efforts due to their ability to detect novel patterns while simultaneously helping to minimize false positives.

1.1 Banking as a service (BaaS)

Banking as a service model enables third-party companies to embed financial services into their offerings by leveraging the regulated infrastructure and APIs provided by licensed banks [6]. The transactional data in a BaaS model is significantly different from traditional retail banking. Since the bank's direct customers are other service providers rather than end-users, traditional customer data aggregation methods are not always the best solution for creating comprehensive risk profiles that aid in detecting AML patterns. To illustrate this ecosystem, consider the following example: An e-commerce platform partners with the bank to provide payment services. In this scenario, the e-commerce platform's customer is not directly the bank's customer. The bank only becomes aware of the individual customer when a payment for goods is processed. This creates a unique challenge in customer profiling, as the bank has visibility only through isolated transactions, making it difficult to construct a customer profile.

For instance, if a customer purchases a laptop through an online retailer using the bank's integrated payment system, the bank will only see single transaction details related to the purchase, without context about the customer's broader purchasing habits or relationship with the e-commerce platform.

In this transactional ecosystem, each participant, BaaS provider, the e-commerce platform, and the originating bank, must independently conduct comprehensive monitoring and customer due diligence to ensure the effectiveness of the financial transaction and mitigate potential risks. This multi-layered approach to customer verification and transaction monitoring is crucial for maintaining the integrity of the financial system and complying with AML regulations.

1.2 Transaction monitoring

AML monitoring systems can be categorized into two types: offline and online monitoring.

Online monitoring happens in real-time. Each transaction passes through a rule-based AML system. If flagged as suspicious, the transaction is immediately stopped and sent for manual review. A monitoring specialist then examines the alert and either allows the transaction to proceed or, if something unusual is found, escalates it further. Depending on the risk type, the transaction may be frozen or completely stopped.

In contrast, offline monitoring is a post-transaction analysis approach. The primary goal of offline systems is to detect suspicious patterns retrospectively, rather than prevent transactions in real-time. A key characteristic of offline monitoring systems is batch processing. Transactions are accumulated over a specified period and then processed collectively, generating batches of flagged transactions for monitoring specialists to review. This approach allows for the identification of more complex money laundering patterns that may unfold over an extended period, rather than being evident in a single transaction [9].

Monitoring, offline alerts can be categorized into two distinct subcategories: periodic and non-periodic alerts.

Periodic alerts are designed to capture sophisticated money laundering techniques like smurfing. These alerts aggregate transactions over specific time intervals, identifying patterns that might not be apparent in individual transactions. In the BaaS context, such alert aggregation becomes computationally intensive and time-consuming, requiring complex data processing and pattern recognition algorithms.

This research concentrates on non-periodic offline alerts, which offer a more streamlined approach to transaction monitoring. These alerts typically capture individual transactions with specific risk characteristics, such as transactions originating from jurisdictions with complex financial regulations or payments to high-risk geographical regions.

Transactions with higher-risk jurisdictions present a monitoring challenge. While conducting business with these countries is not illegal, the regulatory environment introduces elevated money laundering risks. The complexity stems from looser financial regulatory frameworks, potential tax exemption mechanisms, limited international financial oversight, and reduced transparency in financial reporting [3].

A key challenge in current monitoring systems is the massive volume of transactions, which results in rule-based systems generating extensive alert batches for specialists to review [3]. Typically, the majority of these alerts are false positives, meaning monitoring specialists must examine tens or hundreds of transactions, cross-reference multiple information sources, and engage in client communication to ultimately classify an alert as a false positive.

1.3 Key Challenges in ML-based AML

Machine learning approaches to AML face several critical challenges, beginning with the scarcity of publicly available datasets. Due to the sensitive nature of financial data, most real-world datasets remain confidential, limiting research to synthetic datasets or proprietary datasets used in internal studies by financial institutions [2]. These proprietary datasets often involve undisclosed features and risk scoring methodologies, making it difficult to reproduce results or make meaningful comparisons across different ML approaches.

Even when access to relevant data is available, the extreme class imbalance presents a significant obstacle [2]. The vast majority of financial transactions are routine and legitimate—such as salary deposits, purchases, and savings contributions. Consequently, only a small fraction of alerts raised by AML systems correspond to suspicious activity [10]. It is important to note that sometimes even after a transaction is classified to be suspicious by the specialist it does not mean that these transactions are actually related to money laundering or other illicit activity. Usually a very small proportion of reported transactions are actually related to real criminal activity. In some cases when a transaction is flagged and classified as suspicious it means that there is not enough information to state that alerted transactions are not related to illicit activity due to the lack of background information. As a result, labeled datasets often contain a high proportion of false positives, which complicates model training and undermines the performance of machine learning systems [10].

The dynamic nature of money laundering presents another layer of complexity. Money laundering tactics evolve in response to detection methods, creating a constantly moving target. Furthermore, since money laundering has no direct victims, illicit transactions can potentially go undetected for longer periods than other forms of financial fraud [11].

Lastly, regulations in the financial sector require that the decisions made by machine learning or deep learning systems be explainable. Supervisory authorities don't impose specific requirements on what type of system is used for AML as long as it can be properly explained both internally and to regulators [12]. This poses a challenge for the developers of the systems since they have to choose algorithms that are more explainable and are reliant on relevant features for predictions since those explanations should be understandable to both supervisory authorities and stakeholders in the financial sector.

1.4 Overview of Thesis Structure

Following this introduction: Chapter 2 reviews relevant literature on AML detection methods, including traditional approaches and machine learning applications. Chapter 3 describes the data and methodology used in this study, including preprocessing and model validation techniques. Chapter 4 presents experimental results, comparing different models in reducing false positives. Chapter 5 discusses the findings in the context of Bank's operations and broader AML compliance, addressing both achievements and limitations it also concludes by summa-

rizing the experiments and suggesting directions for future work. This structure demonstrates how machine learning can enhance AML efforts in modern banking while balancing vigilance and efficiency.

2 Literature Review

The development of AML systems has progressed through several generations, from manual review processes to increasingly sophisticated automated solutions. The earliest AML systems were entirely manual, relying on human analysts to review transactions and identify suspicious patterns [11]. As transaction volumes grew, rule-based systems emerged in the 1990s, implementing simple if-then logic to flag transactions meeting specific criteria, such as threshold amounts or high-risk jurisdictions [2]. While rule-based systems are valued for their transparency and operational simplicity, they have demonstrated serious limitations—particularly a high rate of false positives and poor adaptability to novel laundering typologies [3].

To address these shortcomings, the financial sector has increasingly explored the use of supervised machine learning. These models learn from historical labeled data to identify patterns and relationships that signify suspicious activity. Unlike rule-based systems, which rely on explicit logic, supervised learning models can detect complex, non-linear relationships and adapt over time. A wide range of supervised algorithms have been applied to AML detection, including logistic regression, decision trees, random forests, support vector machines (SVMs), and deep learning architectures such as multilayer perceptrons [13]. These models have demonstrated improved detection capabilities and reduced false positive rates in controlled experiments and pilot implementations.

Among these, tree-based models such as random forests and gradient boosting have gained popularity due to their balance of performance and interpretability. In benchmark evaluations, they have often matched or outperformed more complex deep learning models when trained on structured transaction data. For instance, Bakry et al. [14] showed that ensemble classifiers achieved over 88% recall and significantly reduced false alerts when applied to a labeled AML dataset. Similarly, Weber et al. [15] evaluated a graph neural network approach on the Bitcoin transaction dataset but found that traditional random forests classifier still outperforms a graph neural network when properly tuned.

Despite their potential, supervised learning approaches face several significant challenges in the AML domain. One of the most persistent issues is extreme class imbalance. In typical financial datasets, illicit activity accounts for a very small fraction—often less than 2%—of all transactions [10]. This skew leads to models that are biased toward the majority class, reducing their ability to detect actual money laundering. Addressing this imbalance often involves resampling techniques, class weighting, or synthetic data generation using methods such as Variational Autoencoders (VAEs) or Wasserstein Generative Adversarial Networks (WGANs) [16]. These synthetic techniques attempt to simulate plausible illicit transaction patterns to improve the model's learning capacity. Chen et al. [16] successfuly implement VAE-s and WGAN-s on a banking dataset containing fraudulent transactions and reduced the false positive rate down to 7% by synthetically generating additional true positive samples for the data.

Another key challenge is the limited availability of publicly labeled AML data. Due to pri-

vacy, legal, and competitive concerns, most financial institutions do not share labeled datasets, making it difficult for researchers to benchmark and validate their models. This has led to increased reliance on synthetic datasets like PaySim and Elliptic, which—while useful—do not fully replicate the complexities of real-world financial systems or BaaS environments [15].

In addition to data-related challenges, AML models must satisfy explainability requirements. Supervisory authorities generally do not prescribe which models should be used but require that the decision-making process behind each flagged transaction is clearly interpretable and auditable [12]. This creates a tension between the performance of complex models and the transparency of simpler ones. To mitigate this, explainable AI techniques such as SHAP values and LIME are increasingly used to provide insight into feature contributions and model rationale [17]. In their research, Kute et al. [17] utilized a synthetic, highly imbalanced transaction dataset to train various models—including neural networks, Support Vector Machines (SVM), XGBoost, and Random Forest—with the aim of understanding their decision-making processes through SHAP and feature importance visualizations. Their attempts to implement LIME were unsuccessful due to its incompatibility with one-dimensional convolutional neural networks, highlighting the ongoing challenges in model explainability.

Given these practical and regulatory constraints, interpretable models are preferred for their strong accuracy, recall, and compliance with internal governance processes. At the same time, the research literature consistently emphasizes that achieving effective AML outcomes depends not only on model performance but also on operational integration, regulatory compliance, and alignment with the investigative workflow [18]. In this context, the targeted application of supervised machine learning—to reduce false positives and enhance signal quality in alerting systems—presents a viable and impactful use case.

3 Methodology

3.1 Proposed framework

The author proposes a framework designed to reduce false positive alerts in the current monitoring system by incorporating a machine learning algorithm into the monitoring pipeline. While a similar framework was proposed by [14], this research distinguishes itself through a key methodological difference. The previous approach was developed using a retail bank dataset with training data based on aggregated customer profiles rather than scenario types, in this case individual customers might have multiple alerts with very different scenario types. In contrast, the proposed framework aims to filter out false positives using information contained within a single alert. This addresses a critical limitation in existing methodologies by focusing on more granular, single-alert level analysis.

This research proposes a machine learning approach visualized in Figure 3 to reduce alert noise by minimizing false positive alerts. The process begins with transactional data from the bank's data warehouse (DWH) flowing into the AML system, which generates alerts stored in an alerts database. This alert data is then preprocessed to make it machine readable. Then the processsed data is fed into a ML model that classifies the alerts either FP or TP. Alerts classified with very high probability of being false positives are routed for dismissal, while those with low false positive probability are directed to monitoring specialists for further investigation. The framework aims to not only significantly decrease the workload of monitoring specialists but also enable more effective prioritization of potentially serious cases.

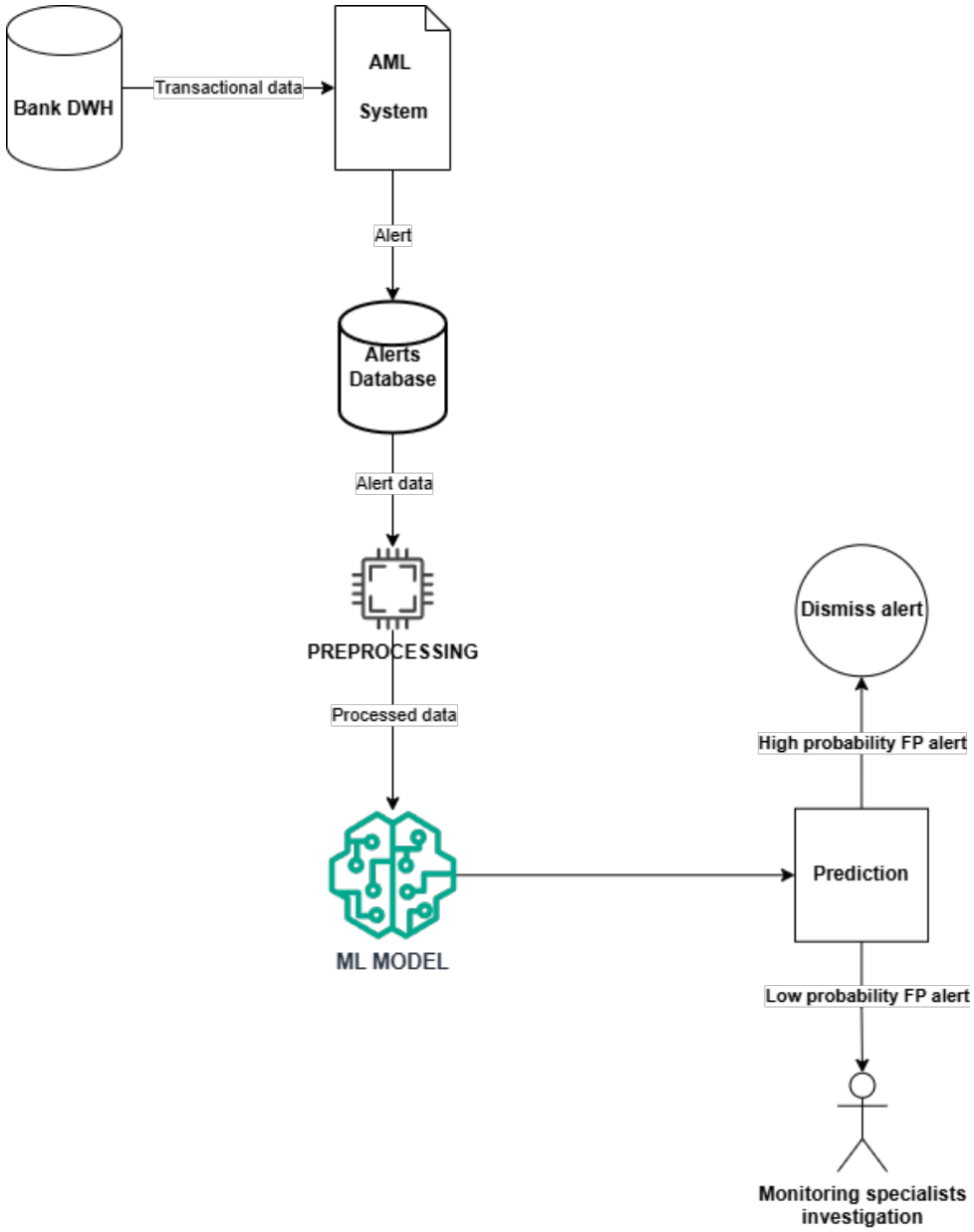


Figure 3: Proposed AML framework (simplified)

3.2 Data and preparation

Dataset used for the model was collected from the bank alert database and consists of raised alerts. These alerts are of offline non periodic type, this subcategory of offline alerts was chosen for the initial noise reduction model because the patterns in the data are simpler and the overall risk is much lower if a model misclassified an alert.

The total number of offline alerts during this time was much larger but after conducting data analysis it was decided to remove some of the alerts from the training data that had low value. Key reason behind that decision was that monitoring rules have been enhanced and thus became more accurate, this means that keeping older alerts would have added unnecessary noise to the training dataset, since such alerts would not be raised in the future according to the

new enhanced rules. Missing values in the data were replaced by placeholder "missing", this is considered an important piece of information when a certain value is missing for an alert by the author. The reason behind it being that some of the BaaS customers have different approaches when it comes to collecting their client information. For example some might collect birth country information and some might not, therefore if this value is missing in the data we can assume what type of customer we are dealing with.

The focus on non-periodic alerts was strategic, chosen for their simpler underlying data patterns and lower inherent risk in potential misclassifications. This methodical approach to data preparation sets a robust foundation for developing an effective noise reduction model in the complex BaaS transaction monitoring landscape.

3.3 Dataset description

The training dataset consists of 14 522 unique offline non-periodic alerts of which 557 are marked as a TP and 13 965 as a FP so the class distribution is 96/4. The dataset contains alert and transaction information that was collected over a period of 1.5 years. A more extended historical time frame was deliberately excluded as significant operational changes in AML rules implemented in recent years rendered many older alerts obsolete for current modeling purposes.

The dataset contains transactional features like amounts, direction of transactions and geographical features. Alert specific information like monitoring scenario type and risk typology that was identified.

To prevent potential bias in model decision making, personally identifiable information was deliberately limited. Only essential person-related features were retained, while nonpredictive demographic data (such as date of birth) was excluded to ensure the model focuses on behaviorally relevant transaction patterns rather than customer demographics.

Before performing the train/test split, categorical features were encoded using Label encoding and numerical features were scaled using MinMax Scaler.

3.4 Feature selection

The initial feature selection for the training dataset was guided by domain-specific AML expertise. Drawing on nearly three years of experience as an AML analyst specializing in transactional pattern analysis—with approximately 10,000 customer profiles reviewed manually during this period, author developed comprehensive insights into relevant predictive factors. Selection criteria focused on features that provide meaningful context to monitoring specialists in their decision-making process. These features effectively characterize both transaction attributes and client profiles, offering sufficient information for alert evaluation. This approach was deliberately chosen to align the model's decision-making framework with that of human monitoring specialists, ensuring the algorithm considers the same contextual elements that inform human expert judgment.

3.5 Model dictionary and performance metrics

In the initial phase of model selection and evaluation, a dictionary of machine learning models (see Table 1) was compiled to obtain preliminary results with default parameters for most.

Table 1: Machine Learning Algorithms

Model Name	**Implementation**
Random Forest	RandomForestClassifier(n_estimators=100, random_state=0)
K-Nearest Neighbors	KNeighborsClassifier()
Support Vector Machine	SVC()
Logistic Regression	LogisticRegression()
Decision Tree	DecisionTreeClassifier()
XGBoost	XGBClassifier(use_label_encoder=False, eval_metric='logloss')
Gradient Boosting	GradientBoostingClassifier()
AdaBoost	AdaBoostClassifier()
Naive Bayes	GaussianNB()
Nearest Centroid	NearestCentroid()
Extra Trees	ExtraTreesClassifier()

To get a baseline performance of the models, two fundamental metrics were selected. The accuracy and recall score of the test set results served as preliminary indicators of the model effectiveness. Although the dataset is highly imbalanced, meaning that high accuracy values exceeding 96% do not necessarily indicate optimal performance, when combined with with recall these metrics still provide valuable preliminary insights into each model's performance on the test set. Recall provides particularly valuable information about correctly identified positive alerts. As shown in the formula below, when a model misclassifies actual positive results as negative, the recall value decreases significantly, serving as a good indicator of the model's ability to identify positive cases. Even though we are focusing on noise reduction, decreasing the number misclassified TP alerts is still one of the priorities for the model.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} = \frac{TP}{TP + FN} \tag{1}$$

Additionally, for best performing models, the Receiver Operating Characteristics (ROC) curve will be plotted and the Area Under the Curve (AUC) scores will be calculated. Although for highly imbalanced data the ROC AUC plots are not considered to be the best indicators of model performance [19] the author believes that there may still be some benefits to include those plots when reviewing the results of the initial training. In addition to these metrics, the confusion matrix serves as an important tool for deriving valuable insights about model performance. As shown in Figure 4, a confusion matrix clarifies how many predictions of each class the model classified correctly and incorrectly. This visualization provides a more comprehensive

overview of the model's actual performance by counting the occurrences of each prediction type (True Positives, True Negatives, False Positives, and False Negatives), enabling a more detailed assessment of classification effectiveness [20].

	Pred Neg	Pred Pos
Act Neg	100	4
Act Pos	11	20

Figure 4: Example confusion matrix

3.6 Hyperparameter optimization and autoML

In order to maximize the performance of the models, Optuna was selected for hyperparameter optimization. Optuna is a modern hyperparameter optimization framework that makes machine learning model tuning easier and more efficient. Unlike older frameworks, Optuna lets users build their search space dynamically while the program runs, rather than having to define everything upfront. It combines effective search methods with pruning capabilities that automatically stop underperforming trials early to save time. The framework is designed to be simple to use, working equally well for quick experiments on a laptop or large-scale distributed computing. Optuna requires minimal setup while outperforming many competing optimization tools. It has proven successful in real applications ranging from competition-winning object detection models to optimizing database and multimedia encoding parameters [21]. Table 2 describes the parameter search space that was defined for the experiment.

Table 2: Compact summary of hyperparameter search space used in Optuna optimization

Model	Parameter Search Space
XGBoost	n_estimators: [50, 500], max_depth: [3, 15], learning_rate: [0.01, 0.3], subsample: [0.5, 1.0], colsample_bytree: [0.5, 1.0], gamma: [0, 5], reg_alpha: [0, 5], reg_lambda: [0, 10], min_child_weight: [1, 10], scale_pos_weight: [1.0, 10.0]
Random Forest	n_estimators: [50, 500], max_depth: [3, 20], min_samples_split: [2, 10], min_samples_leaf: [1, 5], max_features: {sqrt, log2}, class_weight: balanced
Decision Tree	max_depth: [3, 20], min_samples_split: [2, 10], min_samples_leaf: [1, 5], criterion: {gini, entropy}, class_weight: balanced
Extra Trees	n_estimators: [50, 500], max_depth: [3, 20], min_samples_split: [2, 10], min_samples_leaf: [1, 5], max_features: {sqrt, log2}, class_weight: balanced

In addition to Optuna, an automated machine learning (AutoML) solution was selected to find a best performing model without having to conduct manual hyperparameter optimization. TPOT (Tree-based Pipeline Optimization Tool) is an AutoML system that automatically designs and optimizes machine learning pipelines using genetic programming. Unlike traditional machine learning approaches that require considerable expertise and manual tuning, TPOT can discover effective combinations of preprocessing steps and machine learning algorithms without human intervention. The system treats machine learning components as genetic programming primitives and constructs tree-based pipelines as shown in Figure 5, that can process data through multiple pathways before making classifications. TPOT uses a multi-objective optimization approach that aims to maximize accuracy while minimizing pipeline complexity [22].

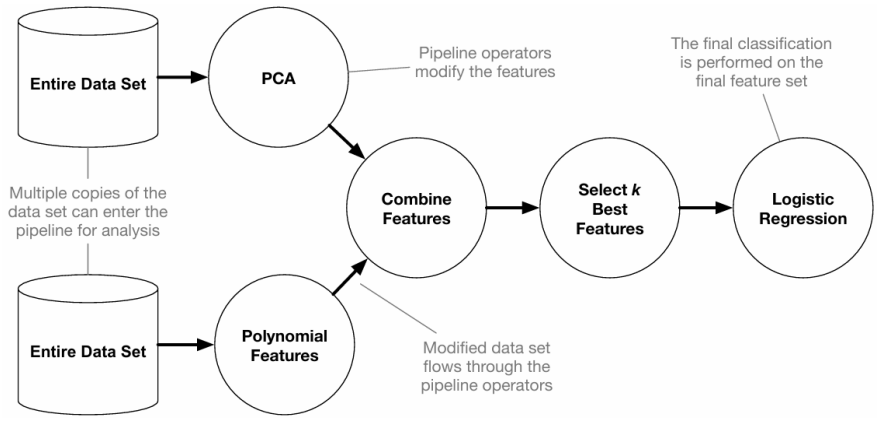


Figure 5: Example tpot pipeline. source: tpot documentation

3.7 Synthetic data generation methods

Since the dataset is highly imbalanced and the number of samples is relatively small, potentially limiting model performance, several techniques were selected to enhance dataset quality. These enhancement methods were implemented with the goal of improving models' generalization abilities on unseen data.

In order to address class imbalance Synthetic minority oversampling technique (SMOTE) was selected. SMOTE is an alternative to traditional over-sampling methods. Rather than simply duplicating minority class examples (which can lead to overfitting), SMOTE creates synthetic examples by operating in "feature space". For each minority class sample, SMOTE generates new synthetic samples along the line segments joining the sample to its k nearest neighbors. This approach helps to form broader decision regions for the minority class [23].

Undersampling is a technique that is used to remove some of the less meaningful data from training in hopes of balancing the classes. It is a useful technique when faced with large datasets with class imbalance. However, since we are trying to reduce FP noise and not misclassify actual TP alerts, undersampling was not considered since the further reduction of samples would reduce the number of FP alerts in the dataset that would probably reduce the models ability to classify FP-s with high probability where necessary and shift the focus towards stronger predictions for TP samples.

Using variational autoencoders or VAE-s was also considered for this research, since this technology allows for a more complex oversampling of the training data, and additionally might offer some insights into the data itself.

A Variational Autoencoder is a generative deep learning model that learns to compress and reconstruct data by mapping inputs to a compressed latent space representation. It consists of two main neural networks: an encoder and a decoder. The encoder takes an input and transforms it into a probabilistic representation in a lower-dimensional latent space. Instead of producing a single point, it generates a probability distribution (typically a Gaussian) with a mean and standard deviation. This allows the model to capture the underlying variability in the data [24].

For a useful visualization of the concept of latent variables, imagine a bridge with a sensor that measures the weight of each passing vehicle. Naturally, there are different kinds of vehicles that use the bridge, from small, lightweight convertibles to huge, heavy trucks. Because there is no camera, we have no way to detect if a specific vehicle is a convertible, sedan, van or truck. However, we do know that the type of vehicle significantly influences that vehicle's weight. This example thus entails two random variables, x and z, in which x is the directly observable variable of vehicle weight and z is the latent variable of vehicle type. The primary training objective for any autoencoder is for it to learn how to efficiently model the latent space of a particular input [25].

During training, the model samples a point from this distribution and passes it to the decoder, which attempts to reconstruct the original input. The training process simultaneously op-

timizes two objectives: reconstruction accuracy and keeping the latent space distribution close to a standard normal distribution. This approach has two key benefits: it learns a compressed, meaningful representation of the data, and it allows for generating new data by sampling from the learned latent space distribution. For example, in image generation, you can sample points from the latent space and use the decoder to create new, synthetic images that look similar to the training data [24].

During training, two primary metrics are monitored to assess the VAE's performance: reconstruction loss and Kullback-Leibler (KL) divergence. The reconstruction loss quantifies how well the decoder is able to reproduce the input data from the latent representation. It is often measured using Mean Squared Error (MSE) or Binary Crossentropy, depending on the nature of the input data. Lower reconstruction loss indicates that the model is accurately capturing the key features of the input. KL divergence, on the other hand, measures how much the learned latent distribution deviates from a standard normal distribution, which is the target prior. Minimizing the KL divergence ensures that the latent space remains continuous and well-structured, allowing for smooth interpolation and sampling. Together, these losses form the total VAE loss, typically with a weighting coefficient (often denoted as β) applied to the KL term to control the trade-off between accurate reconstruction and latent space regularization. A balanced combination of these metrics is essential for producing high-quality synthetic data that maintains the statistical properties of the original dataset while enabling better generalization for downstream tasks [26].

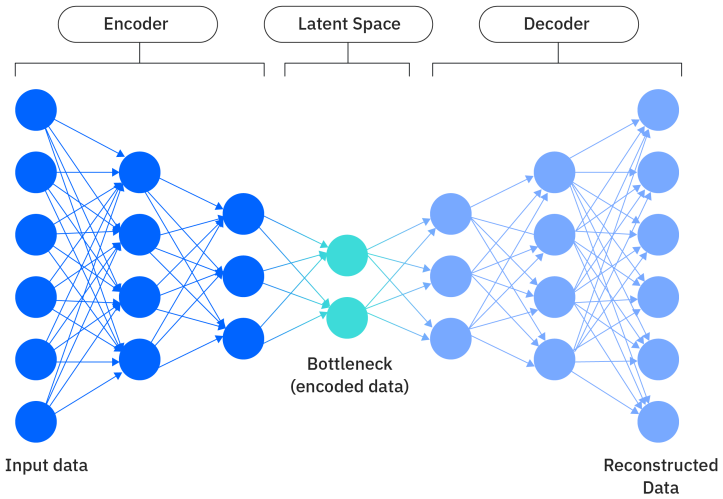


Figure 6: Visual depiction of the architecture of a standard autoencoder neural network. source: https://www.ibm.com/think/topics/variational-autoencoder

Two experiments will be conducted with the VAE-s and architectures used in the experiments will follow a standard encoder–decoder setup with symmetrical dense layers. Table 3 describes the architecture of a first VAE that will be trained on the whole dataset and used for dataset enrichment, its encoder comprised three fully connected layers with 128, 64, and 32

neurons respectively, each using ReLU activation functions, followed by two separate dense layers to parameterize the latent variables: z_mean and z_log_var. A latent dimensionality of 20 was chosen to allow sufficient expressiveness while remaining compact and a batch size of 64 was selected.

The decoder mirrored the encoder, with a similar dense layer structure and a tanh activation at the output layer to map the reconstruction back into the normalized input space. To stabilize training and improve generalization, several optimization strategies were employed: L2 kernel regularization was applied to all dense layers ($\lambda = 10^{-5}$), batch normalization was used before the output layer, and dropout (rate = 0.1) was added to the encoder to prevent overfitting. The model was trained using the Adam optimizer with a learning rate of 10^{-4} and gradient clipping (clipnorm = 1.0) to prevent exploding gradients. The total loss was computed as the sum of the mean squared reconstruction loss and a weighted KL divergence term, with the KL weight (β) set to 0.5 to balance compression and reconstruction quality. This model will be used to enhance the original dataset by adding the 20-dimensional latent vectors as new features alongside the original ones, helping classification models detect patterns that might not be visible in the raw data alone.

Table 3: Architecture of the Variational Autoencoder (VAE) used for full-data enrichment

Component	**Layer Description**	**Activation**
	Encoder	
Input Layer	Input features (standardized)	–
Dense Layer 1	128 neurons	ReLU
Dropout	Dropout layer (rate = 0.1)	–
Dense Layer 2	64 neurons	ReLU
Dense Layer 3	32 neurons	ReLU
Latent Mean	Dense layer with 20 units (z_mean)	Linear
Latent Log Var	Dense layer with 20 units (z_log_var)	Linear
	Latent Space	
Sampling	$z = z\_mean + \exp(0.5 \cdot z\_log\_var) \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$	–
	Decoder	
Input Layer	Latent vector (z)	–
Dense Layer 1	32 neurons	ReLU
Dense Layer 2	64 neurons	ReLU
Dense Layer 3	128 neurons	ReLU
Batch Norm	Batch Normalization	–
Output Layer	Dense layer (same size as input features)	Tanh

The second VAE model will be trained on the minority class exclusively in order to generate synthetic samples of the same class to augment training data with generated samples. A smaller batch size (batch_size = 16) was used to increase the number of updates per epoch and introduce more gradient noise, which is beneficial in low-data settings. The VAE was trained for up to 200 epochs with early stopping based on the validation loss.

As shown in Table 4 the minority-class VAE uses the same general encoder–decoder structure as the full-data VAE, but with key adaptations for the low-data regime. In addition to architectural and loss-based regularization, input-level noise is introduced during training to improve generalization. Specifically, Gaussian noise with a standard deviation of $\sigma = 0.05$ is added to the standardized input features, effectively training the model as a denoising VAE. The VAE was tasked with reconstructing the clean input from its noisy counterpart, encouraging the encoder and decoder to learn more robust, noise-invariant representations. This form of input corruption acts similarly to dropout by preventing the model from over-relying on any single feature and should be especially beneficial in the low-data regime of the minority-class VAE.

The encoder consisted of dense layers with 128, 64, and 32 units using ReLU activations, followed by separate dense layers to estimate the latent mean (z_mean) and log-variance (z_log_var) vectors. The latent dimensionality was set to 10 to preserve generative flexibility. Regularization was applied through L2 kernel penalties (λ = 1e-5) and dropout (rate = 0.1) in the encoder, along with batch normalization in the decoder to promote smooth output distributions. The decoder mirrored the encoder and ended with a tanh activation function, which was appropriate given the standardized input range.

Table 4: Architecture of the Variational Autoencoder (VAE) trained exclusively on the minority class with input noise regularization

Component	Layer Description	Activation
	Encoder	
Input Corruption	Additive Gaussian Noise ($\sigma = 0.05$)	–
Input Layer	Minority-class features (standardized)	–
Dense Layer 1	128 neurons, L2 regularization ($\lambda = 10^{-5}$)	ReLU
Dropout	Dropout layer (rate = 0.1)	–
Dense Layer 2	64 neurons, L2 regularization	ReLU
Dense Layer 3	32 neurons, L2 regularization	ReLU
Latent Mean	Dense layer with 10 units (z_mean), L2 regularization	Linear
Latent Log Var	Dense layer with 10 units (z_log_var), L2 regularization	Linear
	Latent Space	
Sampling	$z = z\_mean + \exp(0.5 \cdot z\_log\_var) \cdot \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$	–
	Decoder	
Input Layer	Latent vector (z)	–
Dense Layer 1	32 neurons, L2 regularization	ReLU
Dense Layer 2	64 neurons, L2 regularization	ReLU
Dense Layer 3	128 neurons, L2 regularization	ReLU
Batch Norm	Batch Normalization	–
Output Layer	Dense layer (original feature size), L2 regularization	Tanh

3.8 Classification thresholding

In order to minimize the risk of misclassifying actual true positive alerts, a thresholding mechanism will be implemented in the framework. Binary classification models typically output a prediction of class 0 if the prediction probability is < 0.50 or 1 if the probability is > 0.50. However, when a model classifies an alert as a true positive with a probability of 0.56 for example, it demonstrates limited confidence in this classification—essentially comparable to a random coin flip [27]. While there is minimal risk when a model predicts a true positive with low probability since such alerts will undergo manual review by specialists, the concern arises when a model predicts a false positive with similarly low confidence. In such cases, the model's uncertainty indicates a significant risk of missing genuinely suspicious transactions if these alerts are excluded from manual review based solely on the model's prediction. Luckily, we can tackle this issue by using models built in features that allow us to output the prediction probability using models built in function predict_proba, rather than a binary output of 0 or 1. The proposed

approach is to implement a classification thresholding mechanism that only classifies alerts as false positives when the model demonstrates high confidence in its prediction for example >95% confidence. This approach ensures that borderline cases receive appropriate human attention, thereby minimizing the risk of overlooking true positive alerts in the AML monitoring system.

3.9 Explainability

In the field of AML, explainability represents a critical component when developing machine learning models for alert classification tasks. As previously established, regulatory requirements mandate that all decisions generated by such models must be transparent and interpretable. To satisfy these requirements, the author will employ following explainability tools:Local Interpretable Model-agnostic Explanations (LIME) analysis, Shapley Additive explanations (SHAP) values, feature importance maps. These techniques will provide insights into the model's decision-making process, ensuring that classification outcomes can be effectively communicated to both internal stakeholders and regulatory authorities.

3.9.1 LIME Analysis

LIME is a explanation technique that provides interpretable and faithful explanations for the predictions of any classifier [28]. The key insight of LIME is to explain predictions locally by learning an interpretable model around individual predictions, rather than attempting to explain the entire model globally.

The approach works by perturbing the input data and seeing how the predictions change. LIME creates a new dataset consisting of perturbed samples and their corresponding predictions from the complex model. It then trains a simple interpretable model (such as a linear regression model) on this new dataset, weighted by the proximity of the perturbed samples to the original instance. This local model serves as an explanation for the specific prediction.

LIME's model-agnostic approach allows it to explain predictions from any classifier or regressor, regardless of how complex the underlying model might be. Experiments demonstrate that LIME explanations help users build appropriate trust in machine learning models by allowing them to understand why individual predictions are made, choose between competing models and improve untrustworthy classifiers through feature engineering. [29].

Lime was selected because this technique is easy to explain to non technical stakeholders and gives a good overview of selected features contributions to each decision.

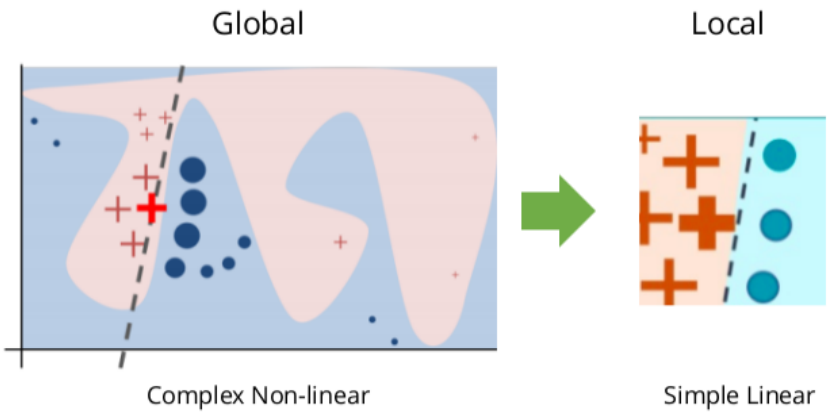


Figure 7: Lime local explanations visualization. source: https://c3.ai/

LIME helps explain complex machine learning predictions by locally sampling data points near an instance of interest and creating an interpretable model that approximates the local behavior of the original complex model.

3.9.2 SHAP Values and feature importance maps

SHAP (SHapley Additive exPlanations) is a unified framework for interpreting model predictions that assigns each feature an importance value for a particular prediction [30].

SHAP values are based on Shapley values from cooperative game theory and represent the only solution that satisfies three desirable properties: local accuracy (ensuring the explanation matches the original model's output), missingness (features absent in the original input have no impact), and consistency (if a feature's contribution increases in the model, its attribution should not decrease) [30].

SHAP values explain how to get from the base value (what would be predicted if we didn't know any features) to the current output by attributing the change to each feature. What makes SHAP unique is that it unifies six existing feature attribution methods under one approach and provides a solid theoretical foundation for interpreting complex models [30].

The framework includes several implementation methods for calculating SHAP values, including model-agnostic approaches like Kernel SHAP as well as model-specific methods like Deep SHAP for neural networks. Studies show that SHAP values tend to be more aligned with human intuition when explaining model predictions compared to other methods [31]. That is the main reason why SHAP values have been selected for this study - while the underlying calculations might be complex to explain to stakeholders, the simplicity of the final representation allows everyone to understand how features contribute to model decisions in individual cases. A good complement to SHAP for understanding overall feature impact is feature importance maps, which are built into models like XGBoost. These maps describe the magnitude of contribution each individual feature makes to model decisions across the entire dataset.

4 Analysis of the results

4.1 Baseline models

In order to get a first overview of models performances on the alert data, the models from Table 1 were trained on the cleaned, scaled and encoded dataset using default parameters. Dataset itself was split as follows: 80% for training and 20% for testing. The results of the initial training were then aggregated in Table 5 and used for preliminary analysis to identify underperforming models. As can be seen in the table below, ExtraTrees had the best performance on the test set with the highest recall and test accuracy. By looking at these results four best performing models were selected for further review based on their recall and a review of a confusion matrix, those models are ExtraTrees, RandomForest, XGBoost and DecisionTree. Even though 3 of those models scored 100% on the training dataset, which is an indication of overfitting, the initial test results are valuable as they highlight the need for further model tuning to improve generalization.

Table 5: Initial Model Performance Comparison

Model	**Test Accuracy (%)**	**Train Accuracy (%)**	**Recall**	**AUC**
ExtraTrees	98.14	100.0	0.65	0.95
RandomForest	98.04	100.0	0.55	0.94
XGBoost	97.97	99.7	0.53	0.94
DecisionTree	96.52	100.0	0.60	0.80
KNN	97.21	97.78	0.5	N/A
SVM	96.63	96.05	0.0	N/A
LogisticRegression	96.63	96.1	0.0	N/A
GradientBoosting	97.01	97.1	0.18	N/A
AdaBoost	96.59	96.01	0.0	N/A
NaiveBayes	92.74	92.42	0.04	N/A
NearestCentroid	50.71	49.51	0.66	N/A

In addition to the metrics presented in Table 5, the ROC AUC visualization was used to better understand the performance of the model. While RandomForest, XGBoost, and ExtraTrees achieved nearly identical AUC scores (0.94-0.95) (see Figure 8, the DecisionTree model scored significantly lower (0.80) despite having better recall than both RandomForest and XGBoost. This performance gap can be attributed to the fundamental limitations of the Decision Tree algorithm. Unlike ensemble methods that create complex decision boundaries through multiple trees, a single Decision Tree makes hard splits that result in more rigid decision boundaries and less nuanced probability estimates. Furthermore, on high-dimensional, imbalanced data typical

in AML contexts, a single tree struggles to generalize effectively, a hypothesis supported by its 100% training accuracy indicating overfitting.

[H]

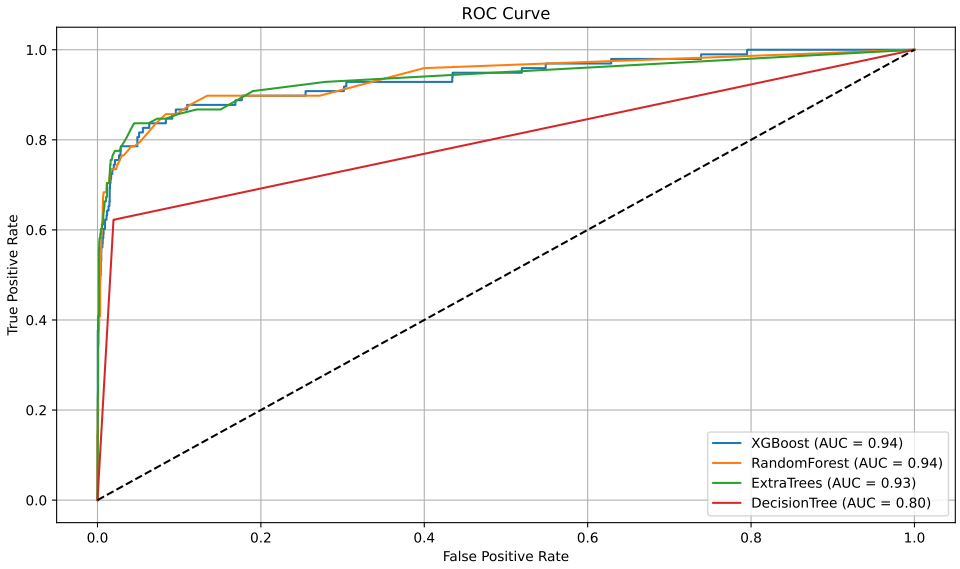
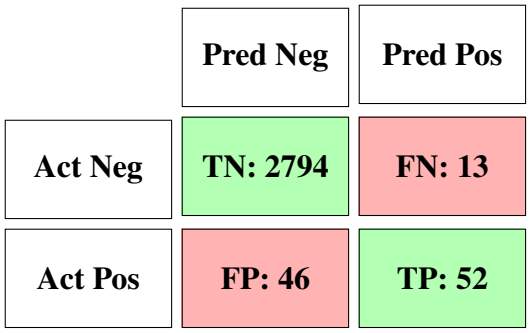
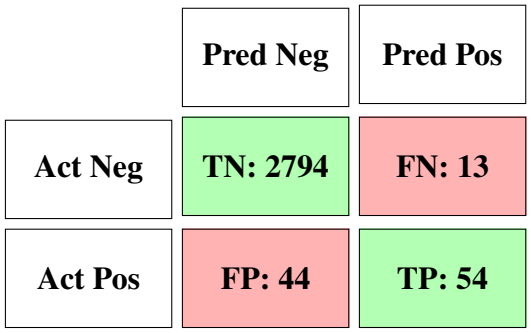


Figure 8: ROC AUC curves for the best performing models

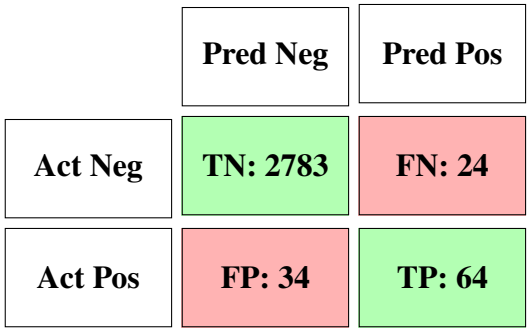
Since avoiding the misclassification of TP-s as FP-s is critical in the field of AML, confusion matrices and manual review of incorrectly predicted TP-s served as the primary performance indicators. As shown in Figure 9, the ExtraTrees model demonstrated best performance in predicting true positives.



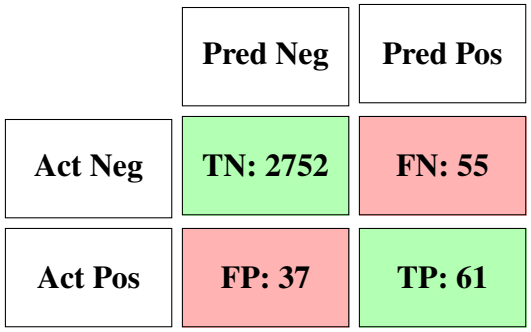
(a) XGBoost (Acc: 97.97%)



(b) Random Forest (Acc: 98.04%)



(c) ExtraTrees (Acc: 98.00%)



(d) DecisionTree (Acc: 96.83%)

Figure 9: Confusion matrices for the best performing models

4.2 Synthetic data generation and baseline results

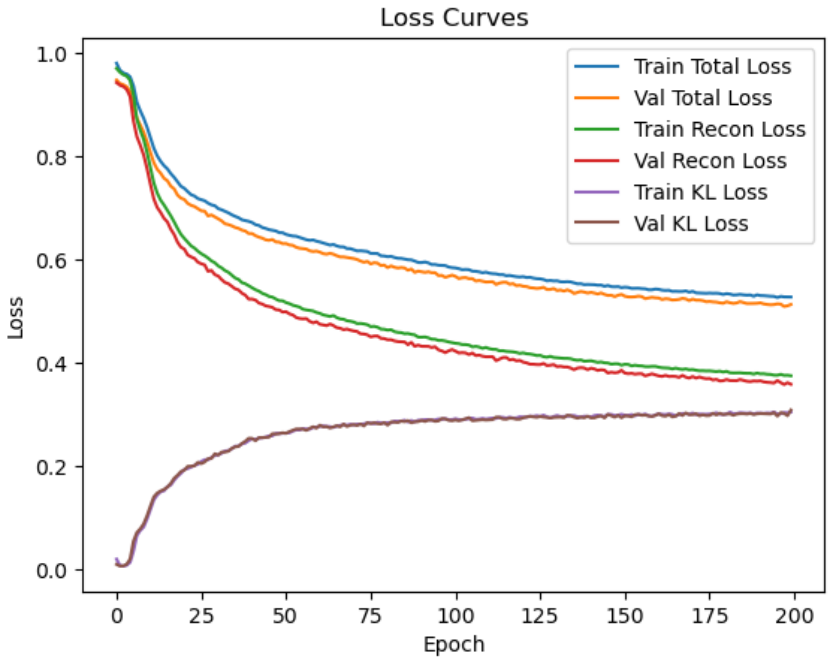
To mitigate the significant class imbalance present in the dataset where the minority class represents less than 5% of the total observations, three augmentation strategies were implemented and evaluated aimed at enhancing classifier performance: SMOTE-based oversampling, Variational Autoencoder augmentation, and minority-targeted VAE augmentation.

The first approach applied was the Synthetic Minority Over-sampling Technique (SMOTE). In total 1000 true positive alert samples were generated using SMOTE and added to the training dataset and no false positive alert samples were generated to avoid introducing additional noise.

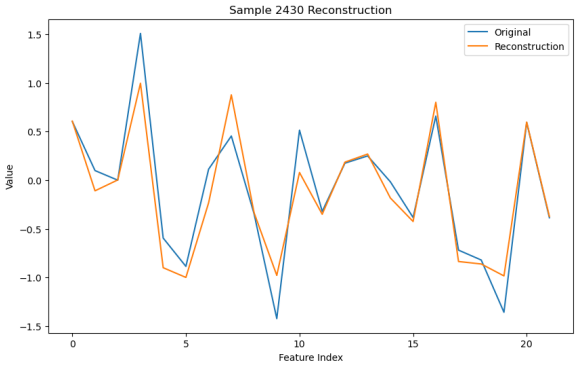
To explore generative modeling for data enrichment, a VAE was initially trained on the entire preprocessed dataset, which included both majority and minority class instances. The objective of this VAE was to learn a general, class-agnostic latent representation of the full feature space. These learned latent representations were then appended to the original dataset as new features, with the expectation that models would be able to extract additional information from these enriched feature vectors.

The model was trained for up to 200 epochs with early stopping and a live loss monitor. Loss curves indicated stable convergence: the reconstruction loss decreased smoothly throughout training, while the KL divergence increased steadily without signs of posterior collapse as shown in Figure 10 (a). Importantly, the validation loss closely tracked the training loss, sug-

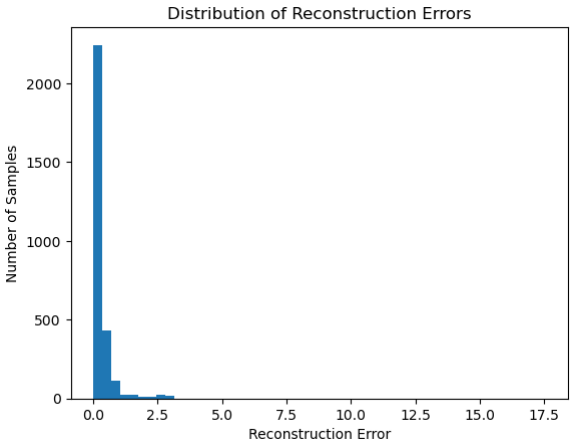
gesting minimal overfitting and good generalization to unseen data. For example, Figure 10 (b) and (c) demonstrate the reconstructed and original values of one randomly selected sample and the overall distribution of reconstruction errors among test samples. As can be seen the decoder was able to reconstruct the samples fairly close to the original from latent space representations. Even though this model was used to only enrich the dataset with latent representations it is still interesting to note how well it was able to reconstruct most of the dataset.



(a) Training history of the VAE



(b) Reconstruction of sample 2430



(c) Histogram of reconstruction errors

Figure 10: Training results of the VAE trained on the whole dataset

Additionally, in order to better understand the structure and quality of the latent space learned by the VAE, a Principal Component Analysis (PCA) was applied to the latent representations produced by the encoder. Since the latent space is high-dimensional (20 dimensions in our configuration), PCA was used to project these representations into a two-dimensional

space while preserving as much of the variance in the data as possible. This dimensionality reduction technique allows for visual inspection of the encoded data distribution, providing insight into how well the model has organized the input space into a latent representation.

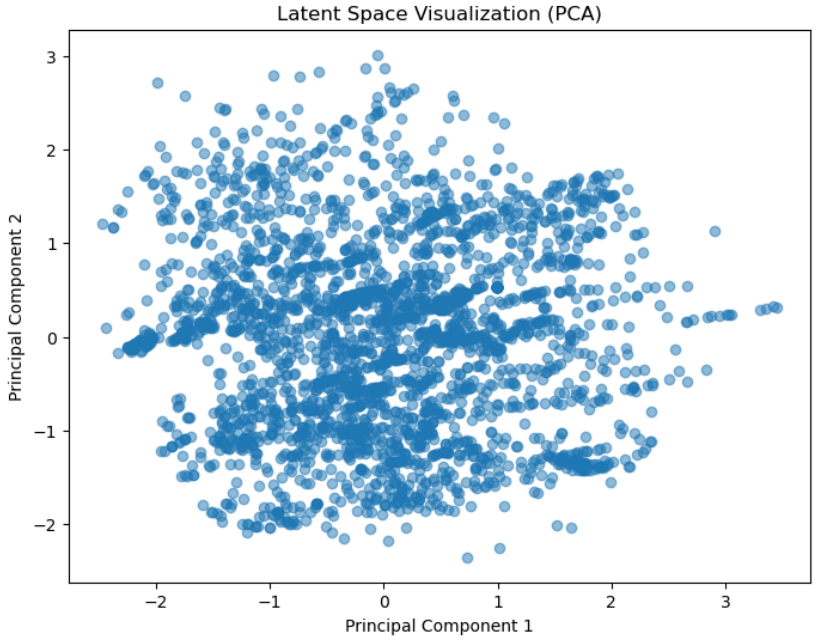
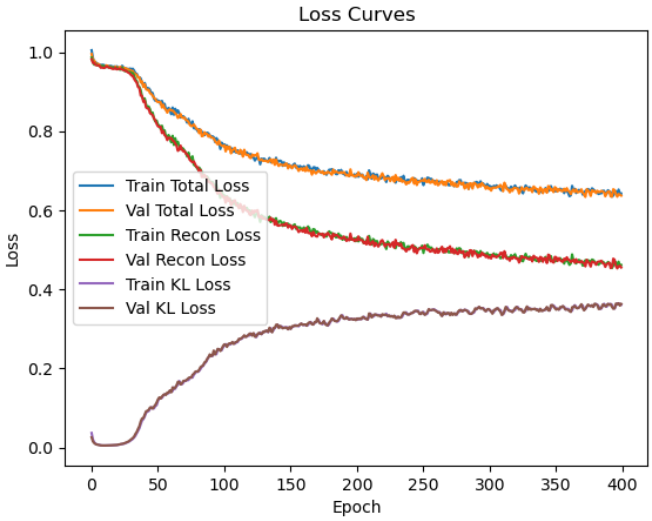


Figure 11: PCA plot of the latent space

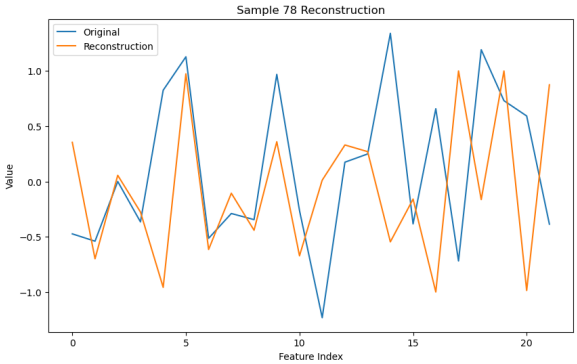
The resulting scatter plot reveals that the latent vectors form a dense, continuous cluster around the origin, which is consistent with the assumption of a standard normal prior $\mathcal{N}(0, I)$ imposed on the latent space during training. The absence of large outliers or fragmented sub-clusters suggests that the VAE has successfully learned a meaningful and stable encoding of the input features. This structure is important because it ensures that new samples drawn from the latent prior will be decoded into valid, coherent feature vectors. In other words, the quality of the learned latent space directly affects the quality and diversity of the synthetic samples generated by the decoder. Therefore, this PCA visualization serves as both a diagnostic tool for model evaluation and a visual confirmation of the VAE's ability to generalize over the feature distribution of the input data. Additionally this PCA plot can be used to find outliers in the underlying training set. If it would have shown distinct clusters it might mean that the model sees an underlying pattern and is able to distinguish common feature sets that could then be used as an insight to generate new AML rules.

To generate synthetic data that more accurately reflects the characteristics of the minority class, a second VAE was trained on a dataset that only contained true positive alert samples. This class-specific generative modeling approach was intended to address the limitations of the original, class-agnostic VAE by focusing solely on the minority class distribution. The dataset for this model included only TP 557 samples. Due to the small sample size, careful architectural choices and regularization techniques were employed as described in methodology section to ensure stable training and prevent overfitting.

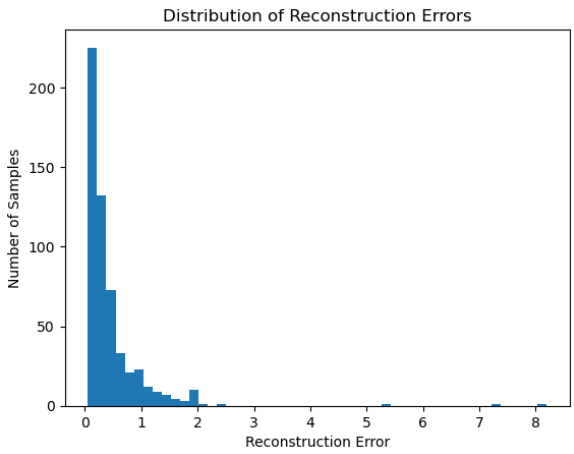
Loss curves from training shown in Figure 12 (a) indicated effective convergence. The reconstruction loss decreased steadily, and the KL divergence term increased gradually, suggesting that the model successfully learned a meaningful latent representation of the minority class. Importantly, the training and validation losses remained closely aligned, indicating that the model generalization was decent despite the small dataset size. Even though the histogram shows that reconstruction error distribution in Figure 12 (c) is a bit worse than the previous VAE, it is still considered to perform well enough taking into account the size of the dataset. In addition reviewing the reconstruction charts of random samples Figure 12 (b) confirms that the model is mostly able to generate similar samples close enough to the original. This minority-class VAE was used to generate 1000 synthetic samples, which were added to the original training set.



(a) Training history of the VAE



(b) Reconstruction of sample 78



(c) Histogram of reconstruction errors

Figure 12: Training results of the VAE on minority class dataset

To evaluate the quality and structure of the latent space learned by the VAE trained exclusively on minority-class samples, PCA was applied to the latent representations. This projec-

tion reduced the 10-dimensional latent vectors to two principal components, enabling visual interpretation of the latent space distribution. Importantly, this visualization includes both the original minority-class samples (encoded by the VAE encoder) and synthetic samples (generated by decoding vectors sampled from the standard normal prior $\mathcal{N}(0, I)$). Including both real and synthetic samples in the same latent space projection provides a direct way to assess how well the model has learned to generalize the underlying minority-class distribution.

The resulting PCA plot (Figure 13) shows a clear and compact clustering of both real and synthetic data points centered around the origin. This is consistent with the use of a standard normal prior during training, which encourages the encoded latent vectors to follow a Gaussian distribution centered at zero. The overlap between real and synthetic samples in this space suggests that the decoder successfully generates samples that are statistically similar to the training distribution. Additionally, the synthetic samples are reasonably well-dispersed with some outliers that are considered normal since about 5% of values in a standard normal distribution fall beyond ± 2 in any one dimension. This indicates a robust mapping from latent space to feature space. The inclusion of real samples in the plot is essential in this case because the VAE was trained solely on minority-class data, and the validity of synthetic data generation depends on the extent to which the decoder reproduces the true data distribution. Thus, this visualization serves as both a sanity check and an empirical confirmation that the VAE's generative capabilities are grounded in the actual structure of the minority class.

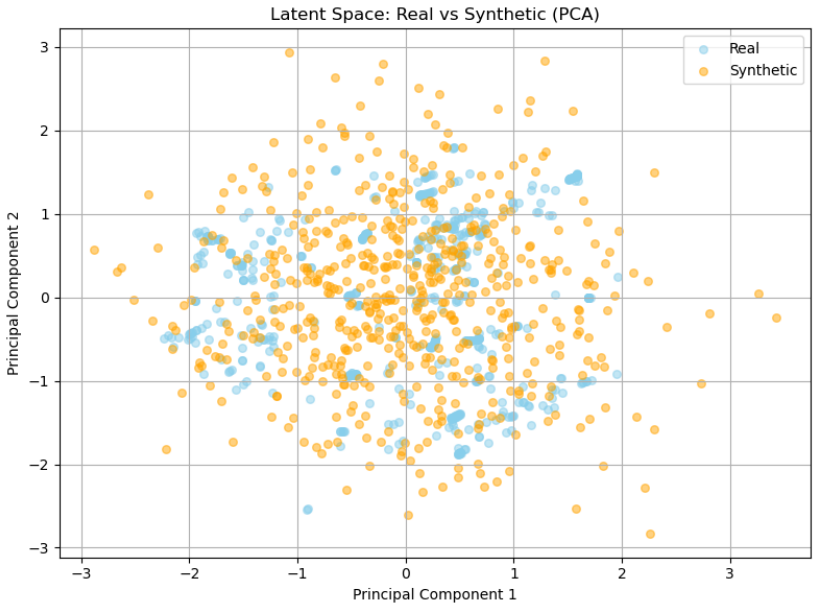


Figure 13: PCA plot of the latent space

This progression—from SMOTE to class-agnostic VAE augmentation, and finally to class-specific generative modeling—demonstrates the increasing effectiveness of tailored generative methods in improving minority-class representation without sacrificing performance on the majority class. Table 6 shows the model performances using feature-enriched (as explain in chap-

ter 3.7), SMOTE-Augmented, and VAE Minority-Augmented datasets. The best recall was achieved on the SMOTE-Augmented dataset by the ExtraTrees model, however test accuracy scores indicate that models trained on minority augmented VAE dataset had the best overall performance.

Model	Feature-Enriched				SMOTE-Augmented				VAE Minority-Augmented			
	Test Acc.	Train Acc.	Recall	AUC	Test Acc.	Train Acc.	Recall	AUC	Test Acc.	Train Acc.	Recall	AUC
ExtraTrees	97.87	100.00	0.66	0.95	97.87	100.00	0.69	0.94	98.11	100.00	0.65	0.93
RandomForest	97.93	100.00	0.58	0.94	97.93	100.00	0.65	0.94	98.18	100.00	0.56	0.94
XGBoost	97.97	99.97	0.56	0.93	97.73	99.62	0.63	0.93	98.31	99.64	0.60	0.94
DecisionTree	96.90	100.00	0.58	0.78	95.90	100.00	0.66	0.82	96.59	100.00	0.60	0.79

Table 6: Comparison of Model Performance Across Feature-Enriched, SMOTE-Augmented, and VAE Minority-Augmented Training

4.3 AutoML results

Before starting tests with Optuna, tpot autoML solution was also tested on the same dataset as the baseline models. Since tpot was selected for testing because of its simplicity and ease of implementation, only some parameters were tweaked for the test by the author. Figure 14 visualizes the parameters selected for tpot and the reasoning behind them.

Parameter	Value	Rationale
generations	50	Base value is set at 100 if not changed, 50 iterations for an initial test considering the database size was selected.
population_size	40	Was selected as opposed to default 100 to ensure pipeline diversity while saving on overall computation time
cv	5	5-fold cross-validation for reliable performance estimation
scoring	f1	Selected to maximize f1 score instead of accuracy
verbosity	2	To provide sufficient progress information
early_stop	5	Terminates search if there is no improvement after 5 generations, saving computational resources and time taken for optimization

Figure 14: Selected TPOT parameters

With set parameters it took approximately 105 minutes to complete 50 generations. When trying to visualize all of the pipelines tpot tested and adding the calculation of a pareto front

which helps to find pipelines that have the best trade-offs between multiple conflicting objectives, for example model accuracy against pipeline complexity. This allows to visualize pipeline that have a good balance of complexity vs performance score which helps to reduce overfitting by selecting slightly less accurate pipeline that is much simpler [22].

By visualizing a scatterplot of the best scores obtained by each pipeline and pareto front, it was determined that tpot classifier did use the simplest pipelines in all of the 50 generations meaning that the pipelines were one step, for example the best performing pipeline consisted only of RandomForest classifier without any extra steps. This might have happened because tpot thought that the model performance was good enough because of the unbalanced data and did not explore more complex pipeline options.

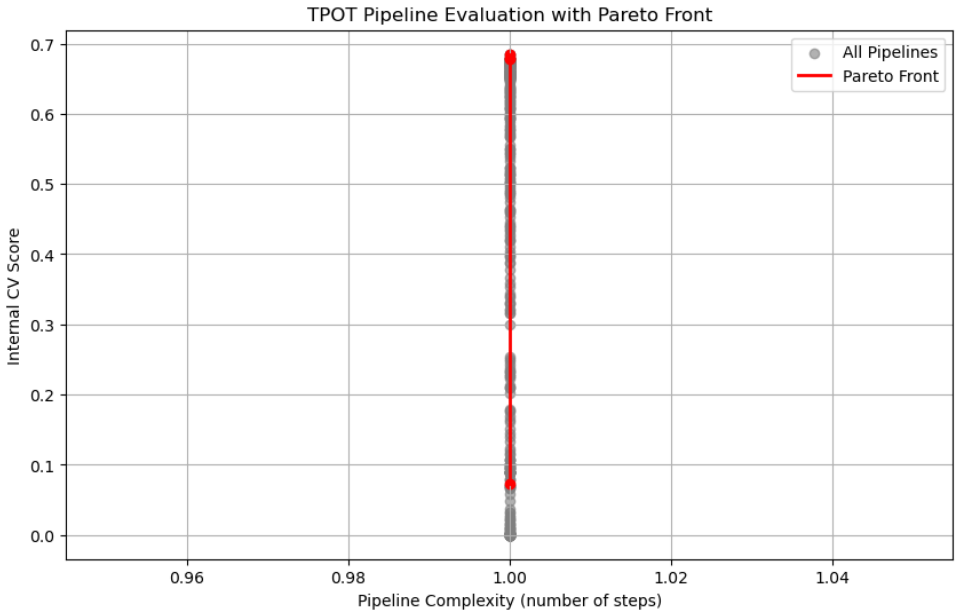


Figure 15: Results of visualizing of tpot scores with pareto front

Even though a simpler pipeline was created, tpot optimized hyperparameters for RandomForest and the resulting confusion matrix was considerably better than the baseline RandomForest model showing a decrease of 5 misclassified TP cases.

	Pred Neg	**Pred Pos**
Act Neg	**TN: 2791**	**FN: 16**
Act Pos	**FP: 39**	**TP: 59**

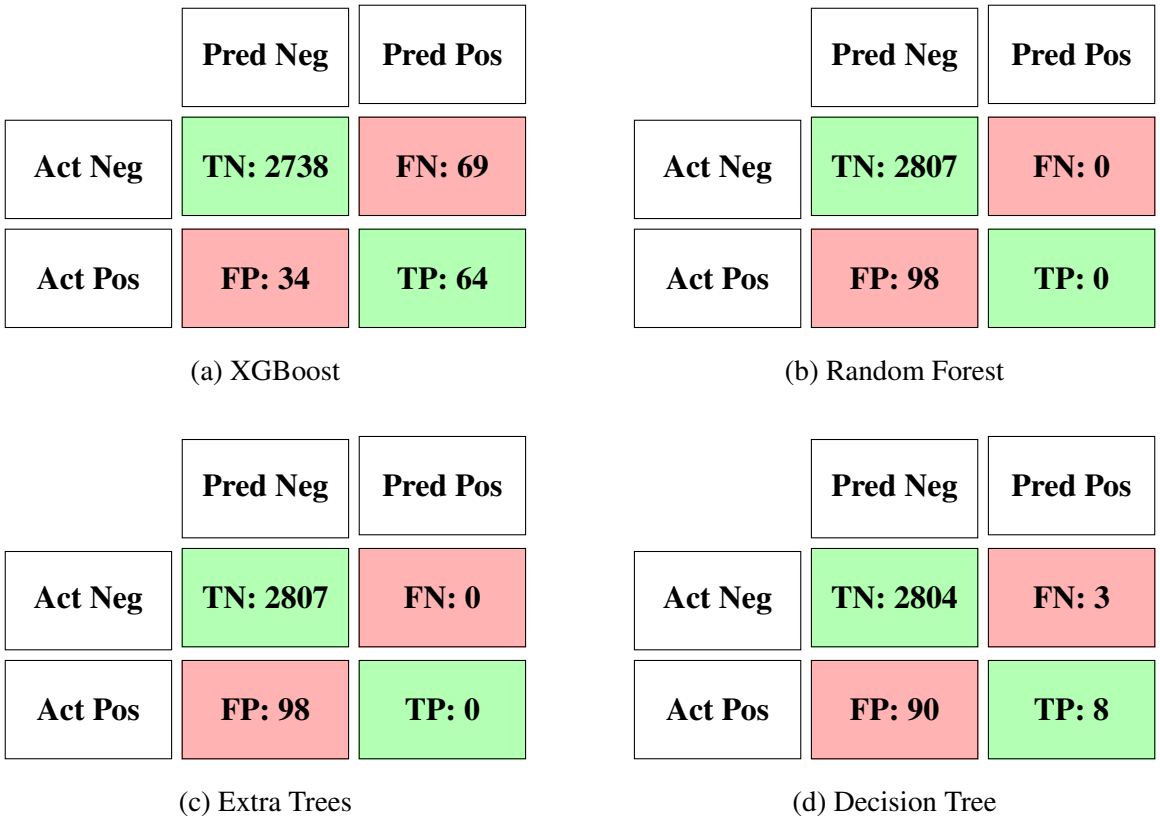
Figure 16: tpot result with RandomForest

4.4 Optuna hyperparameter optimization on baseline data

To improve baseline model performance and ensure a fair comparison with augmented strategies, hyperparameter optimization was conducted using the Optuna framework. This experiment focused on tuning the core classification models used throughout the study: XGBoost, Random Forest, ExtraTrees and Decision Tree. In total two experiments were conducted. The objective of the first optimization run was to maximize the F1-score and the second experiment tried to maximize recall, both via stratified 5-fold cross-validation on the original training data. F1 maximization was included in case recall maximization would result in overfitting. For each trial, Optuna selected a model type and proposed a set of hyperparameters, which were then evaluated using cross validation score. A total of 200 optimization trials were conducted for both experiments, each exploring a unique combination of hyperparameters within defined ranges described in methodology section.

The results from running Optuna with the objective of maximizing the F1-score showed that the best-performing model was Random Forest, achieving an F1-score of 0.68 and a recall of 0.59. In a second experiment, where the objective was changed to maximize recall, the best-performing model was a Decision Tree, reaching a recall score of 0.87. Although this result initially appeared promising, a closer inspection of the confusion matrices revealed that all models except for XGBoost were overfitting.

As visualized in Figure 17, both the Extra Trees and Random Forest models failed to identify any TP alerts correctly when recall was maximized. This behavior was most likely driven by the extreme class imbalance in the dataset, leading the models to prioritize true negatives and avoid minority class predictions entirely.



(a) XGBoost

(b) Random Forest

(c) Extra Trees

(d) Decision Tree

Figure 17: Confusion matrices of best Optuna-tuned models optimizing for recall.

In contrast, models trained under the F1-score objective performed more reasonably. However, further examination of their confusion matrices revealed that the high F1-score achieved by the Random Forest was largely due to correct classification of false negatives, a result that is less meaningful in the context of detecting actionable alerts. In comparison, XGBoost consistently performed better in identifying true positives and reducing false positives (RandomForest correctly predicted 60 TP-s while XGBoost 64), aligning more closely with the practical goals of the task.

These observations highlight the limitations of relying solely on standard evaluation metrics such as F1 or recall in highly imbalanced settings. They underscore the importance of qualitative evaluation—such as confusion matrix inspection—to understand model behavior in real-world applications.

4.5 Final model experiment

In the final experiment, only XGBoost was used for training, as it consistently demonstrated more trustworthy and stable performance across all prior experiments after manual review of prediction probabilities. To maximize its effectiveness, a combination of Optuna optimization and a VAE minority-augmented dataset was selected. The Optuna configuration for XGBoost remained the same as in earlier experiments, with F1-score set as the optimization objective and the study running for 200 iterations. The training data was augmented by adding 1000 VAE-

generated TP samples, replicating the approach used in the previous augmentation experiment where XGBoost had already shown superior performance.

Although SMOTE was considered as an alternative to VAE-based augmentation, a manual review of the predictions from the SMOTE experiment revealed a critical issue: after training on SMOTE-augmented data, XGBoost exhibited high confidence in misclassifying certain TP cases. This behavior is particularly problematic when trying to adjust thresholds for optimal alerting performance, as the model becomes biased and less flexible in boundary decision making.

In order to better understand overall model performance SHAP values were computed, LIME analysis was conducted and feature importance plots were created for the final model. LIME analysis was used only to visualize individual samples to better understand models decisions behind them, and the main reason for incorporating LIME into the review was to later present individual results to specialists because LIME calculations are a bit easier to explain.

As can be seen on the SHAP beeswarm plot some features barely contribute to the decisions of the model (narrow swarm) but some strongly affect predictions in both TP and FP directions like feature 10 for example. It is also important to understand how underlying value impacts the models decision, if red points are mostly on the right side it means that the higher underlying values steer the models decision toward TP and vice versa for blue. The SHAP plot clearly shows that some features could be left out in future training, but overall most of them contribute well to models predictions.

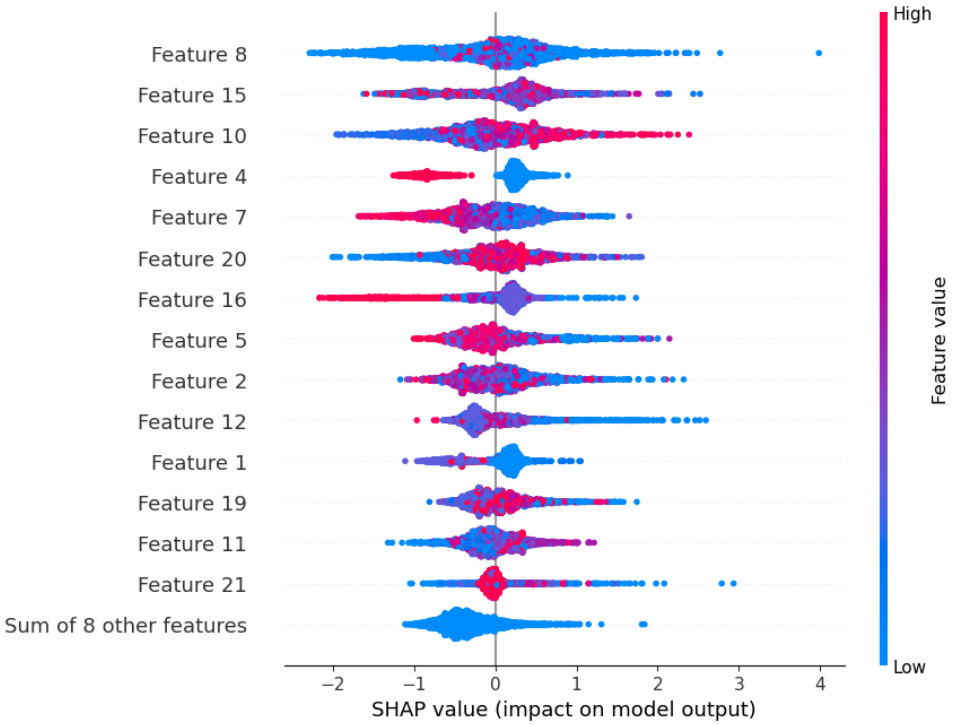


Figure 18: SHAP beeswarm plot

After completing the optimization process, the best cross-validated F1-score achieved was 0.91, representing a significant improvement over all previous results. The VAE+Optuna model demonstrated improved performance in several key metrics compared to the baseline. While recall for the TP class increased to 0.63, which was slightly lower than the Optuna-only model as shown in Table 7, the confusion matrices revealed crucial differences in prediction patterns. The Optuna-only model correctly predicted 64 TP samples compared to the VAE+Optuna model's 62, but the VAE+Optuna approach dramatically reduced false negatives from 69 to 27. As illustrated in Figure 19, this resulted in the most optimal tradeoff between identifying true positives and minimizing false positives. This also means that VAE+Optuna model predicted correctly more actual FP alerts (2780 compared to 2738), ultimately outperforming a model trained solely with Optuna tuning in overall classification effectiveness.

Table 7: Comparison of XGBoost Models Recall Performance

Model	**Recall**
Baseline XGBoost	0.53
Feature Enriched XGBoost	0.56
SMOTE Augmented XGBoost	0.63
VAE Minority Augmented XGBoost	0.60
Optuna Optimized XGBoost	0.65
Optuna + VAE XGBoost	0.63

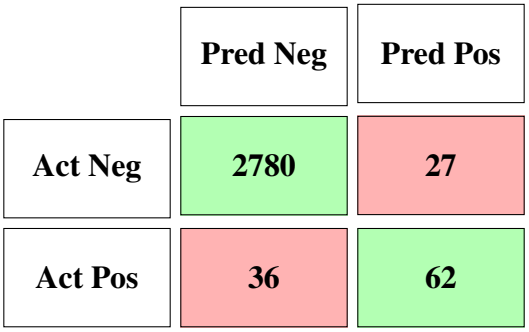


Figure 19: XGBoost trained on minority augmented VAE data and optimized with optuna

In addition to comparing metrics a manual review of the test set prediction probabilities was conducted.

XGBoost classifier demonstrated that it's predictions showed greater uncertainty in edge cases, which paradoxically improved the model's practical applicability in an AML context. Through probability threshold adjustments and manual validation, the XGBoost model enabled a significant reduction in FP classifications from 36 to just 4, while maintaining approximately 40% noise reduction in true negative results. In addition while reviewing the underlying mis-

classified TP alerts it is important to understand the risk involved, for example there is a difference in the context of underlying risk when an alert is marked as a true positive, in some cases all necessary information could not be collected about a transaction and that results in a true positive label because a specialist can not ensure that the underlying risk was mitigated. In our case all 4 alerts that would have been misclassified after probability thresholding were marked as TP by specialists precisely because not enough information was collected to mitigate the risk, meaning that these alerts could probably be FP if we had all the information about the transaction necessary.

Using probability thresholding highlights the importance of not simply optimizing for statistical metrics, but critically examining model predictions in the context of financial crime detection. The ability to introduce controlled uncertainty can be more valuable than blind confidence, especially in high-stakes domains like anti-money laundering where the cost of misclassifications can be substantial.

This final configuration demonstrates that the model is capable of significantly reducing noise while leaving more complex alerts to the specialists, effectively meeting the core objective of this work.

5 Conclusion and future work

Not only is money laundering a serious global issue but also an operational challenge for every financial institution. If we consider the fact that the number of transactions is growing every year there is no feasible way to continue transaction monitoring without implementing automated solutions. Currently the financial industry is at that critical point where pragmatic automated solutions must be implemented in order to be able to deal with the incoming workflow and comply with the regulations. This is certainly an area where machine learning and deep learning can lend a helping hand. This thesis attempts to provide a comprehensive solution on how to combat the rise of FP offline alerts in a BaaS environment by utilizing machine learning, variational autoencoders, automated machine learning solutions and hyperparameter optimization using optuna.

In the beginning of this study baseline results were calculated for 11 machine learning models in order to have a solid point to compare further results. It quickly became apparent that only 4 models out of those 11 were actually worth pursuing on the highly imbalanced BaaS AML dataset, these models were XGBoost, RandomForest, ExtraTrees and DecisionTree. Core metrics were selected to compare the models but on this dataset the only actually reliable way of interpreting the results is the use of confusion matrices and manual review of predictions since that gives the best overview of how the models actually arrive at their results.

After obtaining baseline results on the original dataset, synthetic data generation was implemented. First models were benchmarked on the dataset augmented by SMOTE, then two VAE-s were trained, one to enrich the original data with latent representations and another one to generate synthetic TP alerts to augment the original dataset with more minority class datapoints. The synthetic data benchmark revealed that the best performance was achieved by minority augmented VAE dataset on an XGBoost model showing significant increase in correctly predicted TP alerts while maintaining a optimal noise reduction of FP alerts.

Once synthetic benchmarks were set, an autoML solution was tested called tpot, the resulting best model after running tpot was RandomForest which was an improvement on the baseline results but not enough to justify it's use in the further experiments.

The final baseline was set by using Optuna on the original dataset, the 4 core models were tuned twice, once while maximizing the F1-score and once while maximizing recall. Best results were obtained while maximizing the F1-score and once again XGBoost showed the most promising results after being optimized. Even though RandomForest obtained a better F1-score the confusion matrix revealed that XGBoost classified more TP alerts correctly and was better at reducing noise of FP alerts.

For the final experiment XGBoost was trained using a combination of optuna hyperparameter optimization on the minority augmented VAE data. This combination was selected because VAE augmented data showed the most promise in the previous experiments and XGBoost was used standalone because it proved the most useful for the task of classifying alerts. The ex-

periment produced the most optimal result where XGBoost was able to classify enough TP examples while maintaining a good noise reduction threshold. After manually reviewing the predictions and applying probability thresholding the final XGBoost model achieved a 40% reduction in FP noise while only misclassifying 4 TP alerts. In addition to manual review, SHAP values were computed for the final model and LIME analysis was created for local explanations to better understand how the model arrives at its predictions. SHAP revealed that the model is not making biased decisions and is actually using relevant features to classify alerts. This might be one of the most important finding during the testing because even though the model performance is good, if it made decisions based solely on demographics for example it could not be used as that would point to overfitting by the model.

Next step is to start testing the model on previously unseen data, meaning that the trained model will be subjected to analysis on the alerts raised during a testing period of 3 months to gain confidence in models ability to generalize. In addition, model retraining is an important aspect to consider before putting it into production, since the BaaS clients transactions are very dynamic it means that the underlying transactional patterns might rapidly change, testing on newly raised alerts previously unseen by the model should give a good indication how often must this model be retrained in order to stay relevant.

This thesis demonstrates that machine learning models in combination with synthetic data augmentation produce promising results for reducing FP noise in BaaS datasets. However further experiments with synthetic data generation and larger datasets is still required to keep enhancing the models performances. In addition similar models could be expanded to other AML rule sets for example periodic alerts to assist with noise reduction. This research suggests that effective approach to mitigating the prevalence of false positive alerts is the integration of existing AML frameworks with machine learning algorithms specifically calibrated for false positive noise reduction. Such methodological combination enables AML specialists to allocate their cognitive resources toward more complex investigations, thereby enhancing the efficacy of money laundering prevention mechanisms within financial institutions.

References

[1] Karel Lannoo and Richard Parlour. Anti-money laundering in the eu: Time to get serious. Technical Report 31980, Centre for European Policy Studies, 2021.

[2] Rasmus Jensen and Alexandros Iosifidis. Fighting money laundering with statistics and machine learning. *IEEE Access*, PP:1–1, 2023.

[3] Adetoyese Omoseebi, Godwin Ola, and Jackson Tyler. Rule-based systems in aml. 2025.

[4] H. Veen, L. F. Heuts, and E. C. Leertouwer. Dutch national risk assessment on money laundering 2019. Technical report, 2020.

[5] Institute of International Finance. Machine learning in anti-money laundering – summary report, 2018.

[6] Prashant Bansal. Baas: Banking as a service brings industry disruption. *International Journal of Science and Research Archive*, 2024.

[7] Ana Isabel Canhoto. Leveraging machine learning in the global fight against money laundering and terrorism financing: An affordances perspective. *Journal of business research*, 131:441–452, 2021.

[8] Nevine Labib, Mohamed Rizka, and Amr Shokry. Survey of machine learning approaches of anti-money laundering techniques to counter terrorism finance. pages 73–87, 2020.

[9] Pavlo Tertychnyi. *Machine learning methods for anti-money laundering monitoring*. Doctoral dissertation, University of Tartu, 2023.

[10] Nazanin Bakhshinejad, Reza Soltani, Uyen Nguyen, and Paul Messina. A survey of machine learning based anti-money laundering solutions. 2022.

[11] Shijia Gao, Dongming Xu, Huaiqing Wang, and Peter Green. Knowledge-based anti-money laundering: A software agent bank application. *Journal of Knowledge Management*, 13, 2009.

[12] Ouren Kuiper, Martin van den Berg, Joost van der Burgt, and Stefan Leijnen. Exploring explainable ai in the financial sector: Perspectives of banks and supervisory authorities. pages 105–119, 2022.

[13] Zhiyuan Chen, Le Dinh Van Khoa, Ee Na Teoh, Amril Nazir, Ettikan K. Karuppiah, and Kim S. Lam. Machine learning techniques for anti-money laundering (aml) solutions in suspicious transaction detection: a review. *Knowledge and Information Systems*, 57(2):245–285, 2018.

[14] Ahmed Bakry, Almohammady Alsharkawy, Mohamed Farag, and K. Raslan. Automatic suppression of false positive alerts in anti-money laundering systems using machine learning. *The Journal of Supercomputing*, 2023.

[15] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel K. I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. Anti-money laundering in bitcoin: experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[16] Zhiyuan Chen, Waleed M. Soliman, Amril Nazir, and Md Shorfuzzaman. Variational autoencoders and wasserstein generative adversarial networks for improving the anti-money laundering process. *IEEE Access*, 9:83762–83785, 2021.

[17] Dattatraya V. Kute, Bhasker Pradhan, Neeraj Shukla, and Abdullah Alamri. Explainable deep learning model for predicting money laundering transactions. *International Journal on Smart Sensing and Intelligent Systems*, 17(1):1–8, 2024.

[18] Institute of International Finance and EY Global Services Limited. Iif and ey survey report on machine learning: Uses in credit risk and aml applications — public summary, 2022.

[19] Takaya Saito and Marc Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10, 2015.

[20] Zohreh Karimi. Confusion matrix, 2021.

[21] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework, 2019.

[22] Randal Olson and Jason Moore. Tpot: A tree-based pipeline optimization tool for automating machine learning. pages 151–160, 2019.

[23] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[25] Bergmann and Stryker. What is a variational autoencoder?, 2024.

[26] Andrea Asperti and Matteo Trentin. Balancing reconstruction error and kullback-leibler divergence in variational autoencoders, 2020.

[27] Fanous Karim. Classification models and thresholds, 2021.

[28] Damien Garreau and Ulrike von Luxburg. Explaining the explainer: A first theoretical analysis of lime. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 1287–1296. PMLR, 2020.

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[30] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[31] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.

License

Non-exclusive licence to reproduce the thesis and make the thesis public

I, **Anton Malkovski**,

1. grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the digital archives of the University of Tartu until the expiry of the term of copyright, my thesis

   Machine Learning Methods in Anti Money Laundering,

   supervised by **Kadir Aktas**;

2. grant the University of Tartu the permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work from 14.05.2025 until the expiry of the term of copyright;

3. am aware that the author retains the rights specified in points 1 and 2;

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Anton Malkovski
14.05.2025