Faculty of Economics and Business, ICADE

Entrepreneurship Final Thesis:

Machine Learning Techniques for Detecting Internal Threats in *ReceiptVault,* an imaginary FinTech startup.

(Short version)

Author: Santiago López de Toledo Soler, Data Science and International Studies

Director: Carlos Bellón Núñez-Mera

MADRID | June 2025

Abstract

This thesis analyzes the application of machine learning techniques for insider threat detection in ReceiptVault, a FinTech platform focused on digital receipt and subscription management. Based on the sensitive nature of the data handled by the application, the study examines the main cybersecurity risks and reviews the applicable regulatory framework (GDPR, DORA, ENS, and PSD2). The analysis focuses on two academic studies using the CERT r4.2 dataset, applying supervised models (neural network, random forest, SVM, K-NN, and a hybrid model). A customized cost function is applied, prioritizing the reduction of false negatives, given the criticality of such errors for ReceiptVault. Although the hybrid model shows the best numerical performance, its lack of methodological transparency leads to recommending the neural network as the most suitable model due to its robustness, consistency, and adjustability. Finally, future research lines are proposed, focusing on hyperparameter optimization and computational cost assessment.

Keywords

ReceiptVault / Machine Learning / Cybersecurity / Insider threats / FinTech / Threat detection / Supervised classification / Cost function

Index

1. Cybersecurity issues and challenges for ReceiptVault.

Although ReceiptVault does not execute financial transactions directly, it manages highly sensitive user-related data. To offer a centralized and functional experience, the platform requires authorized access to user credentials (username and password), as well as certain specific consumption data. In addition, it needs to establish a secure communication channel with the relevant financial institution, exclusively for transmitting user orders such as subscription cancellations or renewals. This limited and controlled access is essential to ensure the proper functioning of the service without compromising customer autonomy or security. The fact that ReceiptVault handles this type of data makes it a particularly attractive target for cybercriminals.

The main cybersecurity challenges include unauthorized access through techniques such as *credential stuffing* or brute force (Doerfler et al., 2019), social engineering-based *phishing* attacks (Aleroud et al., 2017), and vulnerabilities arising from third-party technologies connected via API in *open banking* environments (Rasner, 2021). Although all these risks are relevant, this paper focuses exclusively on the *insider threat*, as it is particularly difficult to detect.

An insider threat is one of the most dangerous threats within the FinTech sector. It occurs when an employee, contractor, or someone with legitimate access to a system abuses that access to compromise users' personal and sensitive data or company assets. It differs from *phishing or account takeover* mainly because here the attacker in question has valid credentials. Later on, this paper evaluates and studies different detection models that have already been used by academics and researchers in order to compare them and establish which one ReceiptVault should implement in its defense strategy against insider threats.

2. Why is machine learning relevant to ReceiptVault's cybersecurity?

To understand the relevance and role of machine learning in the FinTech industry and in ReceiptVault's cybersecurity plan, we must first address what a machine learning model is based on and how an algorithm works, explained in simple terms. Machine learning is a branch of Artificial Intelligence (AI) that uses learning algorithms to identify patterns in data and improve its performance without the need for constant reprogramming. An algorithm, in simple terms, is nothing more than a set of instructions or rules that a system must follow to solve a problem. In fact, a simple recipe could be considered an algorithm, as it consists of a sequence of steps to follow that generate a result. In machine learning, these steps are applied in code format to achieve a specific result (in our case, a prediction). These algorithms try to identify patterns and adjust their behavior based on the data (information) they receive (Irekponor et al., 2024). Consequently, the operation of a machine learning algorithm is based on *learning* from training data. Instead of following predefined rules, the algorithm of a machine learning model analyzes a large volume of data and identifies trends, optimizing its predictions over time. This is the key process within the functioning of machine learning, where the detection of complex patterns or decision-making

based on real-time data is essential, especially in the FinTech sector in matters of cybersecurity (Ekundayo et al., 2024).

There are three machine learning models, which have been linked to the financial industry to contextualize their application in this work:

1. *Supervised:* The model learns from previously labeled data, already knowing the correct answer during its training process and applying that "learning" to new data. A typical case is digital fraud detection, where the algorithm classifies which payments were fraudulent and which were legitimate based on a previously loaded database on which the model is trained (Dahiya & Ranjan, 2021).
2. *Unsupervised:* The model looks for patterns and trends in data that has not been previously labeled. The algorithm must find differentiating factors between different types of data in order to arrive at a categorization and differentiation on its own. These models are useful in detecting financial anomalies or unusual behavior in financial networks (Guijarro Rodríguez et al., 2022).
3. *Reinforcement learning:* The algorithm makes decisions and receives penalties or rewards based on the outcome, previously established by the developer. Over time, it optimizes its behavior, offering better results (Ekundayo et al., 2024).

Within each type of model, depending on the type of learning, there are different popular algorithms. However, these have been explained in a separate sectionas threat detection models that may be viable for ReceiptVault.

2.1. Threat detection in the FinTech sector and application for ReceiptVault

Machine learning has become an essential tool for detecting cyber threats in the digital finance sector, largely thanks to its ability to analyze large volumes of data, as mentioned above. However, the main reason why it has become a crucial tool in the sector is the ability of models to adapt, without being subject to specific, immutable rules. It is important to note that different types of cyber threats are constantly evolving, and if the security operating system of a company such as ReceiptVault is based on static rules, it will be insufficient to stop increasingly sophisticated attacks (Stojanovic et al., 2021). In this scenario, machine learning makes it possible to identify unknown threats and generate new patterns for detecting them, in order to anticipate new attack tactics and changes in the behavior of cybercriminals (Ekundayo, 2024).

In the FinTech environment, and specifically in ReceiptVault, the nature of the data processed, such as bank accounts, purchase receipts, and consumer behavior patterns, makes it an attractive target for cybercriminals. This data can be used for information theft, identity theft, or

even to carry out extortion targeting customers, such as the threats previously described in section 2 of this paper.

To mitigate these risks, advanced machine learning models analyze user behavior, allowing for the detection of anomalies that could indicate fraudulent activity (Guijarro Rodríguez et al., 2022). In addition, machine learning enhances security on platforms such as ReceiptVault, optimizing the detection of unusual transactions, preventing fraud attempts, and improving protection against automated attacks targeting the application infrastructure (Dahiya & Ranjan, 2021).

In this context, different types of machine learning models with the potential to be implemented in ReceiptVault have been analyzed. Before analyzing the models found and their advantages and weaknesses, general classification criteria have been established, taking into account the overall performance of a model, the cost of implementation (both computational and monetary), and its complexity. This will allow us to follow a structured classification pattern to make an informed decision aligned with the sector's own cybersecurity standards and requirements to determine which model is optimal for implementation in ReceiptVault.

3. **Classification criteria for *machine learning* models for threat detection.**

In the case of cyber threat detection using machine learning models, it is important to note that all the models analyzed in this work are supervised classification models. Specifically, each model assigns a binary category to the processed instances: 1 (threat), 0 (no threat). This binary nature favors interoperability and comparison between the different models evaluated. Instances classified as 1 are considered positive, while those classified as 0 are interpreted as negative (Dueñas Quesada, 2020). Unlike regression, which is aimed at predicting continuous variables, classification is preferable in this context because the objective is not to estimate a magnitude, but to decide whether or not a behavior corresponds to a threat. Therefore, the nature of the problem justifies the classification approach applied (Dueñas Quesada, 2020).

To evaluate and classify machine learning models in the context of detecting cybersecurity threats or risks for ReceiptVault, this paper uses what is known as *a confusion matrix,* which compares four different classification possibilities for a *machine learning* model:

		Actual values	
		1	0
Predictions	1	TP (*True Positive)*	FP (*False Positive)*
	0	FN (*False Negative)*	TN (*True Negative)*

Table 1: Confusion Matrix

A true positive (TP) is obtained when the model correctly predicts that an instance represents a threat (1), and it actually is one in reality, just as true negatives (TN) are those that were correctly classified as non-threats (0). Conversely, a false positive (FP) occurs when the model classifies an instance as a threat (1), when in fact it is not (0). In machine learning, this is known as a Type I error (Zhou et al., 2021): the system detects a non-existent threat. For example, if a legitimate user accesses ReceiptVault from a new device, the model could misinterpret this behavior as an intrusion, blocking the account and informing the user of a possible credential compromise. This type of error negatively affects the user experience, and if it occurs frequently, it could generate mistrust of the platform and eventually lead to abandonment of the service.

However, Type II errors, associated with false negatives (FN), have potentially much more serious consequences for both the user and the company. This error occurs when the model fails to detect a threat that does exist, i.e., it predicts a 0 when it should have predicted a 1 (Zhou et al., 2021). For example, following a data breach, a cybercriminal accesses ReceiptVault using stolen credentials. Since these credentials are valid, the system does not classify the access as suspicious, allowing the attacker to view or extract sensitive information without being detected. This type of error directly compromises user privacy, damages the company's reputation, and can lead to legal or regulatory consequences. For this reason, in this paper, greater weight will be given to Type II errors in the comparative evaluation of classification models.

In this regard, the first and most important criterion is known as *Recall,* or Sensitivity. The *Recall* of a *machine learning* model is used to measure a model's ability to correctly detect positive instances, and is obtained using the following formula:

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

The value obtained will be between 0 and 1 (1≥ recall≥ 0). If the *recall* is equal to 1, the model detects all real threats, so Type II error is non-existent. If it is equal to 0, the model does not detect any real threats (all predictions are FN). That is why in order to reduce Type II error, *recall* must be maximized (Dueñas Quesada, 2020).

As a result, in this study, it has been decided to give greater weight to Type II error (false negatives) than to Type I error (false positives). This decision has been made because the cost of a Type II error for a company such as ReceiptVault is significantly higher than that of a Type I error. However, this prioritization of Type II error is not entirely without cost: it means accepting a higher number of false positives or "false alarms" that can result in interruptions or redundant verification actions for the user.

The study conducted by Mitrokotsa et al. (2008) justifies this weighting according to the following weighting matrix associated with the aforementioned confusion matrix:

	Predicted: Attack (1)	**Predicted: No attack (0)**
Actual: Attack (1)	TP	False Negative (C_{FN})
Actual: No attack (0)	FP (C_{FP})	TN

Table 2: Confusion Matrix adapted from Mitrokotsa et al. (2008)

In this case, the model assigns different weights to both FNs and FNs:

- FP cost = C_{FP}
- Cost of FN = C_{FN}

After evaluating different weightings during the study, the authors establish that for cases of cyber intrusions and threats, the weighting that offers the best result is as follows:

C_{FP} = 1
C_{FN} = 10

In other words, the cost of a FN or Type II error is ten times greater than the cost generated for the company caused by a Type I error (a false alarm). This weighting allows for a significant reduction in the number of FN cases without excessively compromising the false positive rate, which is the objective of this work (Mitrokotsa et al., 2008).

Continuing with this reasoning, other key metrics to consider are *precision* and the *false positive rate* (FPR), both of which are related to the Type I error we want to weight, assuming that we will allow a greater number of false alarms in order to maintain *recall* at levels very close to 1.

Accuracy allows us to evaluate the reliability of the detections made, measuring what proportion of the predictions categorized as "threats" correspond to real threats (Zhou, 2021). This proportion is obtained using the following formula:

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

The *F1-score* will also be taken into account as a criterion. It is the only criterion used that relates the two measures mentioned above, as it represents the balance between *precision* and *recall.* In other words, between the model's ability to avoid false alarms and its ability to detect real threats. The formula is as follows:

$$F1 - score = 2\ x\ \frac{Precision\ x\ Recall}{Precision + Recall}$$

The value of *the F1-score* will always range between 0 and 1. If *F1-score = 1,* both *precision* and *recall* are perfect. If it is equal to 0, either one of the two is also 0, or both are. The use of this criterion allows us to study in an integrated way how the model behaves in the balance between Type I and Type II errors. Although it does not differentiate between the two, for this we will use *precision and recall* independently, being able to establish which one is affecting the decrease (or increase) in the value of *the F1-score.* In addition, it is useful for ensuring that an excessive number of false alarms are not incurred, even though it is not a priority over FNs, in order to guarantee usability and improve the user experience (Mitrokotsa et al., 2008).

To illustrate this with a simple example, in a case where a model detects 4 threats, 2 of which are real, in a dataset where there were 5 real threats, the *F1-score* would be as follows:

$$Recall = \frac{TP}{TP + FN} = \frac{2}{2 + 3\ (3\ real\ threats\ that\ were\ not\ detected)} = 0.4$$

$$Precisión = \frac{TP}{TP + FP} = \frac{2}{2 + 2} = 0.5$$

$$F1 - score = \ 2\ x\ \frac{Recall\ x\ Precision}{Recall + Precision} = \ 2\ x\ \frac{0.4\ x\ 0.5}{0.4 + 0.5} = 0.44$$

A value of 0.44 indicates that the model has mediocre performance in balancing *precision* and *recall.*

Finally, another measure that will be taken into account is *accuracy,* which measures the percentage of observations that have been correctly identified by the model, or in other words, the total percentage of correct predictions made by the model:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

However, this metric is considered secondary to the rest. It will only be relevant when the other criteria mentioned above are favorable. The reason, as Sarhan and Altwaijry (2023) points out, is that most of the datasets analyzed in the area of threat detection are unbalanced, as they have a disproportionate number of observations that do not constitute a threat compared to those that do. Consequently, if the *accuracy* criterion is applied alone, the result is not reliable.

A simple example applicable to ReceiptVault: an imaginary dataset has 1,000 observations, of which 990 are not real threats and 10 are (unbalanced dataset). If a model that *only predicts that an observation will not be a threat* is used for this specific dataset, i.e., a model that always returns a 0, the criteria mentioned above would have the following results:

$$Recall = \frac{TP}{TP + FN} = \frac{0}{0 + 10} = 0$$

$$Precision = \frac{TP}{TP + FP} = \frac{0}{0 + 0} = Undefined\ or\ 0$$

$$F1 - score = 2\ x\ \frac{Recall\ x\ Precision}{Recall + Precision} = 2\ x\ \frac{0\ x\ 0}{0 + 0} = 0$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{0 + 990}{0 + 990 + 0 + 10} = \frac{990}{1000} = 99\%\ accuracy$$

Due to the nature of the unbalanced dataset, a model that only predicts zeros achieves 99% *accuracy*, but with unacceptable results for the rest of the criteria.

Once these criteria have been established, to complement the technical analysis of the models, a custom cost equation has been designed that quantifies the relative impact of Type I and II errors in the context of ReceiptVault. Given that Type II errors (failing to detect a threat) can pose serious risks to user security and trust, they have been assigned a weight ten times greater than that of Type I errors, following criteria similar to those proposed by Bin Sarhan and Altwaijry (2023). Thus, the formula allows metrics such as *recall* and *precision* to be translated into an

estimated cost that facilitates comparison between models beyond purely technical performance. The cost equation used is as follows:

$$Cost\ per\ prediction = (1 - Recall)\ x\ P_1\ x\ C_{FN} + (1 - Precision)\ x\ P_0\ x\ C_{FP}$$

- *(1 – Recall):* Actual threats that the model *fails* to detect. That is, Type II errors, or false negatives.
- P_1: Proportion of positive instances in the dataset (the real threats).
- C_{FN}: Estimated cost coefficient (approximate) for a false negative (Type II error).
- *(1 – Precision):* Number of false alarms detected by the model, Type I error.
- P_0: Proportion of negative instances in the dataset (non-threats).
- C_{FP}: Estimated cost coefficient (approximate) per false alarm (Type I error).

This equation will return an estimate of the unit cost per prediction of a model's errors, ranging from 0 to 1. A value of 0 means that the cost per prediction is zero, and the model always predicts perfectly, returning 0 false negatives and giving no false alarms. Conversely, the closer the value of the equation is to 1, the worse the model's performance. That is why it has been used to compare the models that have been studied in this thesis[1] .

4. Threat detection models.

4.1. Neural Network and Random Forest for internal threat detection: Models by Bin Sarhan & Altwaijry (2023).

As mentioned in section 2 of this paper, one of the threats that will be addressed most closely in this paper is *insider threats*, understood as malicious actions carried out by users with legitimate access to the system. In this section, we will analyze two of the models proposed by Bin Sarhan & Altwaijry (2023): the neural network and the *random forest*.

To carry out their analysis, the authors use the dataset known as *CERT (Community Emergency Response Team),* version r4.2, created by experts at *Carnegie Mellon University (CMU)*. The descriptive analysis of *CERT r4.2* is as follows:

- **Total number of employees:** 1,000
- **Number of attackers:** 30
- **Total number of events:** 32,770,227
- **Number of malicious events** (intentionally injected by experts): 7,323

[1] The values assigned to the cost coefficients, as well as the result of the equation, are indicative, based on the study by Sarhan and Altwaijry (2023). The coefficients can be changed and manipulated at the discretion of those who decide to use this formula.

The malicious events carried out by the 70 intruders injected on purpose consist mainly of the following threats:

1. **Data theft**: Users who start logging into other devices, connecting USBs or other download materials to leak information.
2. **Exfiltration before changing jobs**: Users who copy confidential information before leaving the company.
3. **Sabotage**: Employees who, for example, connect a device to other computers and access company executive accounts to send alarming emails or leak privileged information.

It should be noted that this database is completely synthetic, with fictitious instances based on a previous exhaustive study of the sector conducted by CMU.

As described in the dataset documentation, the number of observations corresponding to malicious users is considerably lower than that of normal users. Of the 1,000 simulated employees, only 30 exhibit malicious behavior, representing 3% of the total. According to the authors of the study, this disproportion accurately reflects reality, in which cyber attackers or internal users with malicious intentions constitute a statistical minority, compared to the vast majority of legitimate users. This constitutes what is described above as an *unbalanced dataset.* The dataset in question can be found at the following link: https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247

4.1.1. Data pre-processing.

The authors use a method known as *Deep Feature Synthesis* (DFS) to automatically generate variables from the data obtained, saving time by eliminating the need to generate variables one by one. After using DFS, the database is composed of variables such as "number of URLs visited," "number of USB connections outside working hours," "weekend logins," etc. The total number of variables generated with DFS is 69,738 variables for each of the 1,000 observations. Due to the high number of variables, which makes the *dataset* more difficult to handle, a second method of dimensionality reduction (variables) is applied: PCA (*Principal Component Analysis).*

PCA reduces the number of variables in the database into principal components with the aim of not losing too much information in the reduction (maintaining the variance as much as possible). The principal components are linear combinations of the original variables constructed in such a way that:

- The first principal component is the direction in the data space that captures as much information (variance) as possible.

- The second will be orthogonal (perpendicular) to the first, being the second component that collects the most variance.

As many principal components as desired can be extracted. The authors manage to reduce almost 70,000 variables to 553 principal components that capture 95% of the variance (information) in the database, which will be used as inputs for training the models.

Finally, the study applies the SMOTE (*Synthetic Minority Over-sampling Technique)* method. This technique consists of balancing unbalanced data sets. In this study, this occurs because only a small portion of the users in the dataset represent a threat, while the majority class consists of normal users. This imbalance can cause problems when training models, as in the vast majority of cases they tend to favor the majority class[2] .

To solve this problem, SMOTE generates new synthetic (though not identical) observations of the minority class – in this case, malicious users – to balance the dataset. The use of SMOTE allows models to have a more balanced view during r training, improving their ability to detect rare but important cases. However, if too many synthetic observations of the minority class are generated, there is a risk that the model will memorize the synthetic cases, including noise, and lose the ability to generalize correctly. This problem is known in machine learning as *overfitting,* in which the bias is very low and the variance very high, causing the model to obtain excellent results during training but poor results in testing.

4.1.2. Neural Network.

One of the models that Bin Sarhan and Altwaijry use to improve the detection of internal threats is a neural network. A neural network is a machine learning model inspired by the functioning of the human brain, composed of a series of nodes (neurons) that progressively transform input data into an output, in this case, binary (IBM, 2021). The input data or values enter the first layer of the neural network with a nonlinear activation function applied to them. The results are transmitted to the neurons in the next layer, which are associated according to predetermined weights, until they reach the final output. These models are widely used for complex nonlinear problems, as they model nonlinear relationships between variables, even when these relationships are not so obvious.

[2] See the explanatory example in the section describing the criteria for *accuracy,* where an algorithm that always predicts 0 or "normal" in a highly unbalanced data set would have a very high *accuracy*, which can lead to misinterpretations of a model's performance.

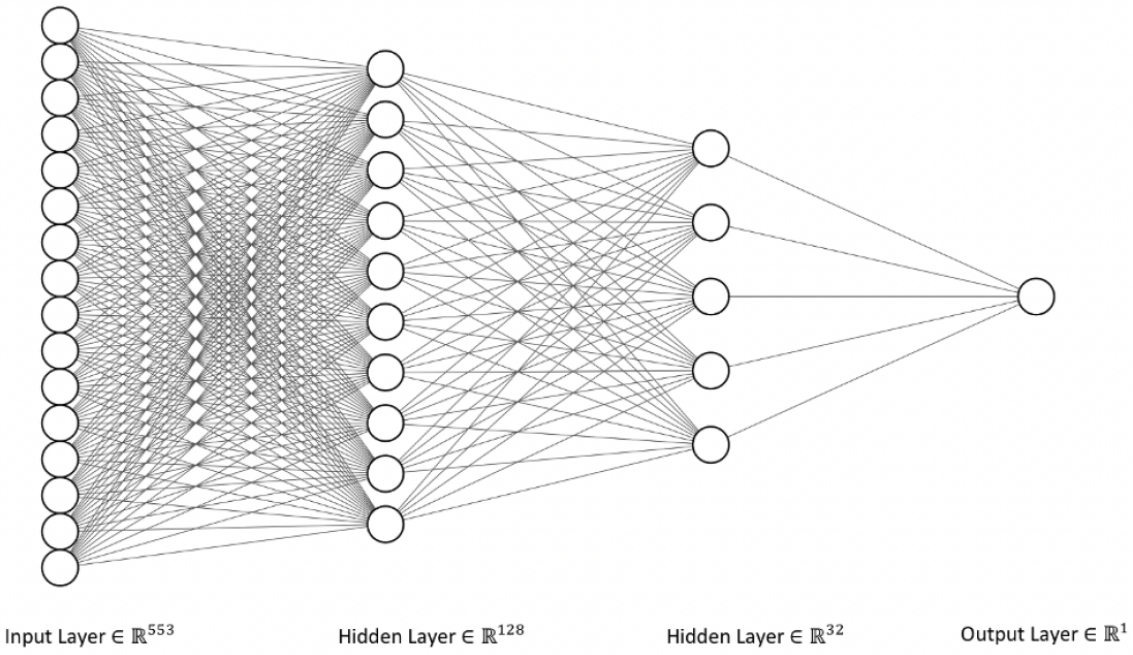

Figure 1: *Neural network architecture (Bin Sarhan & Altwaijry, 2023)*

As can be seen in Figure 1, the authors use a neural network with an initial layer, where the 553 variables obtained after performing PCA are introduced, two fully connected hidden layers, and a single output layer with a single neuron, which will give the binary result: threat or normal. The authors do not detail the activation functions used, but it is reasonable to assume that the ReLU function was used for the *input* layer *and* the hidden layers*,* and a sigmoid function for the *output* (Srivastava, 2024).

- *ReLU activation function:* Widely used for its simplicity and effectiveness in classification tasks. It transforms the input values in a neural network as follows: if the input value is positive, it remains the same. If it is negative or 0, it becomes 0. The mathematical function is expressed as follows:

$$f(x) = \max(0, x)$$

ReLU acts as a filter that only allows positive values to pass and suppresses negative ones. If, for example, one of the 553 variables has the following values when it reaches a layer: [1; -0.8; 3.2; 0; -1.2], ReLU will pass on [1; 0; 3.2; 0; 0] to the next neuron. This is useful for several reasons: it reduces the complexity of calculations, eliminates noise, and increases efficiency.

- *Sigmoid function:* This function is used in the last layer of a single neuron and transforms any real value into a number between 0 and 1 using the following formula:

$$f(x) = \frac{1}{1 + e^{-x}}$$

A *threshold* is applied*,* usually at 0.5, below which if the value is greater than 0.5, the sigmoid will return a 1 (threat), otherwise it will return a 0 (normal). If you want to prioritize reducing false negatives at the expense of a small increase in false alarms, that *threshold* can be reduced (ScienceDirect, n.d.).

To reduce the risk of the aforementioned overfitting, the authors apply two main techniques before using the model:

1. *Dropout:* This consists of randomly deactivating some of the neurons in each iteration, which forces the network not to rely excessively on any specific connection, favoring generalization.
2. *Hyperparameter optimization:* Using an optimization search process known as *GridSearch,* different combinations that directly influence the model's performance are tested. Among the hyperparameters adjusted are the *dropout rate,* number of epochs (iterations), *batch* size, and type of activation functions.

The dataset was divided into 80% for training and 20% for testing, applying stratified cross-validation (*Stratified K-Fold)* with 10 partitions – 9 for training and 1 for validation – ensuring that each partition maintained the same proportion of classes as the original set.

The results obtained by the authors using the neural network were as follows:

With SMOTE

Precision	Recall	F1-score	Accuracy
0.98	0.95	0.96	0.95

Without SMOTE or data balancing techniques

Precision	Recall	F1-score	Accuracy
0.94	0.97	0.96	0.97

Table 3: Neural network results (Bin Sarhan & Altwaijry, 2023).

It can be seen that without using the SMOTE technique in the case of this network, the criterion designated as key in this work*, recall*, improves. This criterion improves at the expense

of precision and *accuracy*, which the authors justify as a desirable result, since the number of false negatives is being reduced at the cost of slightly increasing the number of false alarms (precision is reduced).

This occurs because, by not applying SMOTE, no synthetic instances of the minority class have been generated, and the dataset is unbalanced (30:1000). With many more examples of normal users, the algorithm becomes sensitive to behaviors that deviate minimally from what it considers normal, detecting more true positives (TP) than with a balanced dataset. However, precisely because of this sensitivity, the number of cases in which the algorithm detects a threat where there are none (Type I error) increases. As clarified above, for a company like ReceiptVault, it is more valuable to minimize the number of attackers that have gone undetected by the model, even if the number of false positives increases. That is why the result of the algorithm has been chosen without prior balancing of the dataset to be entered into the cost-per-prediction equation as follows:

- C_{FN} = 10
- C_{FP} = 1

$$Cost\ per\ prediction = (1 - Recall)\ x\ P_1\ x\ C_{FN} + (1 - Precision)\ x\ P_0\ x\ C_{FP}$$
$$= (1 - 0.97)\ x\ 0.03\ x\ 10 + (1 - 0.94)\ x\ 0.97\ x\ 1 = 0.0672$$

The result of the equation establishes that for each prediction made by the model, i.e., for each user analyzed, an expected cost of 0.0672 monetary units (of any choice), for example, euros, is incurred as a result of the errors made.

- Type II error: When it fails to detect a real attacker: the equation penalizes the model by 10.
- Type I error: When it raises a false alarm for a normal user: the equation penalizes the model by 1.

In addition to the result of the equation, an *F1-score* of 0.96 indicates that the model has high and balanced performance, achieving both high precision and *recall*. The *F1-score* value remains high with and without SMOTE, suggesting that the model is capable of maintaining a similar balance in both cases, even though the criteria vary individually [3]. *Accuracy* also proves to be higher without applying SMOTE, which is a logical consequence of the unbalanced nature of a random forest dataset. That said, ReceiptVault has decided to prioritize threat detection (maximizing *recall)*, so it is worth looking beyond the *F1-score.* The model built without the use of SMOTE presents more desirable results for ReceiptVault's objectives.

[3] See the explanation in the criteria description section on the *accuracy* criterion.

4.1.3. Random Forest

The second model analyzed in this paper comes from the same study as the neural network by Bin Sarhan & Altwaijry (2023), as it also seeks to minimize insider threats, offering another alternative to the neural network.

A *random forest* model consists of a supervised learning model, like the neural network, based on the combination of multiple decision trees. A decision tree consists of a hierarchical structure algorithm similar to the shape of a tree, comprising:

- Internal node: Represents a condition or test on a variable (for example: "Is the number of files accessed greater than 1000?").
- Branch: Represents the binary result of that condition (yes or no).
- Leaves: Contain a final prediction (e.g., "malicious user").

A standard structure for a *random forest* model could be as follows:

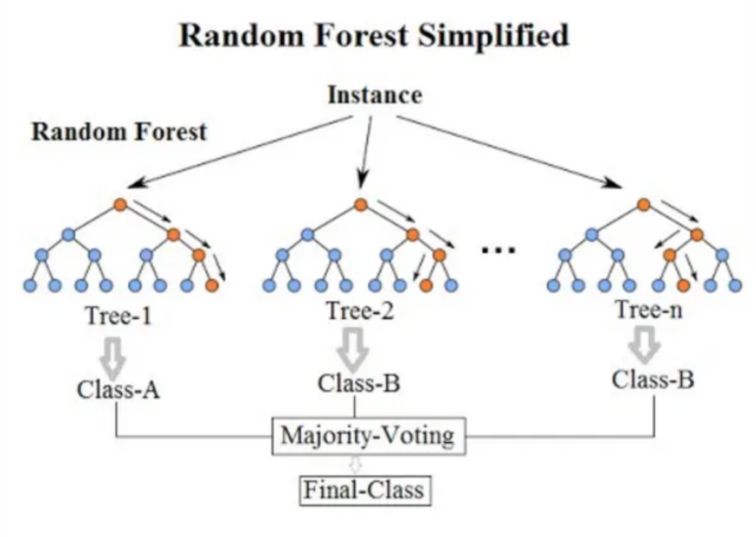

Figure 2: *Explanation of the Random Forest structure (Koehrsen, 2017)*

As can be seen in Figure 2, a Random Forest model consists of a collection of nnn independent decision trees, where each tree issues a class prediction for a given observation: normal (0) or attacker (1). The final class assigned by the Random Forest will be the one that receives the majority of votes among all the trees, using a majority voting mechanism.

As part of the same study as the neural network, the data pre-processing has been the same, applying DFS, PCA, and SMOTE techniques to reduce the dimensionality of the dataset and balance the classes. The division of *the dataset* for training and testing is the same, 80% - 20%, also applying *Stratified K-Fold* with 10 partitions to maintain the actual ratio between classes. When optimizing hyperparameters, the authors again use *GridSearch* to find the optimal points at which they achieve the best results. However, the hyperparameters used for the *random forest* algorithm are different from those used for the neural network:

- *n_estimators:* Number of trees in the forest. Increasing this parameter can improve model performance at the cost of increasing computational cost. The authors tested different configurations: {10, 20, 40, 50, 100, 150, 200, 500, 600}.
- *max_features:* This is a key hyperparameter in *random forest*. It marks the maximum number of variables considered in each subdivision. *Random forest* does not use all the variables in *the dataset* to reach a decision, but randomly chooses a subset of the variables from the original dataset, which is different each time. This ensures that each decision tree "sees" a different part of *the dataset*, reducing the correlation between trees and introducing diversity among them. The parameter controls how many random variables should be selected at each node. The authors evaluated three values:
    - *auto:* all available variables are considered.
    - *sqrt:* the square root of the total number of variables is used.
    - *log2:* the base-2 logarithm of the total number of variables is taken.
- *criterion:* This hyperparameter represents the *random forest's* division criterion. The criterion used by the authors is entropy, which measures how mixed the classes ar ly in a node. The more homogeneous the group, the lower its entropy. During training, each tree attempts to select divisions that reduce entropy as much as possible.

Once these hyperparameters are configured, the results shown by the algorithm are as follows:

With SMOTE

Precision	Recall	F1-score	Accuracy
0.95	0.72	0.81	0.72

Without SMOTE or data balancing techniques

Precision	Recall	F1-score	Accuracy
0.94	0.97	0.96	0.97

Table 4: Random forest results (Bin Sarhan & Altwaijry, 2023).

The results presented by the *random forest* are interesting for several reasons. First, we see that the algorithm's performance is considerably worse when data balancing techniques have been

used, especially *recall*, a criterion that has been determined to be crucial in this work. Furthermore, the results obtained without balancing techniques are identical to those obtained by the neural network, although the neural network performs much better when SMOTE is applied, which may mean that it is a more robust algorithm and less sensitive to changes in the nature of the data. Therefore, the analysis of criteria for the performance of *the random forest* without SMOTE is the same as for the neural network. Furthermore, in both algorithms, the unit cost function for the *random forest* will obtain the same cost per prediction as the neural network: 0.0672 monetary units per prediction.

4.1.4. Analysis of the use of SMOTE in the classification models of the study.

This section has been added to clarify why the classification models used by Bin Sarhan & Altwaijry (2023) in their study perform worse when balancing the classes in the dataset using SMOTE. These conclusions have been reached based on our own research and that of the authors themselves.

The original dataset is undeniably unbalanced, with 3% of observations belonging to the positive class (attacker) and 97% to the negative class (normal). However, the first thing to clarify is that, although this class imbalance is notable, it is common in the field of threat detection and does not prevent the analysis or construction of effective models. According to Google Developers (n.d.), a dataset has different levels of imbalance. When the minority class is between 1% and 20%, the imbalance is considered moderate, but not extreme (less than 1%). Therefore, a 3% minority class, as in this case, is within the manageable range and does not necessarily require balancing techniques such as SMOTE (Google Developers, n.d.).

In fact, we can see how in both models the use of SMOTE impairs the performance of the algorithms. In the neural network, the results continue to establish that the model is solid due to the complexity of the algorithm's architecture, which makes it robust and consistent even when the dataset is more balanced. However, it should be noted that the computational cost of a neural network is generally higher than that required by a *random forest*. The latter is significantly affected by the use of balancing techniques, considerably reducing *recall and* the *F1-score.* This may be because the synthetic observations of the minority class implemented by the use of SMOTE "confuse" the algorithm in the learning phase, introducing noise and overgeneralizing the minority class, thereby impairing the algorithm's performance. In the case of *random forest,* the lower complexity of the algorithm compared to a neural network could explain its lower ability to handle such cases, since the neural network performs well even with SMOTE (Kim, 2023). Despite this, this study shows that for this type of threat (insider threats*)* and using the CERT r4.2 dataset, both algorithms perform better without prior rebalancing.

Finally, it is important to note that there is a possibility that a future dataset may not be unbalanced, and in this regard, as concluded after studying the experiment by Bin Sarhan & Altwaijry (2023), the neural network offers more versatility and consistency in results than the *random forest.*

4.2.Model proposed by Akazue, Muka & Edje (2024). Hybrid SVM and K-NN model.

The study in question also addresses the problem posed by internal threats (users with legitimate access) on digital platforms, as does the previous study analyzed in this paper. The authors' approach in this case is based on the need for increasingly sophisticated systems to deal with more elaborate threats, so they propose a hybrid *machine learning* model that combines two types of classic algorithms: *Support Vector Machine (*SVM) and *K-Nearest Neighbors (*K-NN). Both algorithms are supervised learning and classification algorithms, so the unit cost per prediction equation will be equally applicable (Akazue, Muka, & Edje, 2024).

4.2.1. Data pre-processing.

The dataset used is the same as in the previous study analyzed: *CMU's CERT r4.2*, as it is the most popular *dataset* when dealing with internal threat detection problems. Unlike the study by Bin Sarhan & Altwaijry (2023), the authors decide to use SMOTE from the outset for both models. This decision is based on the need to ensure the robustness of the system even in less unbalanced scenarios. Although the original dataset is designed with an intentional imbalance that reflects real-world situations (high prevalence of legitimate activity versus few threats), there may be specific cases where the classes are more evenly matched. In these situations, the algorithm must still be able to correctly distinguish between malicious and normal behavior. The early use of SMOTE allows these contexts to be simulated by generating synthetic observations for the minority class, promoting more generalizable learning and optimal model performance (Akazue, Muka, & Edje, 2024).

4.2.2. System Architecture

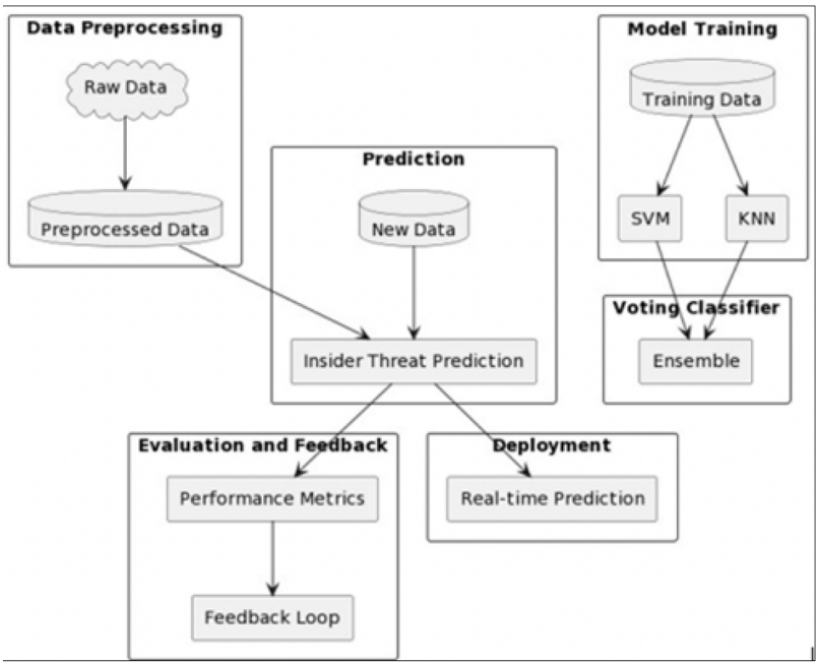

Figure 3: Proposed System Architecture (Akazue, Muka, & Edje, 2024).

- *Data Processing:* This action transforms "raw" data into processed data so that it can be used during model training.
- *Model Training:* The processed data is applied to train each model (K-NN and SVM) separately. Each model independently learns the patterns associated with internal threats from the data entered.
- *Voting Classifier:* An *ensemble* is used through *hard voting* that unifies the predictions of both models, returning a single prediction that is the result of the consensus of the K-NN and SVM. *Hard voting* is a majority voting mechanism that always chooses the most popular class among the individual predictions of the *ensemble* algorithms. In this case, there are two algorithms, so there is a possibility of a tie (Salunke, 2024). The study does not specify what the tiebreaker mechanism is in these cases. However, if we follow the reasoning outlined above regarding the greater severity of Type II errors (false negatives), it would be reasonable that, in the event of a tie (for example, if KNN predicts "threat" and SVM predicts "no threat"), the system would choose to classify the instance as a "threat," even at the cost of increasing Type I errors ([4]).
- *Prediction:* The observations from the *test set* are entered as *input* into *the ensemble*, and a prediction is obtained.
- *Evaluation and Feedback:* In this part, the models are evaluated according to the criteria described above: *recall,* precision, *F1-score*, and *accuracy*.

[4] As it is not specified in the study, this note on the *voting classifier* is merely an assumption, based on what would be done when prioritizing the minimization of Type II error in ReceiptVault.

- *Deployment:* The authors test the *ensemble* with real-time data.

4.2.3. Support Vector Machine (SVM) and K-Nearest Neighbors (KNN).

The SVM algorithm is a supervised learning model that works for both regression and classification, although it is more commonly used for binary classification, as in this case. Its main objective is to find a decision boundary, called a hyperplane, that clearly separates each class in a dataset. SVM searches for this hyperplane by trying to maximize the distance of the observations closest to the boundary (maximizing the margin) until it finds an optimal boundary where this margin is the maximum possible.

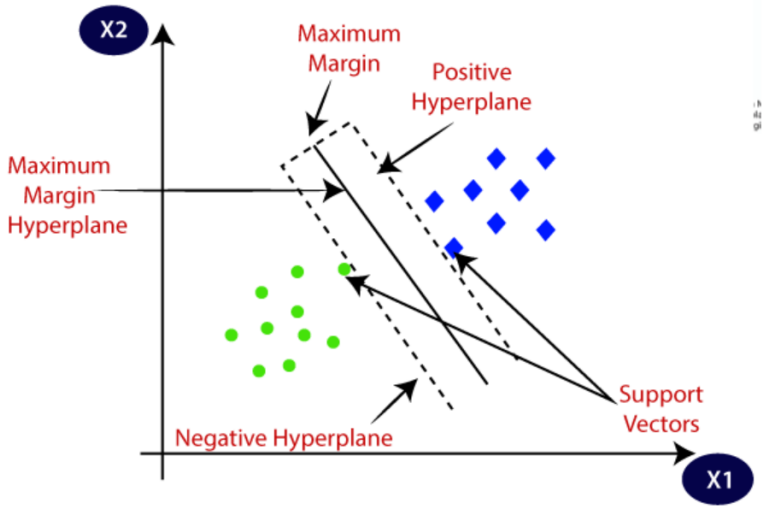

Figure 4: *SVM (Medium, 2024)*

The observations closest to the hyperplane are the *support vectors*, which the algorithm takes as a reference to maximize the margin between them. When the data cannot be separated linearly, SVM uses transformation functions called *kernels* that allow the data to be projected into a higher-dimensional space where it is possible to find a separating hyperplane (Fagbuiro, 2023). The algorithm will classify each observation according to which side of the hyperplane it is positioned on. The performance of an SVM model is regulated through the calibration of hyperparameters. The most common hyperparameters for these models are precisely the aforementioned *kernel,* which will specify the type of transformation to be applied to the data if it is not linearly separable, and C, which controls the balance between having a wide margin and allowing errors in the classification. The higher the value of C, the more errors are penalized and the narrower the margins become (Fagbuiro, 2023).

The *K-Nearest Neighbors* algorithm (supervised learning) is a type of model that is also popularly used for classification tasks. Its operation is based on the premise that similar

observations usually belong to the same class. KNN places the observations in a data space, and for each new observation it calculates the distance from the closest "neighbors" to it (usually the Euclidean distance is calculated, although there are different types). The number of neighbors or observations taken into account is determined by the hyperparameter k. If k = *3,* the 3 closest observations will be taken. The lower k is, the more sensitive the model is to noise and outliers, and it may overfit. That is why, normally, to find the optimal value of k, the cross-validation technique used in the previous study analyzed (IBM, n.d.) is used.

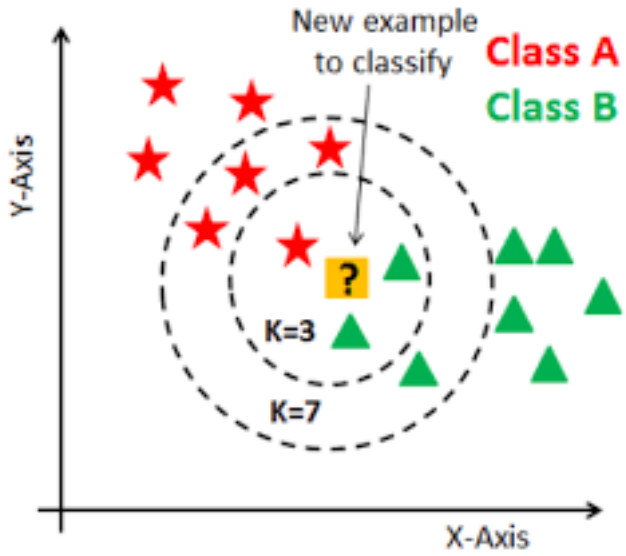

Figure 5: *K-NN (Gómez Servan, 2022).*

One limitation of the study by Akazue, Muka, & Edje (2024) is that it does not specify the value of the hyperparameters for either of the two algorithms they use, which represents a significant lack of methodological transparency. The omission of this key step undermines ReceiptVault's ability to replicate their model, limiting its applicability. Nevertheless, the study was chosen because of the clarity and precision of the explanation of the *ensemble* results and the model architecture, something that was not as clear in the study by Bin Sarhan & Altwaijry (2023). Another reason for choosing this study is that it uses the same model performance evaluation criteria that have been decided upon in this work. Furthermore, by understanding how both algorithms and their hyperparameters work, ReceiptVault engineers can experiment with different values for these, evaluating the performance of *the ensemble* for each configuration[5] .

4.2.4. Model results.

[5] Such experimentation with hyperparameters could be part of future developments by the ReceiptVault technical team.

	Accuracy	Recall	F1-score	Accuracy
SVM	84%	67%	72%	67%
K-NN	93%	89%	90%	89%
Hybrid Model	99%	98%	97%	99%

Table 5: Results of the models by Akazue, Muka, & Edje (2024).

In this case, it can be seen that the performance of the separate models is poor compared to the previous study that analyzed the neural network and *random forest*. In particular, the performance of the SVM is mediocre, with a *recall* of 67%. This disappointing result could be due to inadequate calibration of the hyperparameters. The model seems to penalize errors in the hyperplane delimitation little, possibly due to an inappropriate value of the hyperparameter C, although other factors not specified by the authors may also influence this. Although the K-NN results are better, their analysis could be more rigorous if the value of k and the type of distance used to calculate the proximity between neighbors were specified.

However, the results of the hybrid (*ensemble)* model demonstrate the best performance of any model analyzed in this study. Although the performance of SVM and K-NN separately is not ideal, the combination in the *ensemble* shows very promising results, greatly improving the predictive power of both algorithms.

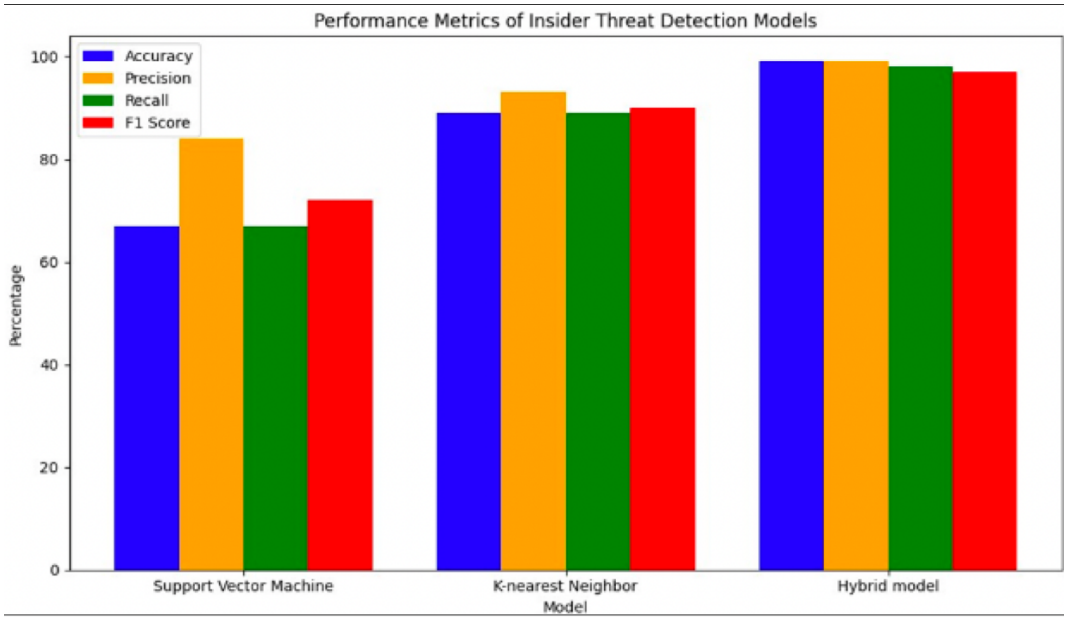

Figure 6: Graphical comparison of models (Akazue, Muka, & Edje, 2024).

Before comparing the cost functions, it is important to remember that the termsP_1 andP_0 will change, as these models have been analyzed after balancing the classes with SMOTE.

Therefore, there will no longer be an imbalance of 3% and 97%. As can be assumed from Brownlee's (2021) article, in order to avoid overfitting, a class rebalancing in the common dataset is 10% of the minority class. Since the authors do not specify what proportion of classes exists after using SMOTE, this paper will assume a 10-90 proportion to use in the unit cost equation.

The unit cost function applied to the three models gives the following results[6] :

- C_{FN} = 10
- C_{FP} = 1
- $P_1 = 0.10$
- $P_0 = 0.90$

$$Cost\ per\ prediction_{SVM} = (1 - 0.67)\ x\ 0.10\ x\ 10 + (1 - 0.84)\ x\ 0.90\ x\ 1 = 0.474$$

$$Cost\ per\ prediction_{K-NN} = (1 - 0.89)\ x\ 0.10\ x\ 10 + (1 - 0.93)\ x\ 0.90\ x\ 1 = 0.173$$

$$Cost\ per\ prediction_{\ Híbrido} = (1 - 0.98)\ x\ 0.10\ x\ 10 + (1 - 0.99)\ x\ 0.90\ x\ 1 = 0.029$$

The cost per prediction of the hybrid model proves to be the lowest, and therefore the most favorable. Assuming that the$C_{FN} = 10€$ and the$C_{FP} = 1€$, the unit cost per prediction of the hybrid model would be €0.029 per prediction. In addition, the *F1-score and accuracy,* which are not included in the equation, are also very high, demonstrating an excellent balance between precision and *recall and* a high capacity for correct prediction by the *ensemble*.

5. Conclusions, discussion, and future applications.

Throughout this paper, different machine learning models for detecting internal threats have been evaluated in order to determine, in accordance with existing regulations, which would be the most appropriate to apply to ReceiptVault's cybersecurity strategy. After evaluating the results of the models proposed by the study by Bin Sarhan & Altwaijry (2023) and Akazue et al. (2024), and applying a custom cost function that prioritizes Type II error (false negatives) over Type I error (false positives), a series of key conclusions have been drawn.

First, although the model presented by Akazue et al. (2024) performs best among all the models analyzed and has a cost function that yields a lower result than the others, its lack of methodological rigor raises doubts about its actual applicability in ReceiptVault. The study does

[6] The coefficients assigned to Type I and Type II errors will be the same as in the first study analyzed in order to compare the cost functions.

not specify key hyperparameter values for any of the models described—such as k in K-NN or C in SVM—which makes it difficult to replicate the hybrid model or evaluate the robustness of the algorithms before changes in configuration. If the dataset used in the future is different, it is crucial to know, for example, whether the data is linearly separable or not for the correct calibration of *the kernel* in SVM, or whether the value of k should be changed to a higher or lower value. That is why, as a possible future study, we propose to investigate in greater depth a model similar to that proposed by Akazue et al. (2024), applying rigorous experimentation with hyperparameters to evaluate the true potential of *the ensemble.*

For the context of ReceiptVault, the neural network model by Bin Sarhan & Altwaijry (2023) has proven to be the most suitable for the company. In addition to specifying how the algorithm has been tuned by describing the hyperparameters used in the model, it demonstrates desirable consistency for both balanced and unbalanced datasets, as its performance is high both with and without the application of SMOTE and class balancing techniques. *Random forest,* on the other hand, also belonging to the same study, demonstrates poorer performance in the balanced *dataset*, with a *recall* of 0.72, compared to the 0.95 obtained by the neural network with SMOTE. Although it is true that in the original unbalanced set both algorithms perform equally—and present the same result in the cost function—it is preferable to opt for the model that demonstrates greater versatility. In this case, it is the neural network.

Furthermore, as a future line of research, it would be relevant to study the computational and monetary costs associated with implementing each of the models analyzed. This assessment would allow decision-making to be adjusted beyond pure technical performance. If, for example, a particular model offers a slight improvement in *recall* but requires more complex infrastructure (such as renting or building a data center to house high-performance servers), it will be necessary to assess whether this improvement justifies the economic and technical sacrifice involved in . In the context of a *startup* like ReceiptVault, scalability and resource efficiency are key elements that should be part of the comparative analysis between models.

It is also important to remember that ReceiptVault does not seek to replace banks or other financial management platforms. It is a complementary, flexible, and lightweight tool designed to make life easier for the average user by organizing their receipts and subscriptions in one place, without the need to access each company's portal individually. It does not carry out financial transactions, and its objective is not to operate as a banking entity, but rather to integrate itself lightly into the current financial ecosystem. This also determines the type of threats it faces and the type of protection it needs: it is not a matter of stopping suspicious transfers, but rather of preventing unauthorized access to sensitive information. Therefore, the detection model implemented must be aligned with this logic.

Ultimately, this work underscores that methodological rigor is even more important than the quantitative results obtained by a model. A model that presents outstanding metrics but lacks transparency in its configuration (as is the case with the hybrid analyzed) seriously limits its practical usefulness. On the contrary, having a clear architecture with well-defined and adjustable hyperparameters allows not only to replicate the model, but also to refine it and adapt it to the changing needs of the platform. This flexibility is essential in an environment such as ReceiptVault, where the threat can evolve, and the ability to readjust the defensive strategy makes all the difference. Detecting internal threats is not just a technical issue: it is a strategic necessity. These attacks, because they come from within and with valid credentials, are one of the most dangerous forms of vulnerability for any company in the FinTech sector. Ensuring their detection is not an option, but a priority.

6. Bibliography.

Akazue, M. I., Muka, N. Q., & Edje, A. E. (2024). Mitigating insider's threats using support vector machine and k-nearest neighbor. *International Journal of Science and Research Archive, 12*(1), 2626–2635. https://doi.org/10.30574/ijsra.2024.12.1.1110

Aleroud, A., & Zhou, L. (2017). Phishing environments, techniques, and countermeasures: A survey. *Computers & Security*, *68*, 160-196.

BBVA. (2023). Digital finance and its regulation in 2023. *BBVA*. https://www.bbva.com/es/economia-y-finanzas/las-finanzas-digitales-y-su-regulacion-en-2023/

Bin Sarhan, B., & Altwaijry, N. (2023). *Insider threat detection using machine learning approach*. *Applied Sciences, 13*(1), 259. https://doi.org/10.3390/app13010259

Brownlee, J. (2021). *Random oversampling and undersampling for imbalanced classification*. Machine Learning Mastery. https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/

Dahiya, S., & Ranjan, P. (2021). *Optimizing API Security in FinTech Through Genetic Algorithm based machine learning Model*. International Journal of Communication Networks and Information Security, 13(3). https://www.researchgate.net/publication/385782853

Doerfler, P., Thomas, K., Marincenko, M., Ranieri, J., Jiang, Y., Moscicki, A., & McCoy, D. (2019). Evaluating login challenges as a defense against account takeover. In *The World Wide Web Conference* (pp. 372-382).

Dueñas Quesada, J. M. (2020). *Supervised learning for web threat detection using decision tree-based classification: Application of machine- -learning techniques to cybersecurity*. Open University of Catalonia, Autonomous University of Barcelona, and Rovira i Virgili University.

Ekundayo, F., Atoyebi, I., Soyele, A., & Ogunwobi, E. (2024). *Predictive Analytics for Cyber Threat Intelligence in Fintech Using Big Data and Machine Learning*. International Journal of Research Publication and Reviews, 5(11), 5934-5948. https://www.researchgate.net/publication/386144447

European Union (2016). *General Data Protection Regulation (GDPR)*. Regulation (EU) 2016/679.

Fagbuiro, D. (2023). Understanding the support vector machine (SVM) model. *Medium*. https://medium.com/@davidfagb/understanding-the-support-vector-machine-svm-model-c8eb9bd54a97

Government of Spain. (2022). *Royal Decree 311/2022, of May 3, regulating the National Security Scheme*. Official State Gazette. https://www.boe.es/buscar/act.php?id=BOE-A-2022-7191

Gómez Servan, W. J. (2022). K-Nearest Neighbors (KNN): Unsupervised learning algorithm. *RPubs*. https://rpubs.com/WaldoGomez10/algoritmo_knn_modulo3

Google Developers. (n.d.). Handling imbalanced datasets. *Google Developers.* https://developers.google.com/machine-learning/crash-course/overfitting/imbalanced-datasets

Guijarro Rodríguez, A. A., Jácome-Morales, G. C., Gonzalez-Mestanza, V., Terán-Zurita, E., & Torres-Martínez, D. E. (2022). *Detection of security threats in a corporate network using machine learning algorithms*. Scientific Series of the University of Information Sciences, 15(12), 183-193. http://publicaciones.uci.cu

IBM. (n.d.). What is the K-nearest neighbors (KNN) algorithm? *IBM Think.* https://www.ibm.com/es-es/think/topics/knn

IBM. (n.d.). What is a neural network? *IBM*. https://www.ibm.com/think/topics/neural-networks

Irekponor, O., Aleke, N. T., Yeboah, L., & Joseph, J. E. (2024). *Machine learning techniques for enhancing security in financial technology systems*. International Journal of Science and Research Archive, 13(01), 2805–2822. https://www.researchgate.net/publication/385275742

Kim, M. (2023). SMOTE: Practical Consideration & Limitations. *Medium*. https://medium.com/@minjukim023/smote-practical-consideration-limitations-f0d926b661a8

Koehrsen, W. (2017). Random Forest simple explanation. *Medium*. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d

Mitrokotsa, A., Dimitrakakis, C., & Douligeris, C. (2008). *Intrusion detection using cost-sensitive classification*. arXiv.

European Parliament and Council of the European Union. (2015). *Directive (EU) 2015/2366 of November 25, 2015, on payment services in the internal market*. Official Journal of the European Union, L 337, 35–127. https://eur-lex.europa.eu/eli/dir/2015/2366/oj

European Parliament and Council of the European Union. (2016). *Regulation (EU) 2016/679 of the European Parliament and of the Council of April 27, 2016, on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation).* Official Journal of the European Union, L 119, 1-88. https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=celex:32016R0679

Rasner, G. C. (2021). *Cybersecurity and third-party risk: Third party threat hunting*. John Wiley & Sons.

Salunke, M. (2024). Voting Classifier: Hard and Soft in Scikit Learn. *Medium*. https://medium.com/data-and-beyond/voting-classifier-hard-and-soft-in-scikit-learn-d2f3c091d973

ScienceDirect. (n.d.). Sigmoid function. *ScienceDirect.* https://www.sciencedirect.com/topics/computer-science/sigmoid-function

Scott, H. S. (2021). The EU's Digital Operational Resilience Act: Cloud Services & Financial Companies. *Available at SSRN 3904113*.

Srivastava, S. (2024). Understanding the difference between ReLU and Sigmoid activation functions in deep learning. *Medium*. https://medium.com/@srivastavashivansh8922/understanding-the-difference-between-relu-and-sigmoid-activation-functions-in-deep-learning-33b280fc2071

Stojanović, B., Božić, J., Hofer-Schmitz, K., Nahrgang, K., Weber, A., Badii, A., Sundaram, M., Jordan, E., & Runevic, J. (2021). *Follow the Trail: machine learning for Fraud Detection in Fintech Applications*. Sensors, 21(5), 1594. https://doi.org/10.3390/s21051594

Theogene, D. (2024). Overcoming payment and receipt management hurdles for SaaS companies. *GetPliant*. https://www.getpliant.com/es/blog/obstaculos-de-gestion-de-pagos-y-recibos-para-empresas-saas/

Comillas Pontifical University. (n.d.). *Comillas Emprende*. https://www.comillas.edu/comillas-emprende/

Zhou, J., Gandomi, A. H., Chen, F., & Holzinger, A. (2021). *Evaluating the quality of machine learning explanations: A survey on methods and metrics*. Electronics, 10(593). MDPI.