BACHELOR'S THESIS 2023

Analysis of different machine learning approaches for financial transactional fraud detection

Max Söderlind & Daniel Amir

ISSN 1650-2884
LU-CS-EX: 2023-23

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

LUNDS UNIVERSITET
Lunds Tekniska Högskola

DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE,
15 CREDITS LUND, SWEDEN 2022

Analysis of different machine learning approaches for financial transactional fraud detection

Max Söderlind & Daniel Amir
2022
Bachelor thesis in Computer Science and Engineering

Faculty of Engineering LTH Department of Computer Science and Engineering

Authors

Max Söderlind <ma1606so-s>
Computer Science and Engineering student
The Faculty of Engineering at Lund University

Daniel Amir <da8446am-s>
Computer Science and Engineering student
The Faculty of Engineering at Lund University

Swedish title

Analys av olika metoder för maskininlärning för upptäckt av finansiella transaktionsbedrägerier

Location

Lund, Sweden LTH

Examiner

Marcus Klang <marcus.klang@cs.lth.se>

Supervisor

Dennis Medved <dennis.medved@med.lu.se>

Abstract

Credit card fraud is a societal problem affecting millions of people every year resulting in losses of billions of dollars. Detecting fraudulent transactions is difficult for the human eye and through the rise of technology, there are now other, more complex and accurate solutions to this problem using machine learning and artificial intelligence. The aim of this thesis is to assess how different machine learning algorithms can be used to accurately predict fraudulent credit card transactions.

In this study, 3 different machine learning models were implemented. One random forest model, one neural network model, and an ensemble model, based on a random forest. Moreover, hyperparameter tuning algorithms were used to maximize the accuracy of detecting fraudulent transactions. The models were tested with a dataset that consisted of 284,807 transactions with 492 frauds.

The finding of this work was that the ensemble model outperformed the neural network model and the random forest model in detecting fraudulent transactions with an accuracy of 93.3%, but performed worst in detecting non-fraudulent transactions with an accuracy of 96.0%. The MCC scores indicate there is room for improvement on all three trialed models. The best MCC score on the test data was achieved by the random forest model with 0.399, while the worst was achieved by the ensemble model with 0.284. Furthermore, there were small differences in accuracy between the models and to further improve the accuracy of detecting fraudulent transactions, more complex machine learning models could be implemented. Overall all models performed better than baseline on all datasets and measurements, except non-fraud detection accuracy.

Preface

This thesis is produced as part of the graduating work for our Computer Science and Engineering degree at Lund University of Technology (LTH). The project spanned the duration from September 2022 to December 2022 under the supervision of Dennis Medved (supervisor) and Marcus Klang (examinator).

We have always been interested in computer science and have had a special interest in machine learning and deep learning for many years. During our university studies, we came across different machine-learning methods. We sought to get a better understanding of different models as well as how they work differently from each other. Furthermore, we wanted to implement these models and see how they could be applied to real-life business situations since both of us are interested in business and economics.

In this thesis, the machine learning models Neural Networks and Random Forest Tree were implemented and modeled and applied on a real-life fraud data set, with the intent to improve our understanding of the machine learning models and how well they can perform on a real-life business problem.

We want to thank Dennis Medved for taking the time to supervise us during this project and Marcus Klang for examining us and making sure that our project meets the required standards.

Table of contents

1. Introduction **10**

1.1 Problem statement 10

1.2 Division of labour 11

1.3 Scope 11

1.4 Outline 11

2. Theoretical background **12**

2.1 Artificial intelligence and machine learning 12

2.2 Machine learning terms 13

2.3 Machine learning metrics 14

2.4 Neural networks 14

2.5 Random forest 16

2.6 Machine learning ensemble and stacking 18

2.7 Standard scaling 18

2.8 Tools 19

3. Methodology **20**

3.1 Tools 20

3.2 Implementation 20

3.2.1 Data 20

3.2.2 Machine learning algorithms 21

3.2.3 Data preparation 21

3.2.4 Neural network model training and evaluation 21

3.2.5 Random forest model training and evaluation 23

3.2.6 Ensemble approach: Stacking base model predictions to a random forest 23

3.2.7 Model comparison 24

3.2.8 Baseline 24

4. Results **25**

4.1 Hyperparameters for random forest & ensemble 25

4.2 Accuracy results 25

4.3 ROC & AUC comparison 26

4.4 MCC comparison 27

4.5 Results distribution 27

5. Discussion **28**

5.1 Hyperparameters for random forest & ensemble 28

5.2 Accuracy results 28

5.3 ROC & AUC comparison 28

5.4 MCC comparison 29

5.5 Results distribution 29
5.5 Limitations 29

6. Conclusion **31**
6.1 Future work 31

5. References **32**

1. Introduction

1.1 Problem statement

A credit card is a payment card assigned to a customer (cardholder) and allows the customer to purchase goods and services where the bank issues the money and then invoices the customer for the purchases. A credit card, therefore, allows the customer to make purchases and pay for them later in time. Credit cards are an easy target for fraudsters since the cards allow them to, without any risk, make purchases and withdraw amounts without the cardholder's knowledge.

Credit card fraud is a societal problem that is affecting millions of people every year[1]. In 2020, almost 400,000 people in the U.S reported that they had been a victim of credit card fraud which was the second most reported identity theft category. Furthermore, a total of 28.58 billion dollars was lost to credit card fraud in 2020[2].

Since 2015, credit card fraud has shifted toward card-not-present crime, which is when the purchase is made with the details of the card without the card being present. This is mostly because of the chip being introduced on credit cards that disabled the criminal's ability to skim the cards, access all the data, and produce a clone. The availability of compromised card information, coupled with a rise in online shopping, has dramatically increased card-not-present credit card fraud. At present, there is no technological solution similar to a microchip that can significantly reduce card-not-present fraud. Instead, the burden is on both credit card issuers and merchants to employ stronger digital security measures to identify and block fraudulent transactions in real-time. In recent years, as the amount of data has exploded and the number of payment card transactions has skyrocketed, fraud detection has become largely digitized and automated.

Most modern solutions leverage **artificial intelligence (AI)** and **machine learning (ML)** to perform data analyses, predictive modeling, data-driven decision-making, fraud alerts, and remediation activities that occur when individual instances of credit card fraud are detected. For example, Paypal has managed through a neural network, and machine learning to keep its fraud rate at 0.32 percent of revenue, compared to the average that other merchants see which is 1.32 percent[3].

---

[1] *(*Federal Trade Commission, *Imposter scams top complaints made to FTC in 2018,* 2019*)*
[2] *(*Federal Trade Commission, *Consumer Sentinel Network, 2022)*
[3]*(*Stuart, S*, Paypal vs. fraud - Have no fear, machine learning is here!, 2018)*

1.2 Division of labour

Max Söderlind and Daniel Amir have done the study and report. The division of work has been distributed equally between the two students. Both have worked with implementing the code, research, and writing of the report. Both authors have worked closely to ensure that all tasks were completed and the project was successful. Regular meetings were held to discuss progress and any challenges that arose. By dividing the work in this way, the authors could utilize each author's strengths and expertise and complete the project in a timely and efficient manner.

1.3 Scope

The scope of this project includes conducting research on the effectiveness of predicting fraudulent credit card transactions using machine learning algorithms. This research has focused on three specific machine learning algorithms: Neural network, Random forest, and an Ensemble algorithm. The results of this research were used to determine the most effective approach for detecting fraudulent credit card transactions.

The project began with research of theoretical background to gather existing information on the topic. Next, the implementation of the machine learning algorithms was done. After that, the machine learning results were analyzed to identify trends and key findings. Finally, the project concludes with a written report summarizing the research and results and providing recommendations for further research.

This project is limited in scope to the research and analysis of the three specified machine learning algorithms. It does not include conducting additional research on other algorithms. The findings of this project will provide valuable insights but should not be considered exhaustive or definitive.

1.4 Outline

The report will firstly discuss the theoretical background and explain how the algorithms and models were implemented, secondly present the results from the executions of the models, and thirdly analyze and discuss the results together with the theoretical background.

2. Theoretical background

2.1 Artificial intelligence and machine learning

Artificial intelligence can be described as the science of making machines intelligent[4]. The idea is to mimic how the human brain solves problems and makes decisions by using computer science. Machine learning is a branch within artificial intelligence that focuses on using algorithms and data to copy the way humans learn and thus gradually improving its accuracy[5]. Machine learning models are built by training machine learning algorithms with data.

A machine learning model could be divided into three parts:

1. **Decision process:** The purpose of the model is to reach a decision, usually in the form of a classification or prediction. The algorithm uses data to find patterns that are used to produce a prediction.
2. **Error function:** Error functions are used to evaluate the accuracy of the model by calculating how good the model's predictions are.
3. **Model optimization:** machine learning algorithms use weights to produce better predictions and depending on how these weights are adjusted, the accuracy can shift. They also have hyperparameters that can be tuned that affect how the model operates and how weights are adjusted, which in term can affect how well the model performs. By optimizing these hyperparameters the model can improve its accuracy.

Machine learning algorithms fall into three different categories:

1. **Supervised machine learning:** It trains algorithms with labeled datasets to perform classifications or predictions. Some algorithms used in supervised learning include neural networks, linear regression, logistic regression, random forest, and support vector machine (SVM).
2. **Unsupervised machine learning:** It analyzes and clusters unlabeled data with the help of machine learning. The idea of unsupervised learning is to group data or find hidden patterns without human intervention. As these algorithms are good at discovering differences and similarities in data they are beneficial to use for data exploration, segmenting data, and customer

[4] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019)
[5] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019)

behavior as well as general pattern recognition. Some algorithms used in unsupervised learning include neural networks, k-means clustering, and probabilistic clustering methods.

3. **Semi-supervised machine learning:** As the name suggests semi-supervised learning uses an approach that combines supervised and unsupervised elements. When training the algorithm it can use a small labeled dataset to direct how classification and feature extraction should occur in a large unlabeled dataset. This approach can be beneficial when it is too expensive to label enough data or when not having enough labeled data for a supervised algorithm.

2.2 Machine learning terms

There are a few terms that are relevant to understand in order to get a complete understanding of our analysis:

- **True or false and positive or negative:** In machine learning terms such as true positive, false positive, true negative, and false negative are all used to identify the nature of a prediction[6]. When a model correctly predicts the positive class, for example, that a fraud has occurred, that is a true positive. When a model predicts that a fraud has not occurred and that prediction is correct, that is a true negative. A false positive occurs when a model predicts that a fraud has occurred, even though it in fact has not. A false negative is the other way around where a model has predicted it to not be a fraud, but it in fact is a fraud.
- **Overfitting and underfitting:** Overfitting and underfitting are two important terms to understand to make sense of why a model is performing as it is[7]. A model that is underfitted performs poorly on the data used to train it, while an overfitted model performs well on the training data but poorly on the evaluation or test data. An underfitted model has not learned enough from the training data while an overfitted model has fit too well to the training data by memorizing too exactly in a way that makes it difficult for the model to perform well on new data.
- **Oversampling and undersampling:** Two different techniques that can be used to handle imbalanced datasets in machine learning[8]. Imbalanced data in machine learning classification refers to data that has an uneven distribution of positive and negative samples. Oversampling increases the prevalence of the minority class, while undersampling reduces the prevalence of the majority class.

[6] (Google, *Classification: True vs. False and Positive vs. Negative,* 2022*)*
[7] (*AWS Documentation, Model fit: Underfitting vs. Overfitting,* 2016*)*
[8] (Shelke, M. S., Deshmukh, P. R., & Shandilya, V. K, *A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique,* 2017)

- **Loss:** The loss of a machine learning model is calculated by using a loss function[9]. The loss represents the error between the correct results and the model's predictions and can be calculated in many different ways.
- **Bias:** Bias is a term that can have a variety of meanings in machine learning. In this report, we refer to bias in the context of shifting a model's predictions in a direction that can reduce the loss or error.

2.3 Machine learning metrics

When evaluating the results of a binary classification machine learning model there are some relevant metrics that can be used:

- **Accuracy:** The accuracy can be divided into two categories. The positive accuracy represents how well the model predicted positive samples, while negative accuracy represents how well the model predicted negative samples. The formula for positive accuracy is:
  $positive\ accuracy = 100 * (correct\ positive\ predictions / number\ of\ positive\ samples)$
  and for negative accuracy:
  $negative\ accuracy = 100 * (correct\ negative\ predictions / number\ of\ negative\ samples$
- **Receiver Operating Characteristic (ROC) and Area Under the Curve (AUC):** ROC is a curve of true positive rates at the Y-axis compared to false positive rates at the X-axis for different classification thresholds[10]. This means that the ideal point is the top left corner. AUC can then be calculated on the ROC to show how good the model generally predicts the outcomes. The result is a number between 0 and 1.
- **Matthews correlation coefficient (MCC):** The Matthews correlation coefficient is used as a measure of the quality for machine learning classifications[11]. The correlation coefficient takes true and false positives and negatives into consideration in a balanced way to account for differences in class size. Simplified, it can be explained as a correlation coefficient between -1 and +1, where +1 are perfect predictions, 0 average random predictions and -1 inverse predictions.

2.4 Neural networks

Neural networks, or artificial neural networks (ANNs) is a branch within machine learning that uses a large number of linked processing nodes with different weights to simulate how neural networks work in

[9] (Wang, Q., Ma, Y., & Tian, Y, *A Comprehensive Survey of Loss Functions in Machine Learning,* 2020*)*
[10] (Pedregosa et al. *Scikit-learn: Machine Learning in Python*, 2011*)*
[11] (Pedregosa et al. *Scikit-learn: Machine Learning in Python*, 2011*)*

the human brain[12]. ANNs can be useful for language translation, pattern recognition, image recognition, speech recognition, and image creation. The network consists of node layers with an input layer, one or more hidden layers, and an output layer. Each node in a layer contains a weight and threshold and is connected to a node in the next layer. The node is activated if the specified threshold value is reached and the node then sends data to the next layer in the network.

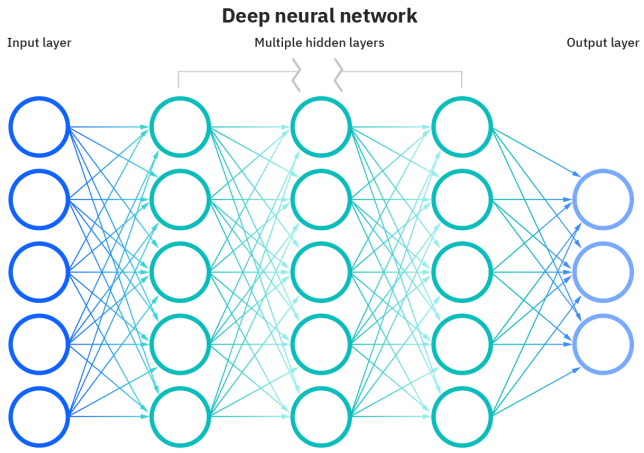

Figure 1: Visualization of a deep neural network[13]

There are a few terms that are relevant to understand in order to understand how a simple neural network works:

- **Optimization:** A neural network uses different optimization techniques during training to optimize the model's performance[14]. These methods enable the model to learn from the data by calculating the loss, the difference between the actual outcome and the predicted outcome, to optimize the weights. The idea is to find out the model's current gradient in order to calculate in what direction the different weights can be nudged to make sure the loss continues to decrease.
    - **Backpropagation:** A method used during optimization to go backwards through all layers in the model to adjust the weights along the way[15].
    - **Adam:** Adam is a gradient-based optimization technique that requires a low computational capacity, has low memory requirements, does not get affected by diagonal rescaling of the gradients, and is good for larger problems in regards to data and parameters[16].

[12] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019)
[13] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019*)*
[14] (Rajendra P., Hanumantha Ravi. P. V. N., Gunavardhana Naidu T., *Optimization methods for deep neural networks,* 2021)
[15] (Li, J., Cheng, J.-hang, & Huang, F, *Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement,* 2012)
[16] (Diederik P. Kingma, Jimmy Ba, *Adam: A method for stochastic optimization,* 2017*)*

- **Activation function:** Used in neural network layers to transform the input from the preceding layer to an output in the current layer, which is then used as input in the next layer[17]. Without activation functions, neural networks would act more similarly to linear regression models with limited performance and learning ability. The output from the activation function has to pass a certain threshold to be passed to the next layer, or else it gets deactivated and does not pass on the output.
    - **Sigmoid activation function:** A non-linear activation function, and the most used one. It transforms values to be between 0 and 1, usually used as the last layer for binary classification. It has the following formula: $f(x) = 1/(1 + e^{-x})$.
    - **Relu (rectified linear unit) activation function:** Widely used non-linear activation function that makes sure not all neurons are activated at the same time and therefore makes the network more efficient as it performs less backpropagation during training. The function is defined as: $f(x) = max(0, x)$.
- **Dense layers:** A layer where all neurons of the layer are connected to every neuron of its preceding layer[18]. The following calculation is performed on the input: $output = activation(dot(input, kernel) + bias)$, where the bias refers to the model bias and activation refers to the layers activation function. The dot function computes the dot product between the inputs and the kernel along the last axis of the inputs and axis 0 of the kernel (using tf.tensordot).
- **Dropout:** A technique within machine learning that is used to address the problem of overfitted neural networks by randomly dropping neurons and their connections from the neural network during training[19]. This makes the neurons less co-adapted.

2.5 Random forest

Random forest is a machine learning algorithm used for classification and regression problems that combine the output of several decision trees to reach a single result[20]. Decision trees consist of a series of questions that help the decision tree reach a final answer. Every question makes up a node in the tree and the leaves represent the final decisions. Each tree starts with an initial basic question that is then followed by a series of other questions. Data that fulfill the criteria will pursue the "Yes" branch and the other will follow an alternative path until the tree has reached a conclusion.

---

[17] (Sharma, S., Sharma, S., & Athaiya, A, *ACTIVATION FUNCTIONS IN NEURAL NETWORKS,* 2020)
[18] (Chollet, F et al. *Keras.* 2015)
[19] (Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R, *Dropout: A Simple Way to Prevent Neural Networks from Overfitting,* 2014)
[20] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019)

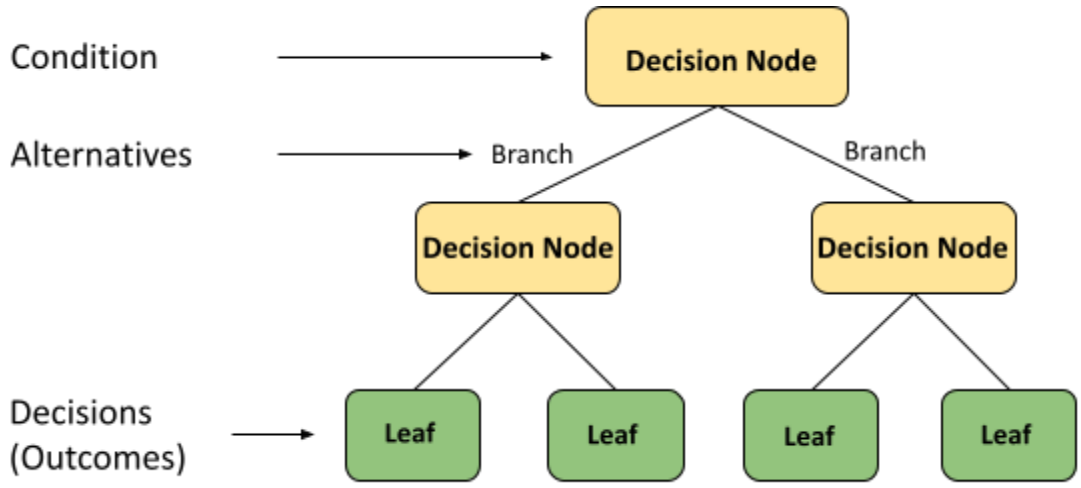

Figure 2: Illustration of a decision tree

In a random forest, the results from several decision trees are aggregated to finalize a decision. The random forest algorithm is an extension of the bagging method; a method where a random data sample is chosen from the training set and can be chosen more than once. The models are trained independently when multiple samples have been chosen, and the joint predictions from these models then yield a more accurate prediction. Random forest utilizes both bagging and feature randomness to make sure the forest of decision trees is uncorrelated. Feature randomness ensures each tree works with a random subset of features to ensure a low correlation between trees.

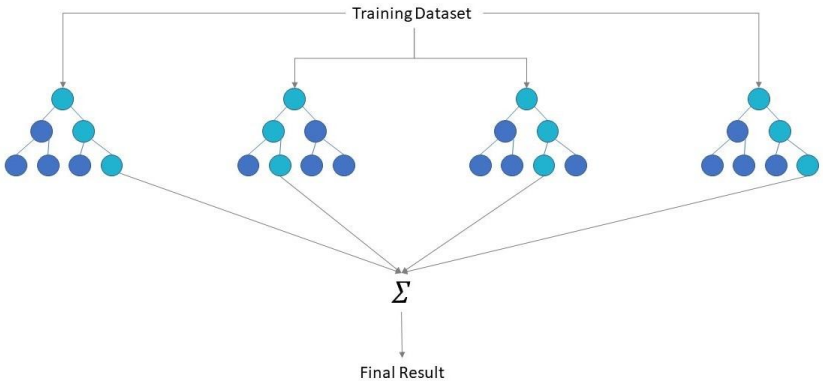

Figure 3: Visualization of a random forest Tree[21]

[21] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019*)*

2.6 Machine learning ensemble and stacking

Creating and combining multiple machine learning models results in an ensemble model[22]. Ensemble models combine hypotheses from the different models used to create a new hypothesis. An ensemble approach usually yields better predictive performance than a single model. Ensemble models tend to give better results when there are significant differences between the models used in the ensemble. Therefore, ensemble models seek to promote diversity among the models used.

Stacking is an ensemble process that uses different machine learning models in sequence, gathering the predictions of each model to create new features[23]. Model stacking only describes the process of combining different machine learning models into a final model, therefore there is no proper way to implement model stacking. A general model of stacking step by step can be seen in figure 3. In this study, the ensemble model was created with a random forest algorithm. The stacking method used is simple stacking. This involves training a single second-level model to make predictions based on the outputs of the individual models. The inputs to the second-level model are the predicted probabilities or class labels produced by the individual models.

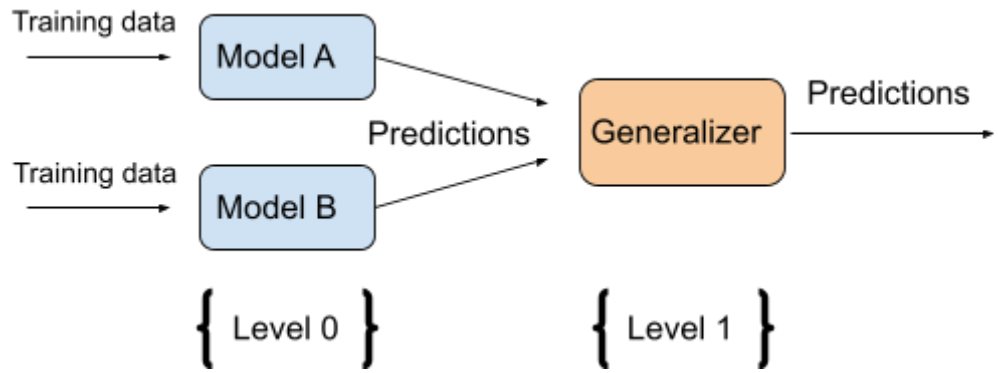

Figure 4: Visualization of an ensemble stacking model.

2.7 Standard scaling

The data obtained from datasets contains features of various dimensions and scales[24]. Different scales of the data features affect the modeling of a dataset adversely. It leads to a biased outcome of predictions in terms of misclassification error and accuracy rates. Thus, it is necessary to scale the data prior to

[22] (Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2019)
[23] (IBM Cloud Education, *Stack machine learning models: Get better results,* 2020)
[24] (Safa Mulani, *Using StandardScaler() ujfunction to standardize Python data*. DigitalOcean | The Cloud for Builders, 2022)

modeling. Standard scaling standardizes features by removing the mean and scaling to unit variance. This report uses the Sklearn standard-scaler.

2.8 Tools

Python is a programming language. Python is dynamically typed and garbage collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming. A library is a collection of related modules and contains packs of code that can be accessed and used by anyone repeatedly. In this study, several libraries were used.

Pandas is a library used for data analysis.

NumPy is used for working with arrays.

Imblearn provides tools when working with classification with imbalance classes.

Sklearn provides a selection of tools for machine learning and statistical modeling.

TensorFlow allows developers to create dataflow graphs.

Keras is an open-source software library that provides a Python interface for artificial neural networks.

Optuna is a hyperparameter optimization framework.

3. Methodology

3.1 Tools

To investigate the problem statement several computer tools were used to conduct the research. The research has only been performed using personal computers and has been centered around the following tools:

- The Python programming language and libraries such as Pandas, Numpy, Imblearn, Sklearn, Tensorflow, Keras, Optuna, and more.
- The Jupyter Notebook web-based interactive computing platform was used for executing the programming code, data exploration, plotting graphs, training and evaluating machine learning models as well as calculating the accuracy and performance of those.

3.2 Implementation

3.2.1 Data

The chosen data contains transactions made by credit cards in September 2013 by European cardholders and was collected and analyzed by Worldline and the Machine Learning group of Université Libre de Bruxelles[25]. The transactions occurred during a time period of two days and contain a total of 284,807 transactions which 492 are frauds. The dataset contains 30 features and 1 classification label. Frauds only account for 0.172% of all transactions, meaning the dataset is highly unbalanced. The data only consist of numerical input variables and 28 of these have been transformed using a principal component analysis (PCA), meaning the dimensionality of the dataset has been reduced. As the features are confidential there is no background information on the data and all features are named V1, V2, and so on. The features "Time", "Amount" and "Class" have not been transformed using PCA. The feature "Time" represents the seconds elapsed from the first transaction in the dataset. The feature "Amount" is the transaction amount and "Class" represents whether the transaction is a fraud or not.

The unbalanced data made the machine learning research more relevant to understand how to manage imbalanced data in machine learning.

[25] (Kaggle, *Credit Card Fraud Detection*. 2017)

3.2.2 Machine learning algorithms

The algorithms included in the research were one neural network algorithm, one random forest algorithm, one ensemble algorithm that used a stacking approach, and a second ensemble algorithm that used mean and threshold calculations to detect frauds. The reason for the choice of algorithms was based on the fact that the authors wanted to test algorithms that differed in structure from one another. Furthermore, the decision of the algorithms was also based on interest in the algorithms from an earlier machine learning course. Additionally, the authors wanted to test the difference in machine learning models compared to an ensemble model.

3.2.3 Data preparation

1. The feature “Time” was removed from the dataset as the algorithms don’t use any sequential analysis. This means the total number of features excluding the class label amounts to 29 features.
2. The feature “Amount” was logarithmically transformed to make it cover a smaller range.
3. The dataset was split into three categories where training accounted for 56%, validation for 24%, and testing for 20% Sklearn’s train-test-split function with no shuffling for reproducible results. The distribution was tuned according to favorable classification results and to ensure there existed enough data for training, validation and testing.
4. All data were transformed using standard scaling to ensure the mean was 0, the standard deviation 1, and the numerical range -5 to 5.

3.2.4 Neural network model training and evaluation

1. The model was defined using Keras from Tensorflow with the following layers linearly stacked:
    a. A densely connected hidden layer with 29 nodes, equivalent to the number of training features, an output size of 16 using the relu activation function.
    b. A dropout layer to reduce overfitting with a drop rate of 0.5.
    c. A dense output layer using the sigmoid activation function that returns the probability of a transaction being fraudulent.

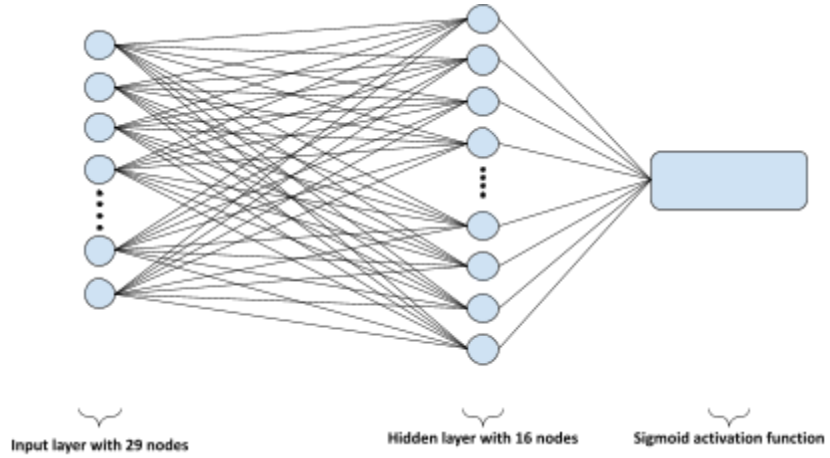

Figure 4: Illustration of the neural network before dropout

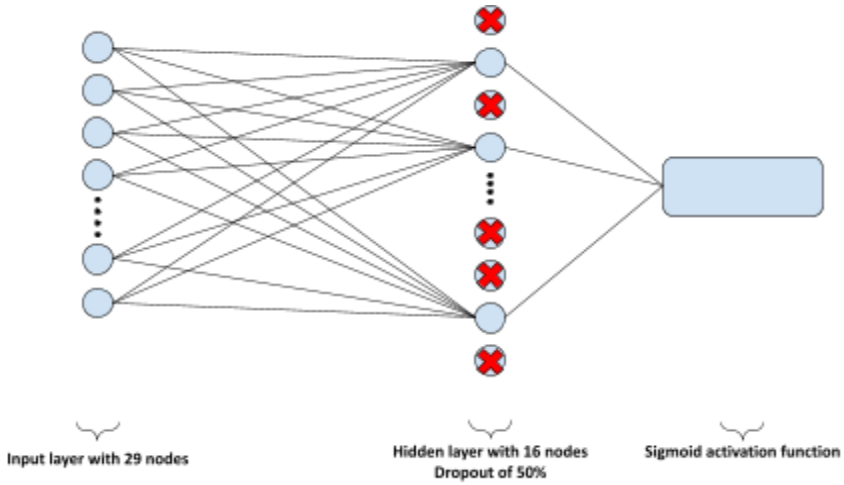

Figure 5: Illustration of the neural network after dropout

2. An early stopping method was defined for the model training to ensure the model would not be overfitted, but at the same time not underfitted. In practice, this means the model stops training a few rounds after it stops improving. The maximum number of epochs was set to 1000. Adam was used as the optimization function with a learning rate of 0.001. The minibatch size was set to 2048. No advanced process was performed to tune the model parameters, but simpler trials were conducted to final optimal values.
3. To handle the unbalanced dataset an approach that resampled the dataset by oversampling the true samples was chosen. The true samples were randomly shuffled and repeated using Tensorflow functions, and then included in a new resampled dataset where the true and negative samples had equal weight.

4. As a last step the model is trained with the resampled dataset using the early stopping mechanism and the validation data for validation. After training, the model's accuracy was determined by calculating the overall accuracy, accuracy of detecting frauds, and accuracy of detecting non-frauds.
5. The classification threshold used in the predictions was set to 0.5.

3.2.5 Random forest model training and evaluation

1. For the random forest implementation, a random forest algorithm that handles unbalanced data was chosen using Imblearn's BalancedRandomForestClassifier. The algorithm randomly undersamples false samples in each bootstrap sample to balance it.
2. To optimize the model performance a hyperparameter tuning algorithm from Optuna was used with the validation data to maximize the accuracy of detecting frauds. The search space for each hyperparameter was determined by trialing different options and realizing an approximate span within which the optimal solution could lie within. This was done to make sure the hyperparameter tuning could be executed in a short amount of time without the need of searching within an infinite space. The following hyperparameters and search spaces were used:
    a. n_estimators (70 - 130): The number of trees in the forest.
    b. max_depth (2 - 30): The maximum depth of a tree.
    c. min_samples_split (2 - 20): The minimum number of samples required to split an internal node.
    d. min_samples_leaf (2 - 15): The minimum number of samples required to be at a leaf node.
    e. random_state (0 - 100): The randomization state of the algorithm.
3. As a final step the model was retrained with the best hyperparameters. After training, the model's accuracy was determined by calculating the overall accuracy, accuracy of detecting frauds, and accuracy of detecting non-frauds.
4. The classification threshold used in the predictions was set to 0.5.

3.2.6 Ensemble approach: Stacking base model predictions to a random forest

1. For the ensemble implementation, the same algorithm and search space was used as for the original random forest model, meaning Imblearn's BalancedRandomForestClassifier.
2. Before training the ensemble model, the original random forest and neural network models were evaluated using the validation dataset. The validation dataset was used to measure the performance of these models by predicting whether a given transaction was fraudulent or not. The

predictions from these models, which were probabilities ranging from 0 to 1, were then used as the input data for the ensemble model. The ensemble model was trained to make predictions based on these probabilities from the two original models. In other words, the ensemble model was trained using the predictions from the random forest and neural network models as its input features. This allowed the ensemble model to combine the predictions of the two individual models in order to make more accurate predictions about fraudulent transactions.

3. The hyperparameter tuning was performed using the same approach and search spaces as the 2nd step of the random forest training and evaluation.
4. As a final step the model was retrained with the best hyperparameters. After training, the model's accuracy was determined by calculating the overall accuracy, accuracy of detecting frauds, and accuracy of detecting non-frauds.
5. The classification threshold used in the predictions was set to 0.5.

3.2.7 Model comparison

To compare the different model's performances a few additional metrics and graphs were used. The following comparisons were used on both the validation and test data:

1. ROC using Sklearn.
2. ROC AUC using Sklearn.
3. MCC using Sklearn.
4. Number of frauds detected by all models or different combinations of models, such as frauds only detected by model 1 and model 2. This calculation was not part of any existing package.

3.2.8 Baseline

To create a comparable baseline for all results the following steps were performed:

1. According to the data distribution of 99.83% false samples and 0.17% true samples, zeros and ones were randomized with that distribution as probability. This was done for both the validation and test data according to their data sizes. This resulted in binary baseline predictions for both the validation and test dataset.
2. To extend those predictions to a range of values between 0 and 1, instead of just binary, every 1 was uniformly distributed between 0.5 and 1, while every 0 was uniformly distributed between 0 and 0.49. This was done as the classification threshold was set to 0.5 for all models. This step generated continuous predictions for both the validation and test dataset between 0 and 1.

4. Results

4.1 Hyperparameters for random forest & ensemble

Hyperparameters	Value
n_estimators	119
max_depth	25
min_samples_split	15
min_samples_leaf	7
random_state	3

Table 1: Hyperparameters and their assigned values for the random forest model.

Hyperparameters	Value
n_estimators	110
max_depth	6
min_samples_split	13
min_samples_leaf	3
random_state	9

Table 2: Hyperparameters and their assigned values for the ensemble random forest model.

4.2 Accuracy results

Algorithm	Accuracy on frauds	Accuracy on non-frauds
Neural network	**88.5%**	96.3%
Random forest	86.7%	**98.7%**
Ensemble	**88.5%**	95.4%
Baseline	0.0%	**99.8%**

Table 3: Accuracy of the neural network, random forest, and ensemble model on the validation data. Also including baseline results.

Algorithm	Accuracy on frauds	Accuracy on non-frauds
Neural network	90.7%	96.6%
Random forest	86.9%	**98.4%**
Ensemble	**93.3%**	96.0%
Baseline	0.0%	**99.8%**

Table 4: Accuracy of the neural network, random forest, and ensemble model on the test data. Also including baseline results.

4.3 ROC & AUC comparison

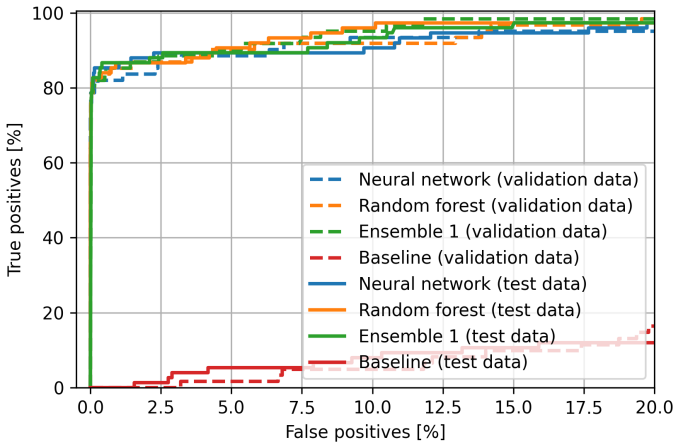

Figure 6: ROC on the validation and test data.

Algorithm	ROC AUC score on validation data	ROC AUC score on test data
Neural network	0.966	0.986
Random forest	0.978	**0.987**
Ensemble	**0.989**	0.985
Baseline	0.510	0.524

Table 5: ROC AUC scores on the validation and test data for the neural network, random forest, and ensemble model. Also including baseline results.

4.4 MCC comparison

Algorithm	MCC score on validation data	MCC score on test data
Neural network	0.263	0.297
Random forest	**0.331**	**0.399**
Ensemble	0.245	0.284
Baseline	-0.001	-0.002

Table 6: MCC scores on the validation and test data for the neural network, random forest, and ensemble model. Also including baseline results.

4.5 Results distribution

The table illustrates how different frauds in the sets were detected by the different models and how some frauds were only detected by a specific model. The frauds detected sum up to the total frauds in the sets.

Category	Fraud count (validation data)	Fraud count (test data)
Total frauds in set	61	75
Frauds no model detected	7	5
Frauds detected by the neural network, random forest, and ensemble model´s	53	66
Frauds only detected by the neural network model	0	0
Frauds only detected by the random forest model	0	0
Frauds only detected by the ensemble model	0	2
Frauds only detected by the neural network and random forest models	0	0
Frauds only detected by the neural network and ensemble models	1	2
Frauds only detected by the random forest and ensemble models	0	0

Table 7: A table showing how the distribution differs on detected frauds between the neural network, random forest, and ensemble model on the validation and test data.

5. Discussion

5.1 Hyperparameters for random forest & ensemble

The results from performing hyperparameter tuning on the models based on the random forest algorithms can be seen in tables 1 and 2. The hyperparameters tuned for the random forest and ensemble model were very similar and might derive from the models being highly correlated. The ensemble model uses a simple stacking approach with the model outputs from the random forest and neural network, which means the random forest predictions will have a great impact on the ensemble model's predictions. This link can be an explanation for the hyperparameter similarities.

As each tuned hyperparameter lies within a good margin of its search interval it can be concluded that the model’s are quite optimized in terms of hyperparameter choice.

5.2 Accuracy results

The ensemble model had the highest accuracy (88.5%) on detecting frauds on the validation data together with the neural network model (88.5%), while the ensemble performed best on the test data (93.3%). The ensemble model performed worse than both the other models on detecting non-frauds for both the validation (95.4%) and test data (96%), while the random forest model achieved the highest accuracy on both the validation (98.7%) and test data (98.4%).

By comparing the accuracy for frauds and non-frauds on both the validation and test data for the three models it is possible to conclude that a higher fraud detection accuracy resulted in lower non-fraud detection accuracy. All models performed very similarly and this might indicate that the standard chosen classification threshold of 0.5 had an effect on what model achieved the highest fraud detection accuracy. The models’ performed way better than baseline on both the validation and the test dataset in terms of fraud detection accuracy, while it performed slightly worse for non-fraud accuracy.

5.3 ROC & AUC comparison

The ROC AUC scores for the three models on both the validation and the test data were very similar. On the validation data, they differ slightly while they are almost identical on the test data. The ensemble model has the highest score on the validation data which can originate from the validation data being used as the training dataset for this model. Hence, the ensemble model might be slightly overfitted to the

validation dataset compared to the random forest and neural network that have used the validation data for loss minimization and hyperparameter tuning respectively. Beyond that, it as clear that all models outperformed baseline significantly.

5.4 MCC comparison

As seen in table 6, the MCC scores on both the validation and test datasets for all models are quite low and close to 0. The neural network model archives the highest score on both the validation (0.331) and test data (0.399). As an MCC score of 1 represents all perfect predictions and 0 average or randomized predictions, this means the scores are generally only partially better than random. This can be a consequence of several different factors. One reason might be a relatively high number of either false positives or false negatives. As MCC takes into account both false and true positives and negatives in a way that aims to be fair on imbalanced datasets, this result can be seen as an indicator of models that generally do not perform well enough. Even though the MCC results were not optimal, the MCC scores were significantly higher than baseline which indicates the models generates some value.

5.5 Results distribution

Based on the results distribution table (table 7), it can be shown that the ensemble model detects a few more frauds than the other models, sometimes as the only detector and sometimes accompanied by the neural network.

5.5 Limitations

The data set used in this study contained a total of 284,807 transactions with 31 parameters. Among the total number of transactions, 492 of them were fraudulent transactions which make up 0.172% of the dataset. Because of the low number of fraudulent transactions, the dataset is highly imbalanced and because of this oversampling and undersampling methods have been used to train the algorithms with more balanced data. With a more balanced dataset, the algorithms would possibly have achieved greater accuracy.

The algorithms implemented in this report were random forest, neural network and one ensemble model using random forest. Because of the imbalanced dataset, some pre-processing as well as oversampling and undersampling methods had to be applied to the dataset before for optimal model performance. Moreover, this study used a simple implementation of the algorithms and they can be implemented in many different ways which means that this is an accuracy prediction with the algorithms implemented and the

implementation in this study may not be the best way to implement them. The neural network was implemented using the basic approach with few layers. A neural network with an increased number of hidden layers could have increased the model's performance as it would increase its possibility to detect patterns in the data. Another limitation in the neural network was the arbitrary choice of setting dropout to 0.5 and improper tuning of the other model parameters. A more sophisticated approach for tuning these values could increase the performance further.

Furthermore, to increase performance on detecting fraudulent credit card transactions, other algorithms can be implemented and checked on performance. In the study, the models are trained with the dataset used, and to more subjectively determine the model's accuracy in detecting fraudulent transactions, the models can be tested with other datasets.

6. Conclusion

This study has evaluated different machine learning algorithms to measure the fraud detection accuracy of fraudulent and non-fraudulent credit card transactions. The accuracy differences between the random forest, neural network, and ensemble model did not differ significantly. The highest fraud detection accuracy was achieved by the ensemble model, while the lowest fraud detection accuracy was achieved by the random forest model. Even though the ensemble model performed best with the regular threshold of 0.5, it did not achieve the highest ROC AUC score on the test data. The three models had a very similar ROC AUC score, but the neural network model achieved the highest with 0.987. This might in fact indicate the models, in reality, had a very similar accuracy but that the threshold of 0.5 was more favorable for the ensemble model in terms of fraud detection accuracy. Those figures were nuanced with MCC scores, who indicated potential performance issues on the models when weighing true and false positives and negatives in an equal manner. The MCC scores point to the neural network model as the best performing model, while leaving room for potential improvements on all models for future implementations. Overall all models performed better than baseline on all datasets and measurements, except non-fraud detection accuracy.

In conclusion, this means the ensemble model had the highest accuracy if the sole purpose is to detect fraud, while the three models achieve very similar scores when also considering the number of false positives. Beyond that the MCC scores indicate that the models can be improved even further.

6.1 Future work

The next step to improve from this report would be to first analyze and improve the MCC scores. The next step would be to evaluate other machine learning models to assess whether this approach was the most sufficient one. Thirdly, a bigger dataset with more fraudulent transactions would improve the models training on the fraudulent transactions and probably achieve a higher accuracy score. A study similar to this should be performed, but include more research regarding the machine learning model´s fit to the dataset and a larger dataset with more fraudulent transactions.

5. References

1. AWS Documentation. *Model fit: Underfitting vs. overfitting - amazon machine learning*. (2016, September 2). Retrieved September 26, 2022, from https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

2. Chollet, F., & others (2015). *Keras.* Retrieved September 25, 2022, from https://keras.io

3. Diederik P. Kingma, Jimmy Ba. (2017, January 30). *Adam: A method for stochastic optimization*. Retrieved December 5, 2022, from arXiv.org. https://arxiv.org/abs/1412.6980

4. *Federal Trade Commission | protecting America's consumers*. (2022, February). *Consumer Sentinel Network.* Retrieved October 06, 2022, from https://www.ftc.gov/system/files/ftc_gov/pdf/CSN%20Annual%20Data%20Book%202021%20Final%20PDF.pdf

5. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc. Retrieved September 28, 2022

6. Google. (2022, July 18). *Classification: True vs. false and positive vs. negative | machine learning | google developers*. Google. Retrieved September 29, 2022, from https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative

7. IBM Cloud Education. (2020, January 16). *Stack machine learning models: Get better results.* Retrieved September 29, 2022, from: https://developer.ibm.com/articles/stack-machine-learning-models-get-better-results/

8. Li, J., Cheng, J.-hang, & Huang, F. (2012, March 12). *Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement*. Retrieved September 22, 2022, from https://link.springer.com/chapter/10.1007/978-3-642-30223-7_87

9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research*, 12, 2825-2830. Retrieved December 5, 2022, from https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

10. Rajendra P., Hanumantha Ravi. P. V. N., Gunavardhana Naidu T. (2021, October 5). *Optimization methods for deep neural networks*. AIP Publishing. Retrieved December 5, 2022, from https://aip.scitation.org/doi/10.1063/5.0066319

11. Safa Mulani. (2022, August 3). *Using StandardScaler() function to standardize Python data*. DigitalOcean | The Cloud for Builders. Retrieved December 5, 2022, from https://www.digitalocean.com/community/tutorials/standardscaler-function-in-python

12. Sharma, S., Sharma, S., & Athaiya, A. (2020, April). *ACTIVATION FUNCTIONS IN NEURAL NETWORKS* (12th ed., Vol. 4, pp. 310–316). IJEAST. Retrieved September 25, 2022, from https://www.ijeast.com/papers/310-316,Tesma412,IJEAST.pdf

13. Shelke, M. S., Deshmukh, P. R., & Shandilya, V. K. (2017, April 4). *A Review on Imbalanced Data Handling Using Undersampling and Oversampling Technique*. Retrieved September 27, 2022, from https://web.archive.org/web/20180602042807id_/http://www.ijrter.com/papers/volume-3/issue-4/a-review-on-imbalanced-data-handling-using-undersampling-and-oversampling-technique.pdf

14. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. *Journal of Machine Learning Research*, *15*, 1929-1958–1958. Retrieved September 24, 2022, from https://www.jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

15. Staff, the P. N. O., & Staff, D. P. I. P. and C. T. O. (2019, July 5). *Imposter scams top complaints made to FTC in 2018*. Federal Trade Commission. Retrieved October 02, 2022, from https://www.ftc.gov/news-events/news/press-releases/2019/02/imposter-scams-top-complaints-made-ftc-2018

16. Stuart, S. (2018, November 11). *Paypal vs. fraud – have no fear, machine learning is here !* Technology and Operations Management. Retrieved October 11, 2022, from https://d3.harvard.edu/platform-rctom/submission/paypal-vs-fraud-have-no-fear-machine-learning-is-here/

17. Wang, Q., Ma, Y., & Tian, Y. (2020, April 12). *A Comprehensive Survey of Loss Functions in Machine Learning*. Retrieved September 15, 2022, from https://link.springer.com/article/10.1007/s40745-020-00253-5

18. Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles). (2017). *Credit Card Fraud Detection*. Kaggle. Retrieved September 29, 2022, from: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud