Data Science for Internal Audit in Banking

Refinement of an Internal Audit Alarmistic System with Machine Learning

Laura Inês Bleeker Casquinha Crisóstomo

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

DATA SCIENCE FOR INTERNAL AUDIT IN BANKING

REFINEMENT OF AN INTERNAL AUDIT ALARMISTIC SYSTEM WITH MACHINE LEARNING

by

Laura Inês Bleeker Casquinha Crisóstomo

Internship report presented as partial requirement for obtaining the Master's degree in Advanced Analytics

Advisor: *Prof Doutor* Flávio Luis Portas Pinheiro

ACKNOWLEDGEMENTS

Throughout the implementation process of the internship project and writing of this internship report, I have received a great deal of support and assistance.

I would like to express my gratitude and appreciation towards my colleagues at the Data & Analytics Team at the Department for Internal Audit at CGD, who made it possible for me to autonomously work on this project, and see the fruits of the results.

I would like to extend my sincere thanks to my thesis supervisor, Prof. Doutor Flávio Pinheiro, for providing guidance, feedback and insightful suggestions throughout this project.

Finally, I am extremely grateful to my family and closest friends for their encouragement and support, especially through the last year.

ABSTRACT

This report presents the work developed during the academic internship required for obtaining the Master's Degree in Data Science and Advanced Analytics. The internship took place in the area of Data & Analytics of the Department for Internal Audit of Caixa Geral de Depósitos (Portugal), from the 14th of September 2020 to the 13th of June 2021.

The internship's goal was the introduction of machine learning to the Department of Internal Audit. In particular, the implementation of three machine learning pipelines to aid in audit activities of the institution, which systematically analyze operations that stand out from the implemented alarm system. The alarm system triggers alerts when an event disobeys a predefined methodology. Each triggering event is reviewed and processed individually by the auditors, either by being classified as a confirmed error or as a false positive. Confirmed errors frequently lead to recommendations to rectify the operations, while false positives are closed without a recommendation. The alerts' triggers are defined by sets of arguably general and manually implemented rules, resulting in high trigger frequencies and low precisions. Trigger frequency, precision, and cost of miss rate differ for each alert.

Based on the alerts' trigger history data, three types of alerts were selected for improvements. The deployment of machine learning pipelines with classification models optimized the triggers' specificity while maintaining high sensitivity, which reduced the number of daily events that have to be reviewed by the auditors. This optimization maximizes the efficiency and productivity of the general alarm system and decreases the auditors' workload.

KEYWORDS

Data Science, Internal Audit, Data Analytics, Machine Learning, Supervised Learning, Binary Classification, Imbalanced Learning

INDEX

1. Introduction 1
   1.1. Company overview 2
   1.2. The team and activities 2
   1.3. Internship goals 3
2. Theoretical Framework 4
   2.1. Artificial Intelligence 4
   2.2. Machine learning 4
   2.3. Data mining 5
      2.3.1. Data pre-processing 6
      2.3.2. Modelling 9
      2.3.3. Evaluation 10
   2.4. Classification models 12
      2.4.1. K-Nearest-Neighbor 12
      2.4.2. Logistic Regression 12
      2.4.3. Decision Trees 13
      2.4.4. Random Forest 14
      2.4.5. Naïve Bayes Classifier 14
      2.4.6. Neural Networks 14
3. Tools and Technology 17
      scikit-learn 18
4. Project discussion 19
   4.1. Timeline 19
   4.2. Implementation of machine learning pipelines to DAI's alarm system 20
      4.2.1. Motivation 20
      4.2.2. Data Description 21
      4.2.3. Model and Analysis 27
      4.2.4. Results and Discussion 31
      4.2.5. Conclusions 33
5. Conclusions 35
   5.1. Connection to the master program 36
   5.2. Internship evaluation 36
   5.3. Limitations 37
   5.4. Lessons Learned 38

5.5. Future work 38
6. Bibliography...39

LIST OF FIGURES

Figure 1: CRISP-DM Model (Wirth and Hipp 2000) 5
Figure 2: 10-fold cross-validation (Berrar 2018) 10
Figure 3: Confusion matrix for binary classification (Chawla et al. 2002) 10
Figure 4: Example of a decision tree classifier (Loh 2011) 13
Figure 5: The Rosenblatt Perceptron (Rosenblatt 1958) 15
Figure 6: Internship Timeline 20
Figure 7: Data type percentages of the variables from Alerts A, B and C. The data is retrieved from the auditors' platform, where the data from the triggering events is stored. 22
Figure 8: Distribution of the target variables of Alerts A, B and C. P stands for the positive class, i.e., confirmed errors, whereas N stands for the negative class, i.e., false positives. 23
Figure 9: Histograms of a continuous variable from the dataset corresponding to Alert A. (1) The data in its original distribution before outlier removal. (2) The data in its original distribution after outlier removal. (3) The data after a log transformation. (4) The data after a log transformation and outlier removal. 24
Figure 10: The category distributions of the 10 most common categories of a nominal variable from the dataset corresponding to Alert A, from July 2018 to July 2021. The category legend is excluded from the figure to not disclose concrete information. 25
Figure 11: Recall / sensitivity (red) and specificity (blue) of the logistic regression classifier fitted to Alert A, in function of the parameter *class_weight,* ranging in the interval of [0,20]. 29
Figure 12: The workflow of the model implementation and monitoring. The first two steps (left to right) were completed once, whereas the area that is shadowed in light green is executed daily, in the depictured order. 31
Figure 13: The results of the implemented logistic regression models for Alerts A, B, and C, based on the alerts' data from 2020. The saturated red and blue colors indicate the percentages of events from the positive and negative classes that were kept by the models, whereas the transparent areas visualize the percentages of excluded events. 32
Figure 14: Risk / miss rate vs. decreased workload / specificity of a logistic regression model for Alert A, in function of the *class_weight* parameter. The implemented model is indicated with the black vertical line. 33

LIST OF TABLES

Table 1: Evaluation metrics for binary classification problems 11
Table 2: Python libraries used during the internship project 18
Table 3: Data structures of the alerts' datasets, both in the auditors' platform and SQL views (as of August 2021).. 22
Table 4: Comparison between the values of the recall and specificity of the logistic regression models for Alerts A, B and C, on the validation (obtained by cross-validation) and test sets, both extracted from the auditors' platform, and samples from the SQL views 30
Table 5: Values of the mean and standard deviation of the recall and specificity of the logistic regression models for Alerts A, B and C, in function of the *class_weight* parameter. The cells highlighted in green indicate the models that were chosen to be implemented. .. 32

LIST OF ABBREVIATIONS AND ACRONYMS

CGD: Caixa Geral de Depósitos

DAI: Department of Internal Audit (original abbreviation of the Portuguese designation Direção de Auditoria Interna)

AI: Artificial Intelligence

ML: Machine Learning

DSAA-DS: Data Science and Advanced Analytics with a major in Data Science

RPA: Robotic Process Automation

SQL: Structured Query Language

1. INTRODUCTION

Banking institutions have worked with information systems and large volumes of data for several decades. However, the true value and profit of all these years' worth of information have not been appreciated until quite recently. Although departments for information technology are part of the core of any big company, decentralized data science teams that work closely with specific departments are on the rise, as the applicability and profitableness of data science are still being researched and gaining collective attention. The internal audit function at CGD is among the departments mentioned above, and one such area with relatively unexplored potential in data science.

Metaphorically, internal audit is the final gatekeeper of the bank's defense, which makes the prevention of business anomalies and the recognition of risks by the auditors, key. Internal auditors evaluate corporate governance processes, i.e., risk management and compliance function, and traditionally work with sampling methods by extrapolating the results and treating eventual conclusions as issues. However, the real problem behind the issues might lie within the underlying and hidden data. Performing internal audit activities with the aid of business intelligence and data science can help to face these challenges.

In this context, the subject of the present report focuses on an internship in the Data & Analytics area of the Department of Internal Audit at Banco Caixa Geral de Depósitos. The Data & Analytics area is a team that cooperates closely with the auditors at DAI, aiding several information technology services and working with the databases where the audit projects are stored, among others.

The frequent mention of AI at CGD's sessions for the professional development of their employees and the consistent promotion of futuristic and modern applications of data science methodologies have created a strong interest in the innovation and modernization of data-based processes. This interest built the foundation of the internship program.

The primary objective of the internship was the introduction of machine learning to DAI, namely the optimization of the continuous audit process with ML pipelines. The responsibility of implementing the pioneer machine learning application in this area at DAI came with the privilege of autonomously processing each step of the ML workflow, from data gathering, preparing and processing, to model training, evaluation, deploying and monitoring.

This report will focus on the implementation of three structurally identical machine learning pipelines that were deployed to the continuous audit process of DAI, which systematically analyzes operations that stand out from the implemented audit procedures, detected by the alarm system. The alarm system is composed of over a hundred different alerts that continuously monitor operations in the institution. The alerts trigger when an event disobeys a predefined methodology, and each triggering event is reviewed and processed individually by an auditor. The events are classified either as confirmed errors or as false positives, leading to recommendations to rectify the former.

The alerts' trigger frequency, precision and gravity of miss rate depend on the manually implemented rules that define the triggers. The triggers' rules can be arguably general, and result in high trigger frequencies and low precisions. Daily, the auditors evaluate and classify the triggering events, recommending rectifications to confirmed events and labelling the falsely triggered events as false positives.

Based on data of events that were triggered and manually reviewed and labelled by the auditors, three alerts with the highest trigger frequencies and lowest precisions were selected for improvements. The internship project aimed to use classification models to reduce the daily samples of triggering events, by detecting and excluding false positives. Furthermore, the deployment of classification models aimed to maximize the specificity of the triggers, while maintaining a high sensitivity. Thus, the final objective of the project was the reduction the total number of daily events that have to be reviewed by the auditors and, hence, significantly maximize the alarm system's efficiency.

1.1. Company overview

Caixa Geral de Depósitos is a Portuguese wholly state-owned banking corporation and the second largest bank in Portugal. Established in Lisbon in 1876, CGD is the largest Portuguese financial group, having a presence in twenty-three countries across four continents. In addition, CGD is nationally recognized with a leading position in the Portuguese retail market, with more than four million customers in Portugal and assets ranging higher than 100bn€.

Currently, CGD is thoroughly present in almost all fields of the banking business, highlighting commercial banking, investment banking, brokerage, venture capital, real estate, asset management, specialized credit, and many more.

1.2. The team and activities

The internship at CGD was developed within the Data & Analytics Team of the Department of Internal Audit.

At CGD, the internal audit function is an objective and independent consulting and assurance activity, aiming to enrich and improve the institution's operations, individually and Group-wise. The main objective lies in aiding the CGD Group in successfully achieving its goals by assessing and optimizing the efficacy of several functions, such as governance processes and risk management.

The Department of Internal Audit is subdivided into four areas: Methodology, Data & Analytics, Audit, and Strategy & Planning.

The Data & Analytics area is a team that cooperates closely with DAI's auditors. The principal activities focus on optimizing information systems to support the audit activity, providing technical support to employees at DAI, and implementing structuring digital projects.

1.3. Internship goals

The internship is part of the ‘Programa Geração Caixa’ program, an initiative of CGD that offers master’s students a year-long opportunity to be integrated into several areas of the institution, transforming their academic knowledge into professional experience.

The internship project focused on DAI’s continuous audit process, which is composed of an alarm system that includes over a hundred different alerts. The alerts are constructed by sets of rules that work as retrieval systems to detect potential conspicuous activity in various sectors of the institution. Specific events trigger the alerts, which an auditor proceeds to review and evaluate as either confirmed errors or false positives. An example of such an alert is the identification of wrongfully deducted commissions or specific anomalies within the commercial sector of the bank. Each alert differs in terms of trigger frequency, precision and gravity of a miss rate.

Three alerts from the alarm system were chosen for improvements, based on their high trigger frequencies and low precisions. ML pipelines were implemented for each of the chosen alerts, following analogue structures and deployment journeys. The pipelines enhance the alerts’ triggers by introducing binary classification models, which are trained with the manually labelled historical data from the events that triggered the alerts. The trained models operate over the daily events that are triggered by the alerts and aim to detect and exclude false positives. The models supplement the performance of the trigger, by maximizing their specificity and keeping the sensitivity as high as possible. Hence, the project aids the responsible auditors, who have to spend less time reviewing daily events.

The workflow of the ML project includes data acquisition and understanding, data preparation and pre-processing, model training, validation, evaluation, and finally, the implementation of the functioning classification models and assurance of their periodic and automatic application on the implemented alarm system.

This project was the pioneer application of machine learning to DAI. Apart from data understanding, the theoretical and technical components of this workflow were worked on autonomously.

The fundamental goal of this project is the long-term optimization of the alarm system’s efficiency, and consequently, a productivity incrementation of the auditors that review the chosen alerts.

2. THEORETICAL FRAMEWORK

This chapter presents a brief theoretical contextualization of some subjects that this report will mention, by describing essential key concepts to comprehend and recognize the underlying work.

2.1. Artificial Intelligence

The term 'Artificial Intelligence' was co-coined by John McCarthy, who defined AI as "the science and engineering of making intelligent machines" (J. Mccarthy 2007). Other popular definitions of AI include "[the automation of] activities that we associate with human thinking, activities such as decision-making, problem-solving, learning ..." (R. Bellman 1978), or "the art of creating machines that perform functions that require intelligence when performed by people" (Ray Kurzweil and Diane Jaroch 1990).

In the mid-twentieth century, Alan Turing, widely considered one of the founding fathers of computational and artificial intelligence, proposed the Turing Test (A. M. Turing 1950), which suggests that the four premises of intelligent computational behavior are natural language processing, knowledge representation, automated reasoning, and machine learning.

2.2. Machine learning

Tom M. Mitchell defined Machine Learning as "the study of computer algorithms that improve automatically through experience and by the use of data" and "a part of artificial intelligence" (T. M. Mitchell 1997). Another popular definition of machine learning is the ability to "adapt to new circumstances and to detect and extrapolate patterns" (S. J. Russell and P. Norvig 1995).

The underlying assumption in machine learning is that known data can be used to extrapolate knowledge about unseen data, by using a fitted model. The fitting process follows different learning paradigms and algorithms, which depend on the model that is chosen. In summary, as an application of AI, machine learning encompasses computational methods that calibrate data to a model by minimizing a loss function, without or with only partial human assistance.

The applications of machine learning are currently very diverse and interfere with many aspects of our daily lives. For example, personalized recommendations on streaming platforms or suggestions and advertisements on social media, or a solution to business problems such as customer churn prediction and credit risk assessment, share the main goal to predict or classify new information.

Peter Norvig subdivides machine learning into three different learning paradigms (S. J. Russell and P. Norvig 1995). The critical difference is the algorithms' learning process and purpose. These "types of feedback that determine the three main types of learning" are:

1. **Supervised learning** aims to "build a concise model of the distribution of class labels in terms of predictor features" (Kotsiantis 2007). This learning paradigm works with labelled data and algorithms that learn from input-output pairs and create a function that maps one to the other. Applications of supervised learning are classification and regression problems, which map qualitative and quantitative outputs to the inputs, respectively. There are several types of classification problems, however, this report will specify binary classification, which consists of predictive classification problems with two possible target classes.

2. **Unsupervised learning** is a learning paradigm that can be used with datasets that are neither labelled nor classified. The algorithms of this family aim to find patterns between the inputs, using similarity or dissimilarity measures, for example. Cluster analysis or segmentation is a type of unsupervised learning.

3. **Reinforcement learning** is a training method that rewards from a sequence of actions, where the main goal is to find the balance between exploration and exploitation in the decided actions of intelligent agents. A reinforcement learning agent aims to comprehend and interpret its environment, to take actions based on trial and error.

2.3. Data mining

Data mining describes the process of "discovery of structures and patterns in large and complex data sets," using methodologies such as statistical analysis and machine learning (Hand and Adams 2015). Thus, data mining is an application of machine learning (Kotsiantis 2007), as ML technologies are used to extrapolate patterns and information from data.

The implementation journey of a machine learning model passes through several data mining phases, which the "Cross-industry Standard Process for Data Mining" conveniently summarizes (Wirth and Hipp 2000). The CRISP-DM describes a standardized framework for data mining tasks, following a standardized methodology. This process allows for the coherent and analogue replication of tasks, valid for the internship project at hand. Figure 1 depicts the steps of the CRISP-DM.

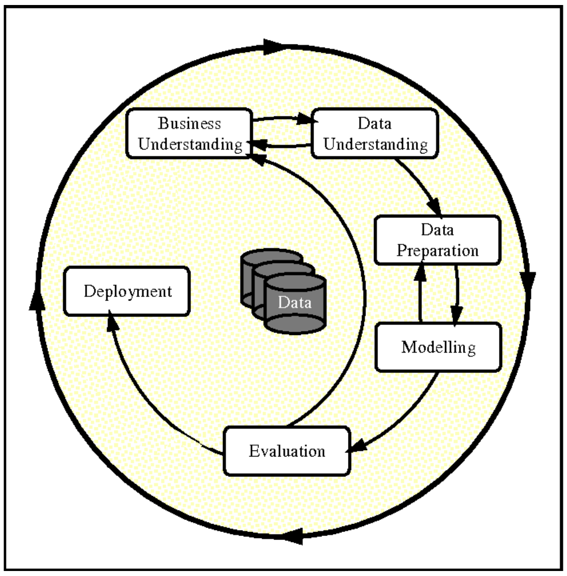

Figure 1: CRISP-DM Model (Wirth and Hipp 2000)

Considering that the main project of this internship consists of the implementation journey of classification models, which are a type of supervised learning, the following chapters are specified this ML learning paradigm.

2.3.1. Data pre-processing

Data pre-processing consists of the dataset necessary operations to transform raw data into input to train the ML algorithms. The steps to pre-process data can be more time-consuming and challenging than data mining (Fayyad and Stolorz 1997) since their order and logic are flexible and subjective.

Data pre-processing can be subdivided into two general tasks, namely "data preparation, compounded by integration, cleaning, normalization and transformation of data; and data reduction tasks; such as feature selection, instance selection [and] discretization" (García, Luengo, and Herrera 2015). This chapter detailly describes the usual order the data preparation steps follow.

2.3.1.1. Data Integration

Data integration describes the process of gathering, joining, and verifying data from various data sources, which can present several challenges, including conflicting data formats, feature redundancies, and logical inconsistencies. An exploratory data analysis identifies and addresses these challenges.

Regarding the problem of redundancy in datasets, Zena M. Hira and Duncan F. Gillies (Hira and Gillies 2015) state that "in machine learning as the dimensionality of the data rises, the amount of data required to provide a reliable analysis grows exponentially", and Bellman (Bellman 1972) referred to this phenomenon as the "curse of dimensionality". As a result, a dataset with redundant features is more extensive than necessary, resulting in higher modelling times and worse model performances (Limshuebchuey, Duangsoithong, and Windeatt 2015). However, two popular techniques (Hira and Gillies 2015) can overcome this challenge: feature extraction and feature selection. According to Terry Windeatt (Duangsoithong and Windeatt 2010), feature extraction transforms the original dataset by lowering the data dimensions, and the process of feature selection chooses an optimal set of features, with the aid of dependence measures, mutual information, or the chi-squared test. The step of feature selection is generally performed as the last step before the modelling process.

2.3.1.2. Data Cleaning

Data cleaning consists of identifying and treating logical inconsistencies in data, for instance, contradictory or impossible information. Missing data imputation and noise identification can be integrated into this step and require a deeper understanding of the data and human audit.

Missing data imputation deals with incomplete or missing parts of the dataset. These challenges can occur for several reasons, such as issues with the databases or faulty user input. Whether and how the missing values are imputed depends on the variable's interpretation and chosen algorithms. Generally, imputing missing values generates better results than leaving the values blank (García et al. 2015). Missing data imputation has a wide range of complexity. Simple methods include the imputation of missing values with the mean or mode of the variable. More complex methods include techniques such as the KNN imputer, which imputes missing values based on the mean (quantitative variables) or mode (qualitative variables) of a selected set of similar records. This technique follows the K-nearest neighbor algorithm, which is more detailly described in chapter 2.4.1.

Noise identification describes the process of identifying observations with abnormally high or low values compared to the other observations of the same feature. These anomalies, also called outliers, can occur due to incoherent values with the data distribution, as well as data errors. Thus, the methodology to remove noise relies on the data characteristics and data distribution. With normally distributed data, records can be removed with an exclusion threshold based on their deviation from the mean. If the data follows a log-normal distribution, the log-transformation can be used to normalize the data before the aforementioned operation. Alternatively, data anomalies from data that is not sufficiently normally distributed can be excluded when records fall outside the Inter-Quartile-Range, which describes the middle 50% of descendingly ordered data. (Ilyas and Chu 2019).

2.3.1.3. Data Transformation

Data transformation describes operations that enhance the informational value of the features. Feature engineering is the "practice of constructing suitable features from given features that lead to improved predictive performance" (Nargesian et al. 2017) and includes several operations that fall under the category of data transformation. For instance, binning quantitative data or binarizing qualitative data, and creating new features, are methods to adjust data types.

The identification of informative attributes reduces the dataset to solely significant features. Feature selection is "primarily focused on removing non-informative or redundant predictors from the model". The adequate feature selection method can be chosen depending on the nature of the problem and the data types of the features in question. Langley categorized feature selection methods into two broad groups: filter and wrapper methods (Talavera 2005). Filter methods are independent of the used algorithm, whereas wrapper methods use the algorithm as an evaluation metric.

Dash and Liu (Dash' and Liu 1997) propose dividing evaluation functions used for feature selection into five categories. The five categories are listed below, adjusted to binary classification problems.

- **Distance measures** compare two features based on the difference between the two-class conditional probabilities.
- **Dependence measures** describe the ability of a feature to predict the target variable. The occurrence of a term in a feature A is independent of the occurrence of a class of target variable B if $P(AB) = P(A)P(B)$.
  This report will mention the chi-squared test, which is a statistical hypothesis test for independence between variables. The chi-squared test measures the dependence between each feature and the target variable. Features independent of the target variable, guided by the chosen significance level, can be discarded from the dataset. The chi-squared test uses the following statistic, where O_i and E_i are the observed and expected frequency of the feature in question, and N is the number of observations (Pearson 1900):

$$\mathrm{X}_{\mathrm{c}}^{2} = \sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{(\mathrm{O}_{\mathrm{i}} - \mathrm{E}_{\mathrm{i}})^{2}}{\mathrm{E}_{\mathrm{i}}} \tag{1}$$

- **Information measures** evaluate features based on the information gain that they offer to the ML model. A popular information measure is feature selection based on mutual information. This method measures how much information a feature and the target variable share, which is proportional to the extent of how much information the presence or absence of a feature contributes in predicting the target variable. In the following formula, $p_{(X,Y)}$ is the joined probability mass function of feature X and target Y, and $p_{(X)}$ and $p_{(y)}$ are the marginal probability mass functions of X and Y (Cover and Thomas 2005):

$$\mathrm{I}(\mathrm{X};\ \mathrm{Y}) = \sum_{\mathrm{y}\in\mathrm{Y}}\sum_{\mathrm{x}\in\mathrm{X}} \mathrm{p}_{(\mathrm{X},\mathrm{Y})}(\mathrm{x},\mathrm{y}) \log\left(\frac{\mathrm{p}_{(\mathrm{X},\mathrm{Y})}(\mathrm{x},\mathrm{y})}{\mathrm{p}_{(\mathrm{X})}(\mathrm{x})\ \mathrm{p}_{(\mathrm{Y})}(\mathrm{y})}\right) \tag{2}$$

- **Consistency measures** select features that better define consistent logical decisions about the training data set (A. Arauzo-Azofra, J. M. Benitez, and J. L. Castro 2008).
- **Classifier Error Rate** measures evaluate features based on their impact on the predictive ability of the algorithm.

Models that are trained on datasets with skewed target class distributions deal with imbalanced learning. Imbalanced learning problems can be addressed in two ways (Chawla et al. 2002) assigning cost weights to the classes or with resampling methods. Resampling methods adjust the discrepancy between class frequencies of the target variable, either by oversampling the minority class or undersampling the majority class.

One such technique is the synthetic minority oversampling technique, also referred to as SMOTE (Chawla et al. 2002), where “the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors “. The authors describe the SMOTE algorithm along the following lines:

- “Synthetic samples are generated in the following way: Take the difference between the feature vector (sample) under consideration and its nearest neighbour. Multiply this difference by a random number between 0 and 1, and add it to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general”.

2.3.1.4. Data Normalization/Standardization

Data normalization reformates the variables to range in a chosen interval, frequently of [0,1], whereas standardization transforms the data to a mean of 0 and a standard deviation of 1. Several possible scalers normalize data, the choice of which depends on the nature of the data, i.e., the presence of data anomalies.

Data normalization or standardization are essential to avoid different data units, leading to bias in distance-based algorithms. After this data cleaning step, the variables have a similar weight and can be easily compared to the target variable.

A standard normalization method is the MinMax Normalization, which rescales the values of the features to range in a specified interval. The following transformation is applied to an attribute x, which will range in the interval $[A, B]$:

$$x' = \frac{x - \max(x)}{\max(x) - \min(x)}(B - A) + A \tag{3}$$

Another popular standardization method is the z-score normalization, which provides a zero-mean and unit variance. In the formula below, $\bar{A}$ and σ_A are the sample mean and standard deviation of attribute A:

$$x' = \frac{x - \bar{A}}{\sigma_A} \tag{4}$$

2.3.2. Modelling

Modelling combines choosing, configuring, and evaluating a machine learning model that extracts patterns from data to solve a prediction or classification problem. The modelling assumptions and objectives of the problem are the primary factors that dictate the choice of the machine learning technique.

As previously mentioned, the primary goal of ML models is to learn from data to predict or classify new information. However, if the model is built and validated with all available data, there is no data left to estimate the model's behavior and performance on new and unseen data.

This type of model training can lead to two problems: under- and overfitting. Underfitting occurs when the model is "poorly adjusted to the data, suffering from high error both in training and test (unseen) data," while overfitting refers to models that are "too tightly adjusted to data offering high precision to known cases but behaving poorly with unseen data" (García et al. 2015).

Since model validation is essential for building a supervised model, splitting the data into train, validation, and test sets are considered good practices. The model is trained with the training set, the parameters are tuned with the validation set, and the final estimation of the model's performance on new instances is obtained with the test set. (Xu and Goodacre 2018)

A popular method to construct an empirical performance estimation of a model is the cross-validation technique. Cross-validation is "one of the most widely used data resampling methods to estimate the true prediction error of models and to tune model parameters [and] to assess the generalization ability of predictive models and to prevent overfitting" (Berrar 2018).

In his paper "Cross-validation" (Berrar 2018), Daniel Berrar explains k-fold cross-validation as the following:

- "In k-fold cross-validation, the available learning set is partitioned into k disjoint subsets of approximately equal size. Here, "fold" refers to the number of resulting subsets. This partitioning is performed by randomly sampling cases from the learning set without replacement. The model is trained using k − 1 subsets, which, together, represent the training set. Then, the model is applied to the remaining subset, which is denoted as the validation set, and the performance is measured. This procedure is repeated until each of the k subsets has served as validation set. The average of the k performance measurements on the k validation sets is the cross-validated performance."

Figure 2 illustrates 10-fold cross-validation. In the depicted case, nine subsets of the data are used as training data, while the fold left out is used to test the model. After each fold has served as the test sample, the average and standard deviation of the chosen evaluation measure's values is used to estimate the model's performance.

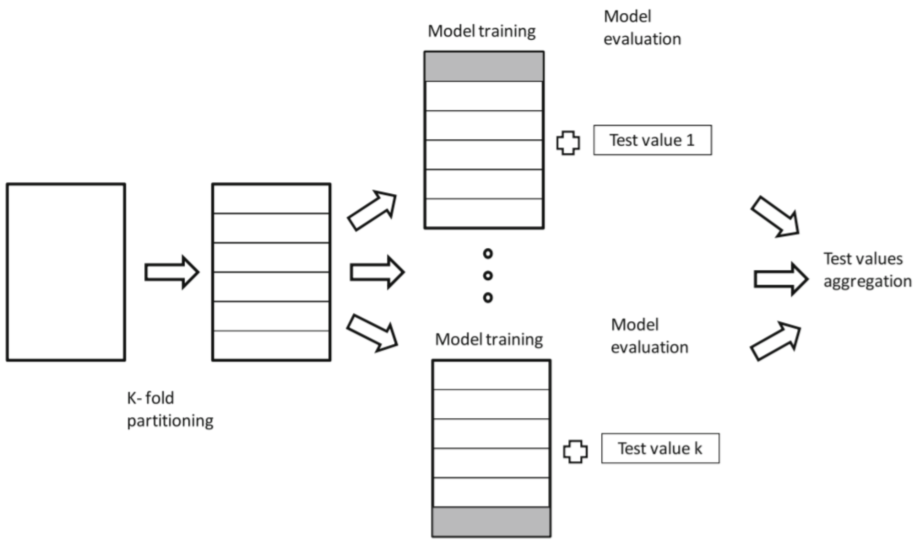

Figure 2: 10-fold cross-validation (Berrar 2018)

2.3.3. Evaluation

Model evaluation is the way of estimating the performance of a model based on different metrics and techniques, the choice of which depends on the data and objective of the problem. The model results can be assessed concerning the business problem and in light of different perspectives. The evaluation of the model also defines the potential and next steps of the project.

For binary classification problems, which this report focuses on, a confusion matrix summarizes the results of a model.

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Figure 3: Confusion matrix for binary classification (Chawla et al. 2002)

Different evaluation metrics weigh the values of the confusion matrix with different weights. Depending on the impact of model error, the metrics in the table below provide specific perspectives, ranging all in the interval of $[0,1]$, where a higher value indicates better performance (Dalianis 2018).

Evaluation metric	Formula
Sensitivity / Recall / True positive rate	$TPR = \frac{TP}{TP + FN}$
Specificity / True negative rate	$TNR = \frac{TN}{TN + FP}$
Miss rate / False negative rate	$FNR = \frac{FN}{FN + TP}$
Fall-out / False positive rate	$FPR = \frac{FP}{FN + TP}$
Accuracy	$A = \frac{TP + TN}{TP + TN + FP + FN}$
Precision	$P = \frac{TP}{TP + FP}$
$\boldsymbol{F_\beta}$ **- Score**	$F_\beta = (1 + \beta^2)\frac{P * R}{\beta^2 * P + R}$

Table 1: Evaluation metrics for binary classification problems

The accuracy measure indicates how many instances are correctly classified out of the total of results. As the arithmetic mean of the precision for both classes, accuracy is a good performance indicator for class-balanced datasets (Canbek et al. 2017).

The measures of precision and recall are metrics that indicate the performance of retrieval systems composed of a positive and negative class. Precision measures the number of instances that are correctly retrieved out of all the retrieved instances of the model, and recall measures the number of positive instances that were retrieved out of all the positive instances in the dataset. Thus, a high precision indicates that the model retrieved mostly positive instances, whereas a high recall indicates that most positive instances were retrieved. Specificity measures the identification performance of instances that are not relevant.

The F_β – Score is defined as the weighted average of both the precision and recall scores, measuring "the effectiveness of retrieval with respect to a user who attaches β times as much importance to recall as precision" (Blair 1979). The chosen value of β defines the relevance and power of either precision or recall. With $\beta = 1$, this measure refers to the F1-Score, the harmonic mean between precision and recall. With $\beta \in [0,1[$, precision is valued higher than recall, and with $\beta > 1$, recall is weighted more than precision.

2.4. Classification models

This chapter will focus on classification models that will be mentioned throughout this report. The models are specified in the perspective of binary classification problems, accordingly to the internship's projects. Apart from their functionality and description, the learning algorithm of the models will also be described.

2.4.1. K-Nearest-Neighbor

The K-Nearest-Neighbor (KNN) is a non-parametric classification algorithm and method of instance-based learning. The algorithm classifies new instances based on a majority vote of their immediate 'neighbors' classes. The 'neighborhood' consists of the k most similar instances (Guo et al. 2003). The training algorithm of a KNN model is mainly guided by the decisive and manually predefined value of k. Optionally, weights can be assigned to each variable to introduce a difference in relevance between the information points. Models trained with many neighbors tend to disregard low-frequency classes, whereas models trained with few neighbors are more susceptible to outliers (Murphy 2015).

The KNN classifier calculates the distances between the instance that is to be labelled, and each instance in the dataset, and provides a classification according to the classes of the k closest instances. The Euclidean distance is the most commonly used distance measure to identify an instance's neighborhood (Murphy 2015). The Euclidian distance between two points $A = [a_0, a_1, \dots, a_i]$ and $B = [b_0, b_1, \dots, b_i]$, is the following:

$$d(A,B) = \sqrt{\sum_{i=0}^{N}(a_i - b_i)^2} \tag{11}$$

However, other metrics can be used.

2.4.2. Logistic Regression

Logistic regression is a parametric classification algorithm that represents regression analysis for binary classification. The goal is to determine the probabilities of each instance to belong to either class and assign a class accordingly.

Logistic regression is composed of a linear combination $\mathrm{y(x)}$ of the inputs x_i, adding an intercept term/bias ϵ, where N is the number of instances of the dataset (Murphy 2015).

$$\mathrm{y(x)} = \sum_{i=1}^{N} w_i x_i + \epsilon = w^T x + \epsilon \tag{5}$$

The values of the coefficients w_i can be obtained using the maximum-likelihood estimation probabilistic framework (Kuhn and Johnson 2013), which constitutes the training algorithm of the model.

The sigmoid or logistic function, also called the squashing function of logistic regression, maps the values or the linear combination between 0 and 1:

$$sigm(w^T x + \epsilon)) = \frac{1}{1 + e^{-(w^T x + \epsilon)}} \tag{6}$$

The result of the sigmoid function can be interpreted with the Bernoulli distribution, a particular case of the binomial distribution with $k = 1$. The Bernoulli distribution is the discrete probability distribution of a random variable that belongs to a class k with probability p. The Dutch-Swiss mathematician and physicist Daniel Bernoulli (1700–1782) defined the probability mass function as the following:

$$Ber(k, p) = p^k (1 - p)^{1-k} \quad for \;\; k \in \{0,1\} \tag{7}$$

Combining these steps generates the logistic regression:

$$p(y|w, x) = Ber(y|\, sigm(w^T x + \epsilon)) \tag{8}$$

2.4.3. Decision Trees

Loh (Loh 2011) defines decision trees as “prediction models constructed by recursively partitioning a data set and fitting a simple model to each partition.”. Tree-like structures, like Figure 4, can visually represent the resulting model. Each branch represents a binary partition, and the terminal nodes/leaves represent simple models which apply to that cell only.

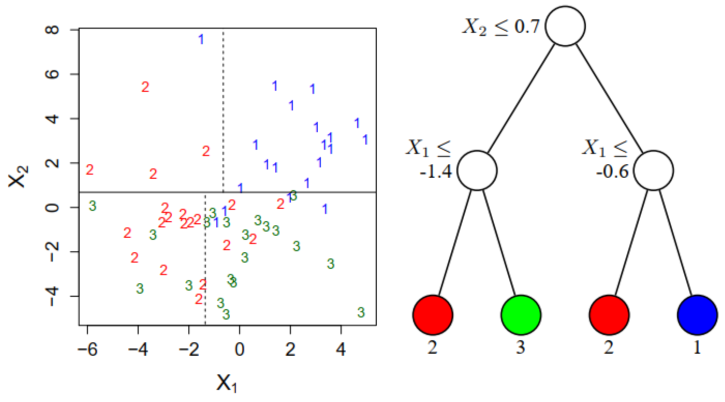

Figure 4: Example of a decision tree classifier (Loh 2011)

At each internal node in the tree, the input passes through a binary test. The outcome of the test defines the sub-branches for the partitioned input. After several of these partitions, a leaf node determines the output class and prediction for each instance. The splitting criteria of decision trees follow impurity functions, which measure the extent of class purity for the data subsets that the partitions generate. Generally, the split that decreases the impurity most, is chosen.

C4.5 (R. Quinlan, M. Kaufmann, and S. L. Salzberg 1994) is a commonly used classification tree algorithm that follows the abovementioned approach, using information entropy as its impurity function.

CART (Rutkowski et al. 2014) uses 10-fold cross-validation, whereas the C4.5 method employs a heuristic formula to estimate error rates. The CART algorithm allows diverse types of data, both quantitative and qualitative. A majority vote determines the basic principle of classification for the tree leaves.

2.4.4. Random Forest

The Random Forest is an ensemble classifier, which consists of the multiple uses of decision trees. Opitz and Maclin define ensemble learning as "a set of individually trained classifiers […] whose predictions are combined when classifying novel instances […] often more accurate than any of the single classifiers in the ensemble" (Opitz and Maclin 1999).

Breiman (Breiman 2001) defines decision trees as the following:

- "A random forest is a classifier consisting of a collection of tree-structured classifiers $\{h(x, \Theta_k), k = 1, \dots\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x."

Random forests are relatively robust to outliers and noise and offer valuable insights into variable importance, correlation, and error estimates.

2.4.5. Naïve Bayes Classifier

The Naïve Bayes Classifier is a probabilistic classifier that relies on the Bayes Theorem, proposed by Thomas Bayes in 1702:

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}, \qquad P(B) \neq 0 \tag{9}$$

This classifier is based on the assumption that the data features are independent. This assumption named the classifier as 'Naïve' since it can rarely be verified. The Naïve Bayes classifier assigns the class probabilities to the instances, which are used as a classification prediction, as shown in the following equation, where $\hat{y}$ denotes the predicted class of an instance $X_k, k \in 0, \dots, p$, and $P(x_k|y_i)$ is given by Bayes Theorem.

$$\hat{y} = argmax_{y_i} \prod_{k=1}^{p} P(x_k|y_i)P(y_i) \tag{10}$$

2.4.6. Neural Networks

Neural networks are a family of algorithms that mimic biological neural networks and learning processes. For example, Rosenblatt (Rosenblatt 1958) proposed the perceptron, a single-layer neural network, as a binary classification algorithm. The perceptron is the simplest forward neural network and uses the threshold function as its activation function.

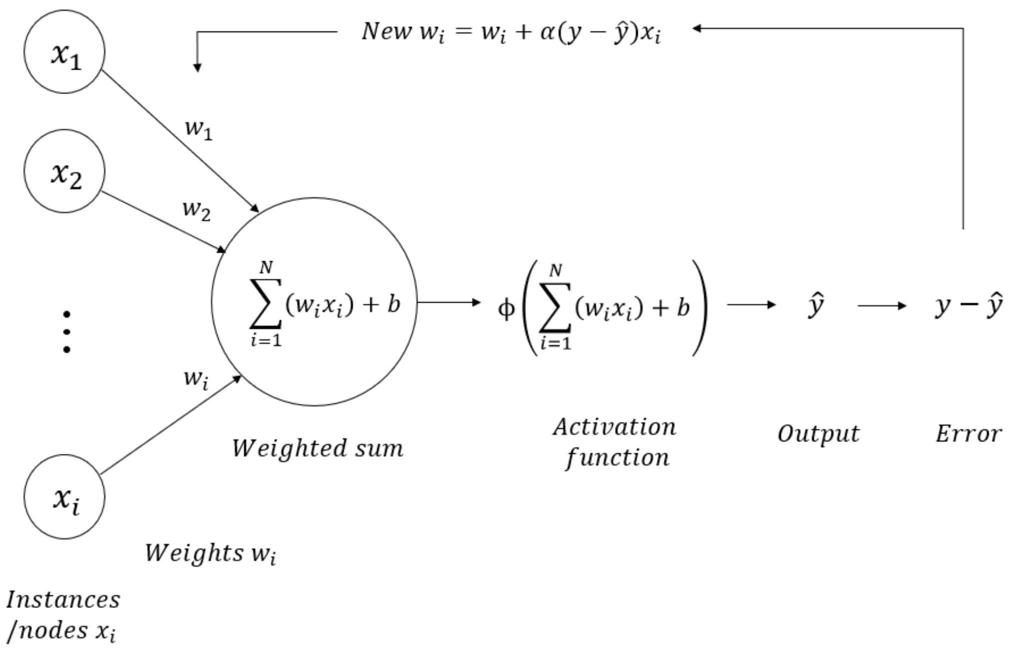

Figure 5: The Rosenblatt Perceptron (Rosenblatt 1958)

This neural network consists of units that connect through links. The links have associated weights w_{ij} and biases b, which transmit information in between the units i and j. The units receive inputs from the environment or other units and a non-linear activation function $\phi(x)$ transforms the received input into an output. The output is then transmitted to other units or the environment. Popular activation functions include the threshold or logistic functions (Du and Swamy 2014):

$$p(y|w,x) = Ber(y|\ sigm(w^T x + \epsilon)) \tag{8}$$

$$Threshold\ function: \phi(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \tag{9}$$

$$Logistic\ function: \phi(x) = \frac{a}{1 + e^{\beta x}} \tag{10}$$

Traditionally, there are three types of units (Quinlan 1998): Units that receive information from the environment, units that receive and transmit information from and to other units, also called hidden units, and units that give the final result, that is, transmit information to the environment.

The training algorithm of a single-layer perceptron consists of the adjustment of the link's weights. While the initial weights are assigned randomly, each iteration slightly modifies their value, according to a learning rate α and estimation error $\hat{e}$. The estimation error is the absolute difference between the output of the neural network and the actual value from the training sample (Du and Swamy 2014).

The training process ends when it reaches a maximal number of iterations or when the estimation error is sufficiently small. With multi-layer perceptrons, this process is called back-propagation.

3. TOOLS AND TECHNOLOGY

One of the internship's responsibilities was the choice of tools to conduct the ML project. Considering that the projects developed during the curricular component of the DSAA-DS Master's program relied majorly on Python and its data science libraries, the tool of choice for collecting, processing, modelling, implementing and monitoring the ML models was also Python.

However, given that the models' training data is stored in MS ACCESS, while the models' results are imported into SQL SERVER, the bridge between these platforms will also be part of this chapter.

The main framework of the project that took place during the internship was Python (van Rossum 1995). The decision to use Python as the tool of choice goes further than the previously explained familiarity with the language and libraries. The internship took place at a financial institution, which includes several security measures and restrictions regarding the installation of external programs. One of the programs that the Data & Analytics team at DAI was already familiar with is Python, namely the data science platform Anaconda (Anaconda Inc. 2020). In addition, the broad package management system conda facilitates access to several essential data science packages, which may not be installed individually on CGD's computers.

The Python libraries that were used during the projects are summarized into the following four steps.

1. Libraries to retrieve data from MS ACCESS
2. Libraries for data preparation, pre-processing and model training
3. Libraries for implementing the results to SQL SERVER
4. Libraries to monitor the models' performance

Library	Description and purpose	Steps
imbalanced-learn	imbalanced-learn is a library that offers tools to deal with imbalanced datasets, such as the SMOTE technique	2
joblib	joblib enables the possibility of saving and loading trained models from and to the environment	2,3
openpyxl	Each time the models are applied to the daily sample of events of the alerts, openpyxl is used to export the percentage of reduced events to an Excel sheet.	4
pandas	All the operations that include data manipulation of any kind were completed with pandas	1,2,3,4
plotly	Plotly (Plotly Technologies Inc. 2015) is an interactive plotting library that offers graphic visualizations, which were essential to the decision-	2,4

	making process in the modelling and validation steps of the project	
pyodbc	The pyodbc library was used to retrieve tables from MS ACCESS, by connecting Python to the database where the alert's data is stored, and using SQL to retrieve the desired information.	1
scikit-learn	Data preparation and pre-processing steps, for instance, the imputation of missing values and the normalization of the variables, were conducted with Python's scikit-learn (Pedregosa et al. 2011), an open-source library that includes a set of predictive data analysis and machine learning tools.	2
sqlalchemy	As the ML models aim to reduce the sample of daily events, the library sqlalchemy was used to export the reduced event sample into a new SQL SERVER table.	3

Table 2: Python libraries used during the internship project

4. PROJECT DISCUSSION

Internal Audit is the metaphorical final gatekeeper of the bank's defense, which makes the prevention of business anomalies and the recognition of risks by the auditors, key. The function of internal audit evaluates corporate governance processes, i.e., risk management and compliance function. Traditionally, auditors work with sampling methods, by extrapolating the results and treating eventual conclusions as issues. However, the real issue of specific problems often lies within the underlying and hidden data. Performing internal audit activities with the aid of business intelligence and data science can help to face these challenges.

The focal project of the internship was the introduction of AI methodologies to DAI, namely the implementation of machine learning pipelines to optimize processes that aid the activity of Internal Audit. The responsibility of implementing the pioneer machine learning pipeline in this area at DAI came with the liberty to autonomously accompany each step of the ML workflow.

The project focused on the deployment of three machine learning pipelines, following a structurally identical workflow. The ML pipelines were applied to the continuous auditing process of DAI, which systematically analyzes operations that stand out from the implemented alarm system.

The alarm system continuously monitors operations in the institution and triggers alerts when an event disobeys a predefined methodology. The auditors review and process the triggering events individually, classifying them either as confirmed errors or false positives. The alerts differ in terms of frequency, precision, and gravity of miss rate. The alerts' triggers generally have low precisions and high frequencies, since they are constructed by sets of manually implemented rules that were designed for a high recall.

Based on data from the historical triggering events and the labels that were assigned by the auditors, three alerts were selected for improvements, based on their high trigger frequencies and low precisions. The project aimed to improve the alerts' triggers by detecting and excluding the false positives before the auditors proceed with the manual classification. The development of binary classification models that maximize the specificity of the triggers, by maintaining their sensitivity as high as possible, aimed to reduce the total number of events that have to be reviewed by the auditors and, hence, significantly maximize the efficiency and productivity of the general alarm system.

Practically, these goals are accomplished by minimizing the trigger's fall-out and maximizing the specificity. Therefore, these two metrics are going to be the focal performance indicators of the models.

4.1. Timeline

The internship at CGD took place for nine months, from the 14th of September 2020 to the 13th of June 2021. The temporal structure of the internship's programs and projects is visualized in Figure 6 (p. 20).

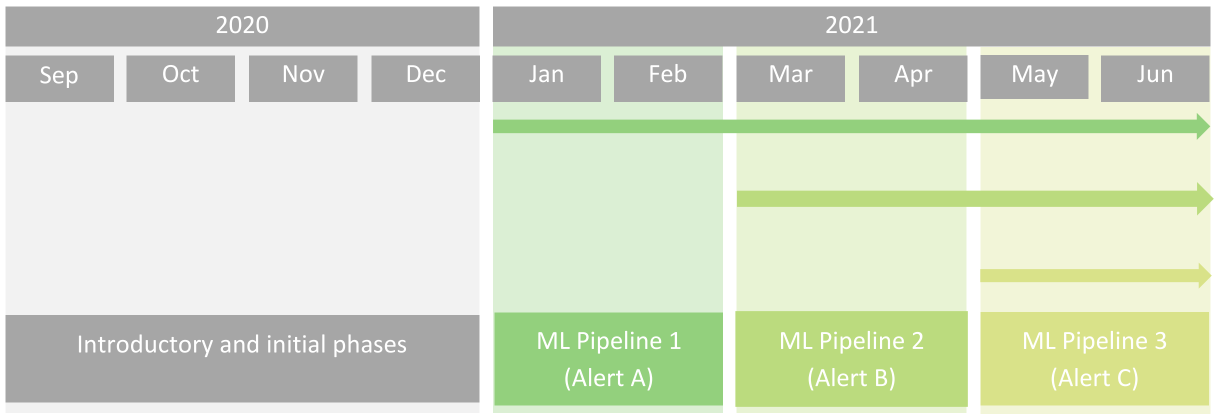

Figure 6: Internship Timeline

The leading project of the internship, namely the implementation of three analogue machine learning pipelines to DAI's continuous audit process, had a total duration of six months, two months for the implementation and monitoring of each ML pipeline. The first four months of the internship focused on the introduction and acclimatization to DAI's information system.

As previously mentioned, the three implemented pipelines follow a similar and equivalent process of data gathering, processing, modelling, model implementation, and monitoring. Due to the non-disclosure agreement that was signed with CGD, this chapter will not focus on the explicit characteristics of the three datasets. However, specific challenges encountered during one or more of the modelling processes will be pointed out, joined by the solution approaches.

4.2. Implementation of Machine Learning Pipelines to DAI's Alarm System

The project that this report focuses on consists of three machine learning pipelines that were implemented to DAI's continuous audit process, namely the alarm system. The three workflows follow the same structure, differing solely in details due to specific characteristics of the alerts' data. The workflows' steps include data gathering and understanding, data preparing and processing, model choice and parameter tuning, model evaluation, implementation, and monitoring.

This chapter will precisely describe the steps that compose the ML pipelines.

4.2.1. Motivation

The continuous audit process' alarm system at DAI comprises about a hundred alerts, which are classified into broad categories, i.e., incorrectly deducted commission charges or diverse types of fraud. The alert's trigger is according to sets of manually implemented rules. Any banking activity that does not fulfil the corresponding restrictions will trigger an alert and be reviewed by an auditor.

The alerts differ in terms of frequency, precision, and gravity of the miss rate. For example, the frequency spectrum of the alerts ranges from multiple times daily, to weekly or even monthly. In addition, fraud-related alerts have lower trigger precisions than alerts from other categories, due to the high cost of the non-retrieval of potentially fraudulent events (miss rate).

The alarm system is designed to identify any conspicuous activity in the institution, which, in statistical terms, means that the alerts focal performance measure is the recall, that is, the percentage of confirmed errors that are detected.

The motivation of this project was the usage of binary classification models to improve the precision of the triggers, by maintaining their recall as high as possible. The models learn from the historical triggering events and the auditors' classifications and aim to improve the specificity of the triggers and maintain their sensitivity. Furthermore, the trigger tuning aims to lead to smaller event samples that have to be reviewed by the auditors and, hence, decrease their workload.

4.2.2. Data Description

Due to the non-disclosure agreement that was signed with CGD, this report will label the three chosen alerts as **Alert A**, **Alert B,** and **Alert C**.

The alerts' datasets are composed of the historical triggering events and the corresponding classifications by the auditors. The auditors' classifications of the events are the variable that this project aims to predict, hence, the output of the binary classification models.

The choice of the three alerts that were tuned was based on the three following factors, ordered by relevance:

1. Trigger frequency/dataset size
2. The gravity of the miss rate
3. Trigger precision/specificity improvement potential

The trigger frequency and precision are negatively correlated, which leads to a forced trade-off between dataset size and target class imbalance. The target class imbalance is a direct consequence of low trigger precision, due to the significant prevalence of false positives in the target data. Knowing that the ideal conditions for the performance of ML models are generally significant and balanced datasets (Fernández et al. 2018), these optimal conditions contradict themselves by the dataset characteristics mentioned above.

The three chosen alerts are the leaders in trigger frequency and have an overall small risk for non-retrieved positive instances. Alert A has, by far, the most extensive triggering event history, and hence the most information to be used in the model training process. Alert B has the second-largest triggering event history, and Alert C was chosen due to a recent high activity, which is an indicator of great potential for improvement.

Regarding the events' date ranges, the datasets' first records start on 02-04-2018.

Dataset structures

The events that trigger the alerts are first temporarily stored in SQL views and afterwards imported to a platform that is available to the auditors. The auditors' platform contains the historical triggering events and the classifications from the auditors, which compose the datasets that are used for model training. The SQL views contain small samples of recent events that have yet to be imported to the platform. The classification models are applied to the events from the SQL views, to reduce the sample of events that is imported to the auditors' platform.

However, not all variables from the SQL views are imported into the auditors' platform. Considering that the models are trained with the data from the auditors' platform, however, applied to the events in the SQL views, the data structures have to be adjusted accordingly. This adjustment includes the unification of column selection, column order, and the data types that vary between these two platforms.

The dataset structures for each alert and platform can be consulted in Table 3.

Alert	**Number of events auditors' platform**	**Mutual columns**	**Exclusively on the auditors' platform**	**Exclusively in the SQL view**
A	~ 11300	22	30	29
C	~ 3300	25	30	24
B	~ 3300	34	30	13

Table 3: Data structures of the alerts' datasets, both in the auditors' platform and SQL views (as of August 2021)

The equalization of the data structures will be described in the subchapter about data integration.

Data types

Figure 7 depicts the data types of the alerts' datasets from the auditors' platform. The category 'Other' includes identification columns, alphabetic data such as comments and specific descriptions, or serial numbers composed of several variables.

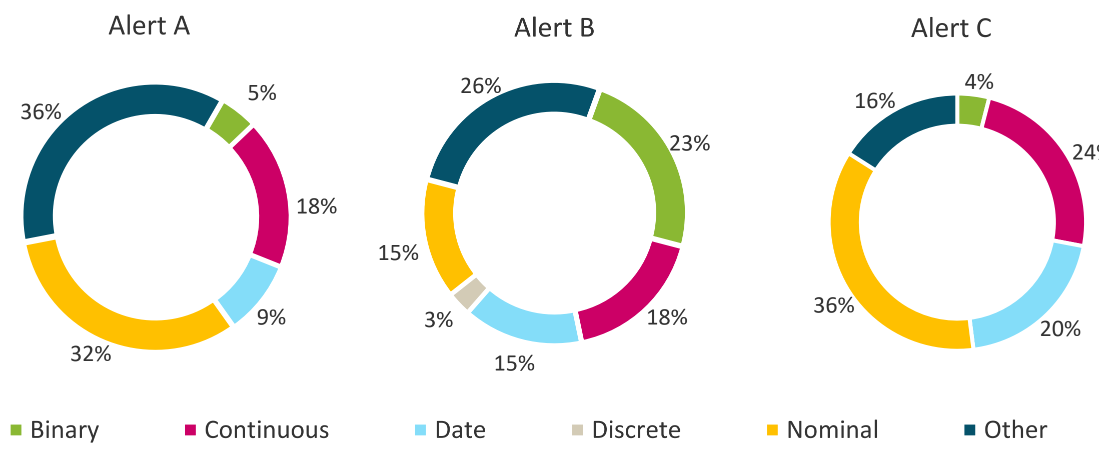

Figure 7: Data type percentages of the variables from Alerts A, B and C. The data is retrieved from the auditors' platform, where the data from the triggering events is stored.

Target distribution

The variable that the classification models aim to predict is the auditors' classification of the triggering events. The events can either be classified as confirmed events or as false positives. The implemented ML models work as retrieval systems, where the confirmed events are labelled as the positive class, and the false positives are labelled as the negative class.

An initial analysis of the target variable revealed that the alerts' triggers have low precisions, meaning that the majority of the events are closed as false positives. This property leads to skewed target class distributions. The ratios of the positive class vs. negative class of these imbalanced datasets are shown

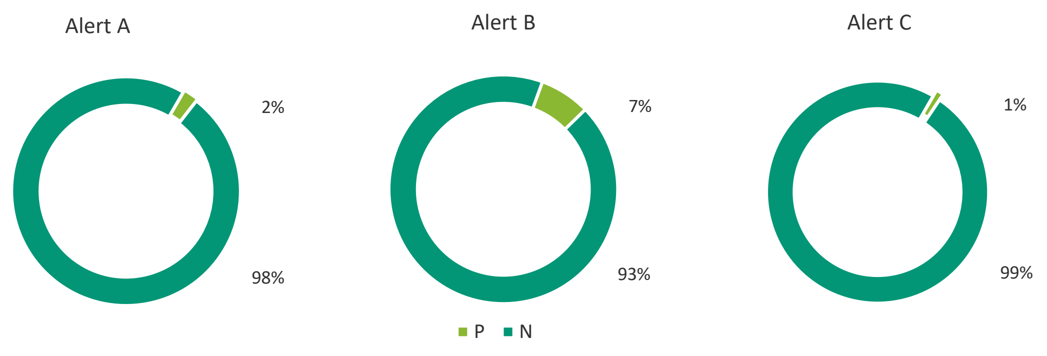

Figure 8: Distribution of the target variables of Alerts A, B and C. P stands for the positive class, i.e., confirmed errors, whereas N stands for the negative class, i.e., false positives.

The target class imbalances offer insights into the triggers' precisions, which are 2% for Alert A, 7% for Alert B, and 1% for Alert C. The models aim to maximize the precision of the triggers by maintaining their recall as high as possible, to reduce the dispensable workload of the auditors. In practical terms, these goals are accomplished by minimizing the models' fall-out and maximizing the models' specificity. Therefore, these two metrics are going to be the focal performance indicators of the models.

4.2.2.1. Data preparation and preprocessing

Before the data preparation and preprocessing steps that are described in the following sectors, the data from the auditors' platform was split into training and test sets. The model training was completed with 85% of the data, while 15% of the data was used to provide a final performance estimation of the models on new information. There was no need to define a validation set, as the cross-validation technique was used to tune the hyperparameters of the models.

The data from the SQL views was also used as an unlabeled test set, given that the events from this source are not yet classified. The only metric that can be used to estimate the model performance on this data sample, when compared to the results obtained by cross-validation, is the event exclusion rate of the models. This comparison provides an estimation of the model performance on recent events.

Data Integration

As shown in Table 3 (p. 22), the alerts' data from the auditors' platform includes 30 variables that are not part of the SQL views. Considering that the models are applied to the daily events from the SQL views, these columns are not of further use.

The columns that exist exclusively in the SQL views and are not imported to the auditors' platform must also be excluded, since the SQL views delete the event history after a specific time, and the information is not saved elsewhere.

Furthermore, several mutual columns have different data types in the auditors' platform and SQL views. For instance, some numerical columns in the auditors' platform are imported as objects to SQL. Therefore, the affected columns from the SQL views were converted to numerical data like in the auditors' platform, to unify these contrasting datatypes.

Data Cleaning

As visualized by Figure 7 (p. 22), the majority of the alerts' variables are qualitative. For the qualitative variables, the missing values were imputed with the category 'Other'. Since the datasets were created by automated processes, the eventuality of data errors is improbable. This type of imputation has the advantage of not introducing potentially incorrect information to the dataset, in opposition to the KNN Imputer or the imputation of missing values with the mode of the data. Some missing values occur in variables containing 'optional' information, such as agreements between the institution and an entity. In this case, missing values merely indicate that such an agreement does not exist. Hence, a binarization with the missing values as the negative class is the most adequate imputation method.

The quantitative variables in the alerts' datasets mainly refer to monetary values, which limits their values to the numerical set Q_+. The quantitative variables' distributions are strongly right-skewed, and the variables include missing data and abnormally high values. Hence, a log transformation was used to remove the original skewness of the data, which allowed for easier identification of potential data anomalies. The missing values in these quantitative variables can be interpreted as 0. Thus, missing values and values below the unit were imputed with the number 1, translating to a 0 after the log transformation.

To exclude data anomalies, the *z-scores* were computed to detect potential outliers, using a threshold of three standard deviations. This method eliminated roughly 0.1% of the data. An example of these data cleaning operations can be seen in Figure 9, which represents the transformations mentioned above for a continuous variable from the dataset corresponding to Alert A.

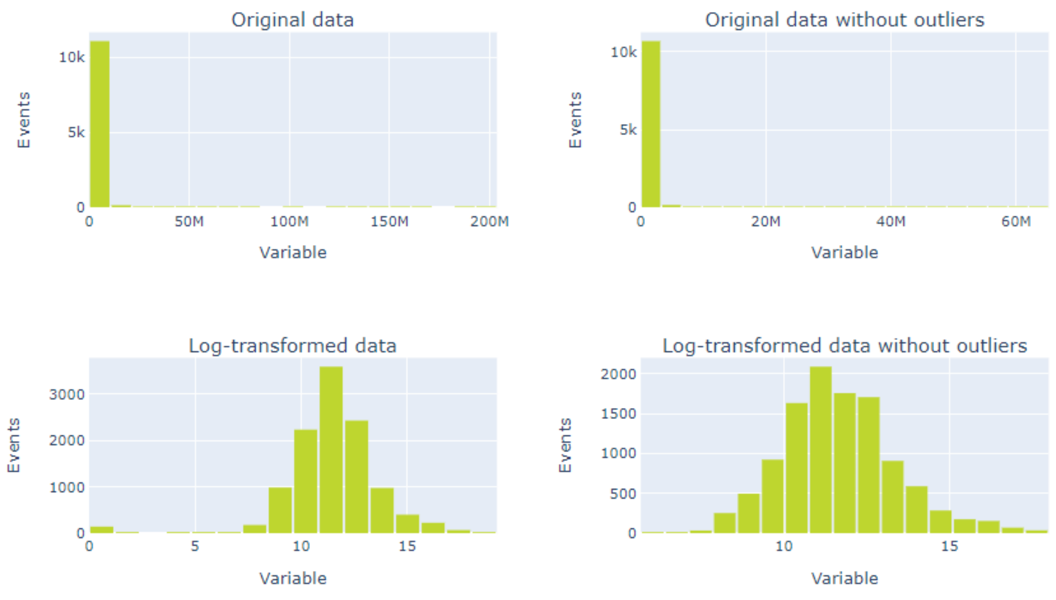

Figure 9: Histograms of a continuous variable from the dataset corresponding to Alert A. (1) The data in its original distribution before outlier removal. (2) The data in its original distribution after outlier removal. (3) The data after a log transformation. (4) The data after a log transformation and outlier removal. (From left to right, top to bottom)

One of the project's main challenges was the shifting distribution of the nominal variables' categories. From this point on, this issue will be referred to as data dynamism.

In addition to the challenge of relatively small datasets, some events in the datasets did not contribute much information to the classification of new events, because the categories of the qualitative data are under constant change. For instance, the first economic effects of the SARS-CoV-19 pandemic in 2020 shifted the class distributions of certain nominal variables and were responsible for entirely new sets of categories.

The effects of the data dynamism came to light when a trained classification model was applied to the events from the auditors' platform (with cross validation), and to the unlabeled test set with events from the SQL views. The event exclusion rates were significantly different, and one possible explanation was that the data dynamism was the main cause for the mentioned model performance discrepancy. When the data from the auditors' platform is split into training and validations sets during cross validation, these sets are formed arbitrarily, not following a chronological order like the recent events from the SQL views. Since the models are intended to be applied to exclusively recent events, the non-occurring patterns that the models would learn to generalize are often not of benefit.

Figure 10 depicts the distribution of the ten most common categories of a nominal variable from the dataset corresponding to Alert A, for the last three years. The impact of the SARS-CoV-19 pandemic can easily be identified as an example of the changing category distributions, as can be verified with the number of triggered events and category occurrences.

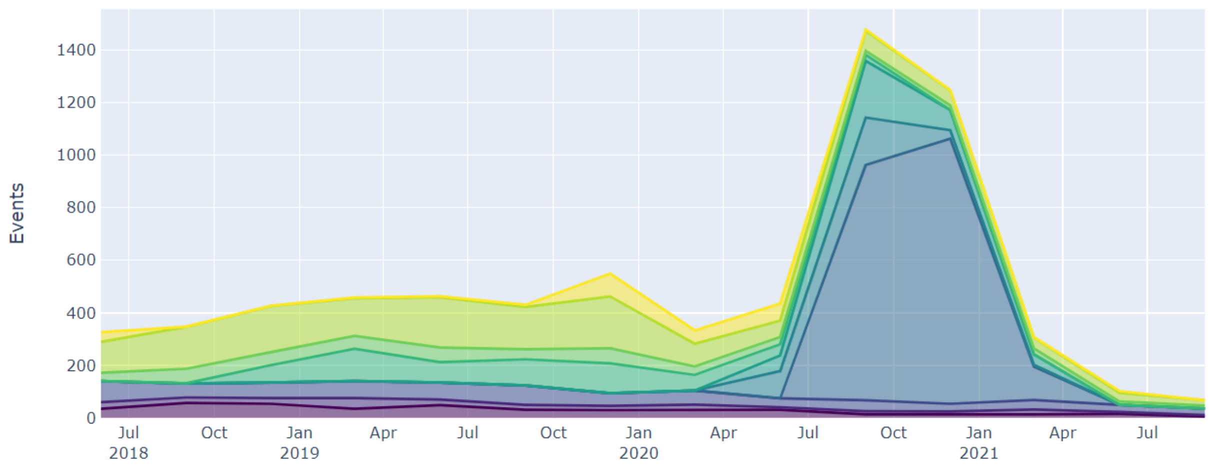

Figure 10: The category distributions of the 10 most common categories of a nominal variable from the dataset corresponding to Alert A, from July 2018 to July 2021. The category legend is excluded from the figure to not disclose concrete information.

One method to overcome this challenge was a data selection process, which ensures that only relevant information is used to train the models. This process of data selection was conducted by trial and error and followed the following two approaches:

1. **The creation of event recency intervals**
   This approach is based on model performance comparisons between models that were trained with fixed event recency intervals. Datasets with events from the last 18, 12, 9, and 6 months were created and used to train classification models as a means to compare the corresponding performances.
   However, the target labels of the events are not distributed uniformly along with the dataset's time windows. Thus, this approach had the disadvantage of unbalancing the dataset further, making the modelling process more challenging. To overcome this disadvantage, the data range restriction was adjusted to merely affect events from the negative class, keeping all positively labelled instances. The resulting measure counteracted the disadvantageous behaviors while solving the data dynamism and the skewed target class data.
2. **The exclusion of events with non-recent category values**
   This approach analyzed the events and their categories of nominal variables individually. Events that did not include any 'recent' categories were excluded from the dataset. The recency interval and the choice of which nominal variables to analyze are customizable and manually adjusted for each alert. Analogically to the first approach, the performance comparison of classification models trained with the resulting datasets revealed the most effective recency interval.

The data dynamism did not occur in every alert. Hence, this data cleaning step had to be concluded optionally for each alert, since the alerts' behaviors are unpredictable and do not follow a specific pattern.

Data Transformation

As depictured in Figure 7 (p. 22), most variables of the event's datasets are nominal. Since some machine learning models, such as the logistic regression classifier, can only interpret numerical data, the nominal variables were transformed into binary features corresponding to each category.

Furthermore, the creation of new features reinforced the extraction of relevant information from the quantitative variables. For instance, if incorrectly deducted commissions trigger an alert, ordinal features were added beyond the expected and actual deduction value. These features described the difference between the expected and actual deduction values as lower, equal, or higher.

Regarding the timestamp and date variables, the extraction of ordinal features also added new information to the datasets, for instance, the definition of the quarters of the year. In addition, for the alerts that include variables of 'start' and 'finish' dates regarding deposits, ordinal duration features were calculated and included in the datasets.

The sets of features labelled as 'Other' in Figure 7 (p. 22) were mainly excluded from the dataset. However, some variables, such as serial numbers constructed as combinations of possibly important variables, were separated into distinct features and encoded as nominal, hence subsequently, binary variables.

In response to the target class imbalance mentioned earlier in this chapter, the SMOTE oversampling algorithm (Douzas et al. 2019) was used to generate synthetic samples for the minority class. The altered dataset was saved separately, as eventual performance improvements are only coming to light during the project's modelling stage.

Data Normalization

Quantitative variables that are measured at different scales do not contribute equally to a model fitting process, which can induce a bias. Therefore, the quantitative variables that were already subjected to the log transformation were squished between 0 and 1, matching the binary variables' range. This transformation was possible due to the previous removal of outliers and the log transformation, which assimilated the quantitative variable's distribution to a normal distribution.

Feature selection

As Table 3 (p. 22) indicates, the variables that do not exist mutually in the auditors' database and the SQL views could not be used for the modelling process. Therefore, with this initial variable selection and the feature engineering described in the subsection above, the datasets were left with mainly binary features and less than a handful of quantitative features.

Due to the great diversity of categories in the original nominal variables, the number of new binary features ranges over a thousand. However, low occurring categories most commonly do not result in significant binary features.

Considering that the problem at hand is a classification predictive modelling problem with mostly categorical input variables, the chosen feature selection method is the chi-squared test. The chi-squared test tested the significance of each binary feature individually, and the resulting *p-values* functioned as exclusion criteria to exclude features that did not show significant dependency with the target variable. The usual levels of significance of 1%, 5%, and 10% were used and compared as decision thresholds to determine which set of features resulted in the best model performance. Additionally, fixed-size sets of most significant features, such as 25, 50, 75, and 100 were also constructed and used for model comparison.

The ANOVA test evaluated the significance of the quantitative features, testing whether the continuous variables and the target variables origin from the same distribution. The result indicated that all the quantitative variables were statistically significant and could be kept in the feature space.

4.2.3. Model and Analysis

Several machine learning algorithms for binary classification were applied to the processed data, and the resulting performances were compared. The comparison process for each algorithm followed two main steps: the dataset and algorithm choice.

Dataset choice

As the previous chapter about data pre-processing and feature engineering/selection explored, several different dataset variations were created and considered for the modelling process. These dataset variations included different approaches to overcome the data dynamism, decide about feature selection, and eventually apply an oversampling technique to overcome the target class imbalance. Finally, the different dataset variations were applied to each ML algorithm to compare the performances of the resulting models.

Regarding the oversampling technique, the three alerts followed a similar behavior. The synthetically altered datasets class did not result in better model performances when compared to the original dataset.

As the subsection about applying the chi-squared test mentioned, the feature selection approaches showed different results for each alert. The choice of a fixed number of the most significant features by the chi-squared test resulted in higher performance indicators for Alerts A and C. The models of these two alerts were trained with the 50 and 100 most significant features, respectively. However, the model corresponding to Alert B performed better when trained with the features classified as significant by the chi-squared test, according to a *p-value* of 5%.

Regarding the data dynamism, not all alerts' datasets needed intervention. However, the model for Alert A showed a better performance when the training data was altered with an event recency interval. The applied interval excluded false positives that triggered more than nine months ago. This alteration resulted in a lower performance discrepancy between the validation and test sets.

Choice of algorithm and hyperparameter tuning

When choosing the best algorithm for a dataset, the dataset's properties and characteristics must be considered. ML classifiers and estimators reach their maximum potential and optimal performance when trained and applied to datasets with the most advantageous properties, which can vary between the classifiers.

The optimal ML classifiers for the alerts' datasets are ideally relatively robust to the following three key dataset characteristics and challenges:

1. Relatively small sample size
2. Strongly skewed class distribution
3. Substantial prevalence of binary features

These peculiar dataset properties narrow down the pool of suitable ML algorithms. Furthermore, experiments with different algorithms have shown that the dataset size significantly impacted the performance of classification models. For instance, complex models such as neural networks generally perform better on large datasets (Althnian et al. 2021), and the K-Nearest-Neighbor and Support Vector Machines tend to achieve good results with imbalanced datasets (Sun, Wong, and Kamel 2009).

Out of the ML algorithms described in chapter 2.3., the best performances were obtained by the Logistic Regression classifier, followed by the Decision Tree classifier.

The hyperparameter tuning was conducted by an exhaustive search over specified parameter values for the estimators (Pedregosa et al. 2011). Additionally, a cross-validated grid search over a parameter grid indicated the optimal values of the parameters.

Considering that the goal of the ML models was to optimize the precision of the triggers while maintaining their recall, and due to the target class imbalance, which turns accuracy scores unreliable, the grid-search was performed with the F_β-score as its evaluation function. The value of β was defined proportionally to the target class imbalance of each alert, to create a balanced environment between the target class frequencies and the weights of each class in the modelling process.

The logistic regression and decision tree classifiers each have a parameter to weight the target classes according to the class distribution, called *class_weight* in Python's machine learning library *scikit-learn*. Due to the target class imbalance mentioned several times throughout this report, this parameter was found to be the most influential in the model's performances, as it penalizes the incorrect classification of a positively labelled instance according to the chosen *class_weight* value. A high value of this parameter solely improved the recall of the models, whereas a lower value benefited only the precision.

Figure 11 graphically showcases the recall and specificity in function of the parameter *class_weight* for the logistic regression classifier. The dataset that was used to train the models for this visualization corresponds to Alert A. In this case, the restriction of an event recency interval of 9 months for negatively labelled instances smoothed the target class imbalance.

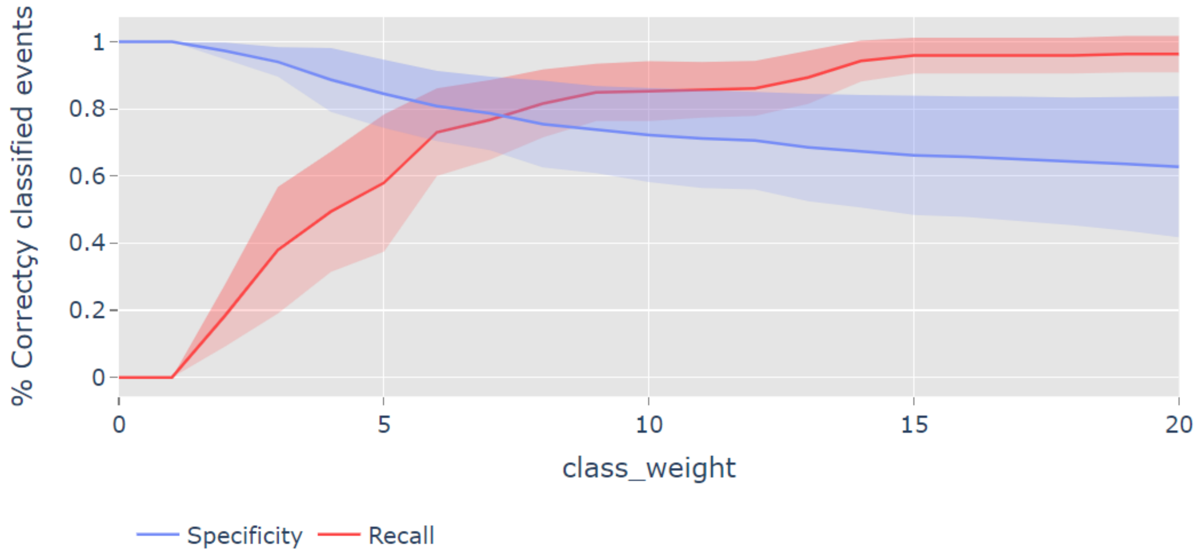

Figure 11: Recall / sensitivity (red) and specificity (blue) of the logistic regression classifier fitted to Alert A, in function of the parameter *class_weight,* ranging in the interval of [0,20].

The optimal value of the *class_weight* parameter depends on the intended output of the model. For instance, if an alert triggers multiple dozen times a day and the cost of a high miss rate is not alarmingly high, which is the case for Alert A, a significant improvement in trigger precision can be beneficial, even if the recall is slightly lower than initially.

The values of recall and specificity in function of the parameter *class_weight* are negatively correlated. Finally, the threshold for the minimum required recall for each alert was set at 90%, taking into account the mean minus the standard deviation. In Figure 11, this threshold is obtained at a value of *class_weight* of 15.

The chosen models for each alert followed the rule of a minimal recall value of 90%, taking into account the mean minus the standard deviation of the results from the cross-validation. The chosen values of the *class_weight* parameter for each alert highly depended on the skewness of the target class distribution.

Model evaluation

The cross-validation technique was the fundament of the hyperparameter tuning and dataset choice for the classifiers. The results presented a relatively robust estimation of the model performances on the validation sets, which were composed of data that had been pre-processed together with the training data, but not used for model training. Furthermore, this method allowed for an estimation of the under- or overfitting of the models.

However, as the models intended to be applied to entirely unseen events, 15% of the events for each alert's datasets were separated before the data processing and modelling steps, to construct an estimation of model performances on previously completely unseen data. This test set was pre-processed analogically to the training data preparation, using the encoder instances that were fitted with the training sets.

After model training, the test set was used to obtain a final model performance estimation of the classification of events that were retrieved from the same distribution as the training set. However, the data dynamism challenge that was described earlier includes the risk of new events not following the data distributions as the events in the training, validation, and test sets. This eventuality is severe because the project's focal point is the correct classification of the new events that trigger the alerts daily.

The only possibility of creating a model performance estimation for the classification of recent events is the application of the models on the events from the SQL views. The events from this data source are recent and not classified yet. Thus, the only model performance indicator is the percentage of event decrease that is obtained by the model application.

The comparison of the event elimination rates from the models' applications on the SQL views, and the specificity of the models that was obtained with cross-validation and the test sets, was a good performance indicator. A coherent and similar rate indicates that the new events' datasets approximately follow similar data distributions as the less recent events' data, that is, the data that is used for model training. This conclusion is essential for a promising and effective model performance on new events.

Table 4 indicates the recall and specificity of the models on the validation and test sets and the event reduction rates from model application to the sets from the SQL views. For comparison purposes, the models that originated these results were fitted to have a mean recall of 95%.

Alert	**Validation set**		**Test set**		**SQL Views**
	Recall	Specificity	Recall	Specificity	Event reduction
A	0.956 +/- 0.042	0.676 +/- 0.035	0.973	0.635	0.671
B	0.938 +/- 0.042	0.278 +/- 0.016	0.948	0.274	0.254
C	0.968 +/- 0.067	0.377 +/- 0.059	0.909	0.385	0.449

Table 4: Comparison between the values of the recall and specificity of the logistic regression models for Alerts A, B and C, on the validation (obtained by cross-validation) and test sets, both extracted from the auditors' platform, and samples from the SQL views

Neither of the alerts indicated signs of overfitting, since the performance indicators were similar for the validation and test sets. Additionally, the event elimination rate from the model application on the SQL views was coherent with the expected results, leading to the assumption that the underlying risk was also equal to the estimations. However, the model performance of Alert B was severely lower than for the other two alerts, indicated by lower recall and specificity values.

Model implementation and motorization

As depicted in Figure 12, the models are trained and updated with new events daily. This daily update ensures that the data distributions between the training and new events are as similar as possible.

The correct functioning and performance of the models are monitored by the exportation of the daily event exclusion rate, i.e., the percentage of detected false positives. In the event of anomalies concerning the model performance expectations, the models are re-evaluated.

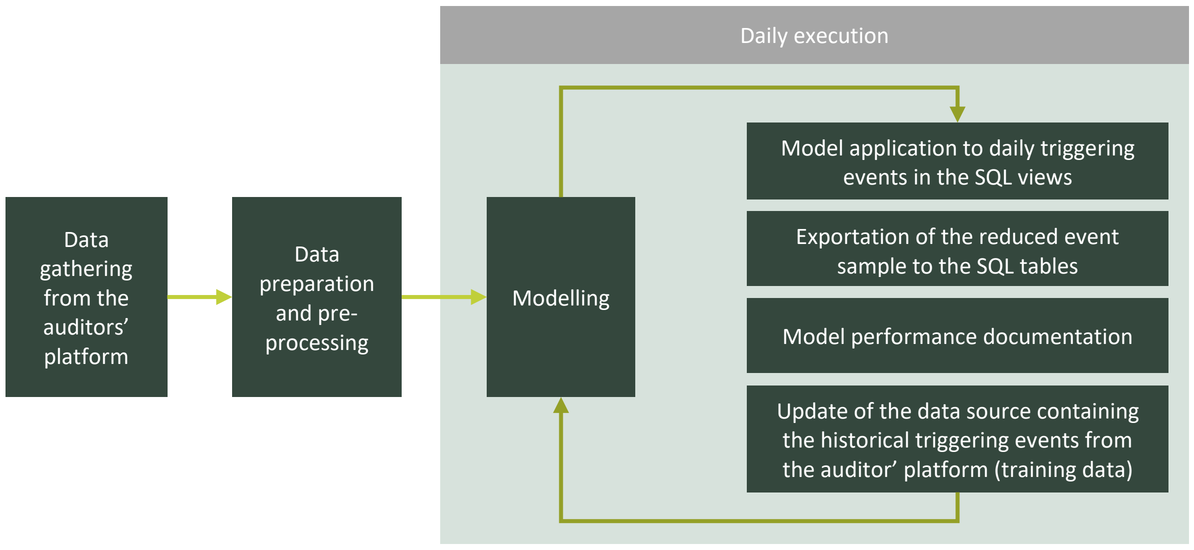
Daily execution
Data gathering from the auditors' platform
Data preparation and pre-processing
Modelling
Model application to daily triggering events in the SQL views
Exportation of the reduced event sample to the SQL tables
Model performance documentation
Update of the data source containing the historical triggering events from the auditor' platform (training data)

Figure 12: The workflow of the model implementation and monitoring. The first two steps (left to right) were completed once, whereas the area that is shadowed in light green is executed daily, in the depictured order.

4.2.4. Results and Discussion

As previously mentioned, the *class_weight* parameter of the logistic regression classifier was the most influential component in the models' outcomes. Hence, three risk thresholds were used as guidelines for each alert to choose the optimal value for this parameter. The risk thresholds referred to the minimal required recall expected from the models and were set at around 95%, 90%, and 80%.

The model performance estimations were obtained with cross-validation, and both the mean of standard deviations of the recall and specificity were taken into account. The models' performances in terms of recall and specificity can be consulted in Table 5, the highlighted cells of which indicate the final chosen *class_weight* parameter for each model. The choice of evaluation metrics is linked to the factors that were listed in chapter 4.2.2.

In practical terms, the specificity indicates the reduction of the trigger frequency, and the recall describes the security of the model.

Alert	Recall	Specificity	Class_weight
A	0.970 +/- 0.034	0.672 +/- 0.032	20
	0.956 +/- 0.042	0.676 +/- 0.035	15
	0.890 +/- 0.066	0.722+/- 0.030	10
B	0.951 +/- 0.032	0.236 +/- 0.022	40
	0.938 +/- 0.042	0.287 +/- 0.016	35
	0.833 +/- 0.074	0.521 +/- 0.034	25
C	0.968 +/- 0.067	0.377 +/- 0.059	80
	0.945 +/- 0.071	0.462 +/- 0.080	60
	0.912 +/- 0.107	0.526 +/- 0.076	50

Table 5: Values of the mean and standard deviation of the recall and specificity of the logistic regression models for Alerts A, B and C, in function of the *class_weight* parameter. The cells highlighted in green indicate the models that were chosen to be implemented.

The model choice for each alert depended on the individually assessed compromise between risk and benefit. Both the recall and specificity were severely higher for Alert A when compared to the other alerts. This difference can be explained by the lower target class imbalance and the larger data volumes used to train the model. Alert A triggers more frequently than the other two alerts.

Figure 13 graphically visualizes an estimation of the alerts' model performances, based on the data from the year 2020. The saturated colors indicate the events that were kept by the models, whereas the transparent areas represent excluded events for each class.

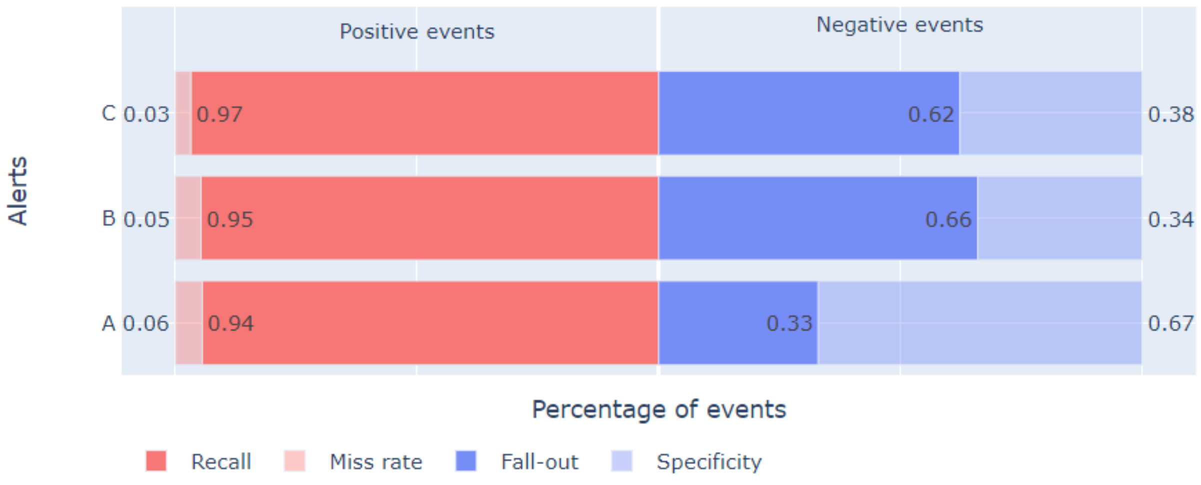

Figure 13: The results of the implemented logistic regression models for Alerts A, B, and C, based on the alerts' data from 2020. The saturated red and blue colors indicate the percentages of events from the positive and negative classes that were kept by the models, whereas the transparent areas visualize the percentages of excluded events.

The trade-off between risk and decreased workload in function of the *class_weight* parameter for Alert A is depicted in Figure 14. The decreased workload estimation is precisely proportional to the percentage of decreased events in Table 5 (p. 32).

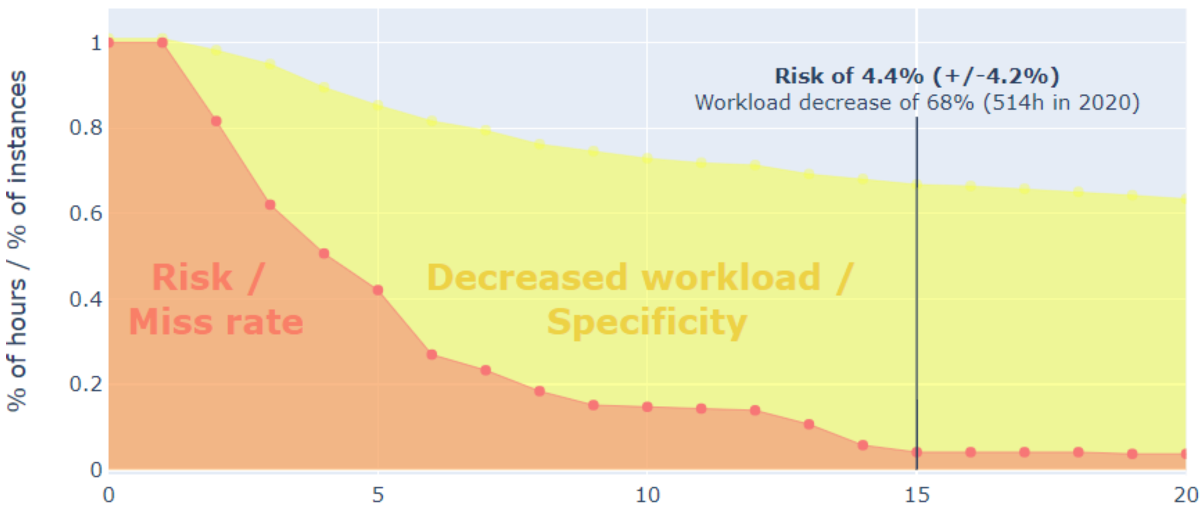

Figure 14: Risk / miss rate vs. decreased workload / specificity of a logistic regression model for Alert A, in function of the *class_weight* parameter. The implemented model is indicated with the black vertical line.

In this case, the value of *class_weight* = 15 satisfies the recall requirement of 95.6% (+/- 4.4%) while reducing the auditor's workload by about 68%.

A simulation on Alert A's data from 2020 revealed that this decrease would approximately correspond to 514 hours of event treatment, assuming that the treatment of an event takes about ten minutes.

4.2.5. Conclusions

Under the assumption that the original trigger recall is equal to 1, the original triggers' fall-outs were 98%, 93%, and 99% for Alerts A, B, and C. After the implementation of the models, the fall-out of Alert A's trigger was improved from 98% to 33%, corresponding to an improvement of 65%. The trigger fall-outs for Alert B and C were improved by 28% and 37%, respectively. This improvement came at the cost of a recall reduction of about 4%, 5%, and 3% for Alerts A, B, and C. Hence, the goal of maintaining the recall as high as possible by minimizing the fall-out and maximizing the precision of the triggers was fulfilled, especially for Alert A. As expected, the model created for Alert A delivered the best results, due to easier conditions to construct the models, such as the three times larger sample size and a ten times lower target class imbalance.

In retrospect, it was interesting to compare the model outcomes for the three alerts due to the similarities and differences between the event's datasets. The similarities included a high target class imbalance, a prevalence of nominal and binary variables, and overall small sample sizes.

Unforeseen challenges, such as the changing category frequencies of the nominal data, were detected and addressed manually. Regarding the data dynamism, it was interesting to conclude that not every alert's data needed intervention. Solely Alert A's data showed a significantly better model performance when the training data was altered with an event recency interval. The exclusion of false positives that triggered more than nine months ago resulted in a lower performance discrepancy between the validation and test sets. Hence, this operation ensured that the model was more robust to new information, and lowered the risk of overfitting.

The mutual characteristics between the alerts gravitated towards similar modelling results, such as the choice of the best classification algorithm: the logistic regression classifier. Thus, the parameter responsible for the target classes’ weights in the modelling process had the highest impact on the compromise between recall and precision and ended up being the main parameter to tune. Furthermore, the flexibility of this parameter could be used to adjust the models to different risk scenarios, which reflect the cost of miss rate for each alert.

Due to the data dynamism of the alerts, it was decided that the models are trained daily with the newly classified events, which are imported daily from the SQL views to the auditors’ platform. This periodic model update ensures that the models are constantly up to date with new characteristics in the data. However, due to the recent activity of the models, the possible effects of the data dynamism issue may not be noticeable yet, which asks for special attention to the future performance of the models.

Finally, a feature importance measure obtained with the random forest algorithm offered an interesting insight into the most decisive variables/categories that yielded the most mutual information with the target variable and potential weight in the model's decision process. Furthermore, the gained information could be used to directly tune the manually implemented rules in the SQL views where the triggering events are saved, to exclude certain events from entering the database directly.

5. CONCLUSIONS

Every data science project derives from a notion of improvement potential, and the incentive to create or optimize an existing system. The project that was described in this internship report is no exception and rose from this exact motivation. In the project's initial phase, namely the evaluation of the business problem and its enhancement perspective, it was challenging to define and pinpoint concrete targets. However, the explicit goals followed along as the project guidelines were set and the data potential was brought to light.

The project's beginning was characterized by the uncertainty of how the alarm system at DAI could be improved. The analysis of the alarm system revealed each component's properties and characteristics, ultimately leading to the identification of alerts that yielded the most potential to be tuned. The alerts were interpreted as retrieval systems, which put the idea in motion to implement classification models to improve the precision of alerts' triggers.

The classification model for Alert A was the pioneer ML application to DAI. Fortunately, its successful implementation led to the decision to apply the same procedures to Alerts B and C, which also triggered frequently and not precisely. Due to the alerts' similarities, the ML pipeline that was built for Alert A could easily be adapted to Alerts B, C, and other future projects.

Even though individual data processing steps always have to be completed manually, the implementation process of new alerts was greatly facilitated. In addition, the ML pipelines decreased the auditor's workload at a reasonable and predefined risk of not retrieving correctly triggered events.

Despite the similarities between the three datasets, each ML pipeline brought on new challenges, which inspired distinct approaches. The principal challenges that were faced during the implementation processes were the choices between different data preprocessing and preparation alternatives. For instance, the data dynamism issue and the changing distributions of certain features demanded creative and flexible solutions.

In conclusion, the implementation of the three ML workflows resulted in the successful optimization of DAI's alarm system. The first ML pipeline for Alert A was implemented successfully, and the favorable outcome of the project led to the acknowledgement of ML applications on the other two alerts. Furthermore, the ML models were sequentially implemented, and the results have been meeting the model performance expectations and estimations, eliminating an average of 45% of daily events that trigger the three chosen alerts. It is essential to mention that no alerts with a high cost of miss rate, such as fraud-related alerts, were chosen to be tuned. The trade-off between precision and recall of the models induced an existent risk to the project's implementation.

The primary goal of decreasing the auditor's workload was completed, as implementing the models for the three alerts resulted in a significant reduction of daily events.

5.1. Connection to the master program

As the name suggests, the master's program in DSAA-DS specializes in information technology and computer science for data science. The technical aspect of the program is very up-to-date to the global market, and the students acquire broad expertise in the vast majority of areas that compose the data science universe.

From the beginning of the internship at CGD, my personal goal was to participate in a project that involved a data processing and modelling process, as I wanted to deepen my newly acquired skills and, at the same time, experience the differences between academic projects and professional challenges. Furthermore, without the technical background and knowledge that I acquired during the masters' program, the ML pipelines would not have been implemented in DAI's alarm system. Hence, my initial personal goal was undoubtedly achieved.

The master's program, especially the data mining and machine learning courses, strongly focus on model construction and evaluation. Therefore, the skills and takeaways of the academic projects that were developed during those courses were directly applied to the internship projects.

For instance, the correct structure of a machine learning pipeline, the individual steps of the CRISP-DM model, and the final critical and objective appraisement of ML models' performances were all meticulously discussed in the course and applied to the internship's projects. Furthermore, the mathematical understanding of the models' mechanisms firmly guided the comparison between different classification algorithms. This knowledge was especially essential for the hyperparameter tuning of the models, and decision-making in the data processing stage, since each algorithm has different ideal data conditions.

One of the main advantages of having been part of the pioneer ML project at DAI was the freedom of tool choice and implementation approach within the institutions' regulations. Thus, the knowledge I acquired during the curricular year of the master program served as the leading guideline to the project. Since Python was the most frequently used language during the DSAA-DS courses, defining Python as the tool of choice was instant. The introduction to the data science platform *Anaconda* in the masters' program was especially relevant, as this granted a fluency in Python's data science libraries. Additionally, understanding the manipulation of SQL objects was essential to the internship's projects and was possible due to the master's program.

Finally, on a more general note, the evaluation methods of some DSAA-DS courses, such as the performance in Kaggle competitions, strongly stimulated the desire to obtain the best results and learn endlessly and consistently. This eagerness and enthusiasm to constantly improve one's performance was the guiding force to implementing the ML pipelines and is the key to excellence and a successful career start.

5.2. Internship evaluation

The internship with the Data & Analytics Team at DAI was an overall positive experience. It was a privilege to individually implement and accompany an ML project from start to finish. All the steps of a classical machine learning pipeline, from the business understanding and data gathering to model implementation and monitoring, were part of my responsibilities. Whereas the master program focused mainly on an ML pipeline's data processing and modelling steps, monitoring the practical model implementation and continuous evaluation steps was equally fascinating.

The freedom of methodology and tool choice came with the responsibility to thoroughly investigate all options and make the best decisions for the business problem in question. The development of this internship report as my master's thesis undoubtedly helped with the choices that were made during the project development, due to the large number of relevant references that were consulted to undermine the decisions.

Furthermore, considering that this internship was my first professional experience, I found it important to humbly and carefully evaluate each possibility for the decisive choices that had to be made along with the project. Thus, I realized that the possibilities in ML are endless and that I still have a lot to learn and improve.

Before I started this internship, I was eager to apply my newly acquired data science knowledge. This goal was accomplished, as I had the chance to experience how machine learning projects are implemented in a professional environment.

5.3. Limitations

Considering that the internship was completed at a financial institution, the security restrictions regarding the usage of specific tools can be regarded as an obstacle. For instance, specific Python libraries that are not part of the *Anaconda* data science platform could not be installed, and some steps of the project, especially the visualization of the results, had to be completed alternatively. Another difficulty that was encountered due to the security restrictions at CGD was the connection between MS Access and Python, which was used to import the events from the auditor's database.

An obstacle encountered on the path to further apply machine learning models to other alerts that compose DAI's alarm system is the substantial class imbalance of the alert's data. In most cases, the petite sample sizes of confirmed events do not yield enough information to use ML to boost the trigger precisions. Additionally, the dynamic data distributions and the approach of removing non-recent events from the datasets significantly reduced the training data that was used to fit the models. As a result, finding a performance balance between the data loss and the event's recency was challenging.

The datasets from the alerts that were chosen to be tuned are majorly composed of nominal data. Unfortunately, encoding nominal data can be incredibly challenging due to the curse of dimensionality. Most of the binary features resulting from the nominal variables' transformation into numerical data did not hold significant information to predict the target variable. This phenomenon occurred due to the high number of categories of the nominal variables, and hence, the binary variables' skewed class distribution. Due to this detail, it was challenging to select the optimal set of features to tune the models and obtain results that correspond to ideal expectations.

Finally, due to the temporal restrictions in a professional environment, it was not always possible to invest time to experiment with state-of-the-art algorithms and carefully examine the effects of each data processing step in the models' performances. This focus shift was different to an academic environment, such as during the master's program, where the academic aspect of algorithm comparison and evaluation is often the main focus point.

5.4. Lessons Learned

Before this internship, my only experience with data science projects was through academic projects and theoretical lessons. This professional adventure indeed emphasized the differences between the academic and corporate worlds. Naturally, academic projects do not cover all potential issues that one could encounter with real-life data.

One of the most remarkable differences between the usage of artificially created data that is meant to be used for machine learning experiments, and authentic data that is gathered from existing systems, is the unknown potential that the projects yield. For instance, there were no possible predictions about the projects' outcomes before the first model's performance results. Furthermore, considering that the auditors manually and individually classify the events, there was no guaranteed way of ensuring that the datasets included a recognizable pattern. Therefore, projects of this category have to be developed step by step, and the performance expectations of the models have to be created simultaneously as the project progresses.

Contrary to academic projects that are developed for a single course and may not be looked at ever again, the code and ideas developed for a professional project are saved for other colleagues to consult and further develop. Hence, the importance of documentation and logical coherence of the decisions that are made are essential.

5.5. Future work

Due to temporal restrictions and expectations, there is still much improvement potential for the models implemented during this internship, especially for Alert B. However, the sky is the limit in machine learning, and new methodologies are suggested and tested every day.

Even though the outcome of this project was at least as good as expected, and the auditors' workload was significantly reduced, the ratio between recall and precision, that is, the risk of non-retrieved confirmed errors and the ratio of correctly triggered events could still be optimized. Eventually, further optimization requires a deeper business understanding of the data.

Furthermore, ML models could and will be applied to other alerts at DAI, since the implemented projects have been successful. Knowing that these projects were the pioneer machine learning implementations to the Department for Internal Audit at CGD, I am confident that many more applications to other databases are possible and would benefit the auditors. Fortunately, the interest in data science and machine learning is strong and growing with the omnipresence of AI in today's world.

Finally, I would like to emphasize that my internship at CGD was a great experience that undoubtedly opened many doors to my future data science career path. It was a great continuation of the curricular component of the master's program, and a great conclusion of the DSAA-DS degree.

6. BIBLIOGRAPHY

A. Arauzo-Azofra, J. M. Benitez, and J. L. Castro. 2008. “Consistency Measures for Feature Selection.” *Journal of Intelligent Information Systems* 30(3):273–92. doi: 10.1007/s10844-007-0037-0.

A. M. Turing. 1950. “Computing Machinery and Intelligence.” *Mind* LIX(236). doi: 10.1093/mind/LIX.236.433.

Althnian, Alhanoof, Duaa AlSaeed, Heyam Al-Baity, Amani Samha, Alanoud bin Dris, Najla Alzakari, Afnan Abou Elwafa, and Heba Kurdi. 2021. “Impact of Dataset Size on Classification Performance: An Empirical Evaluation in the Medical Domain.” *Applied Sciences* 11(2). doi: 10.3390/app11020796.

Anaconda Inc. 2020. “Anaconda Software Distribution.”

Bellman, Richard. 1972. *Dynamic Programming*. Univ. Pr.

Berrar, Daniel. 2018. “Cross-Validation.” Pp. 542–45 in *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*. Vols. 1–3. Elsevier.

Blair, David C. 1979. “Information Retrieval, 2nd Ed. C.J. Van Rijsbergen. London: Butterworths; 1979: 208 Pp. Price: $32.50.” *Journal of the American Society for Information Science* 30(6). doi: 10.1002/asi.4630300621.

Breiman, Leo. 2001. *Random Forests*. Vol. 45.

Canbek, Gurol, Seref Sagiroglu, Tugba Taskaya Temizel, and Nazife Baykal. 2017. “Binary Classification Performance Measures/Metrics: A Comprehensive Visualized Roadmap to Gain New Insights.” in *2017 International Conference on Computer Science and Engineering (UBMK)*. IEEE.

Chawla, Nitesh v, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. *SMOTE: Synthetic Minority Over-Sampling Technique*. Vol. 16.

Cover, Thomas M., and Joy A. Thomas. 2005. *Elements of Information Theory*. Wiley.

Dalianis, Hercules. 2018. *Clinical Text Mining: Secondary Use of Electronic Patient Records*. Springer International Publishing.

Dash ’, M., and H. Liu. 1997. *Feature Selection for Classification*. Vol. 1.

Douzas, Georgios, Fernando Bacao, Joao Fonseca, and Manvel Khudinyan. 2019. “Imbalanced Learning in Land Cover Classification: Improving Minority Classes’ Prediction Accuracy Using the Geometric SMOTE Algorithm.” *Remote Sensing* 11(24). doi: 10.3390/rs11243040.

Du, Ke-Lin, and M. N. S. Swamy. 2014. “Perceptrons.” Pp. 67–81 in *Neural Networks and Statistical Learning*. Springer London.

Duangsoithong, Rakkrit, and Terry Windeatt. 2010. “Bootstrap Feature Selection for Ensemble Classifiers.”

Fayyad, Usama, and Paul Stolorz. 1997. “Data Mining and KDD: Promise and Challenges.” *Future Generation Computer Systems* 13(2–3). doi: 10.1016/S0167-739X(97)00015-0.

Fernández, Alberto, Salvador García, Mikel Galar, Ronaldo C. Prati, Bartosz Krawczyk, and Francisco Herrera. 2018. *Learning from Imbalanced Data Sets*. Cham: Springer International Publishing.

García, Salvador, Julián Luengo, and Francisco Herrera. 2015. *Data Preprocessing in Data Mining*. Vol. 72. Cham: Springer International Publishing.

Guo, Gongde, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. 2003. “KNN Model-Based Approach in Classification.”

Hand, David J., and Niall M. Adams. 2015. “Data Mining.” in *Wiley StatsRef: Statistics Reference Online*. Wiley.

Hira, Zena M., and Duncan F. Gillies. 2015. “A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data.” *Advances in Bioinformatics* 2015. doi: 10.1155/2015/198363.

Ilyas, Ihab F., and Xu Chu. 2019. *Data Cleaning*. Association for Computing Machinery.

J. Mccarthy. 2007. *What Is Artificial Intelligence?*

Kotsiantis, S. B. 2007. *Supervised Machine Learning: A Review of Classification Techniques*. Vol. 31.

Kuhn, Max, and Kjell Johnson. 2013. *Applied Predictive Modeling*. New York, NY: Springer New York.

Limshuebchuey, Asavaron, Rakkrit Duangsoithong, and Terry Windeatt. 2015. “Redundant Feature Identification and Redundancy Analysis for Causal Feature Selection.” in *2015 8th Biomedical Engineering International Conference (BMEiCON)*. IEEE.

Loh, Wei Yin. 2011. “Classification and Regression Trees.” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1(1):14–23. doi: 10.1002/widm.8.

Murphy, Kevin P. 2015. *Machine Learning A Probabilistic Perspective*. Cambridge, Mass.

Nargesian, Fatemeh, Horst Samulowitz, Udayan Khurana, Elias B. Khalil, and Deepak Turaga. 2017. “Learning Feature Engineering for Classification.” Pp. 2529–35 in *IJCAI International Joint Conference on Artificial Intelligence*. Vol. 0. International Joint Conferences on Artificial Intelligence.

Opitz, David, and Richard Maclin. 1999. *Popular Ensemble Methods: An Empirical Study*. Vol. 11.

Pearson, Karl. 1900. “On the Criterion That a given System of Deviations from the Probable in the Case of a Correlated System of Variables Is Such That It Can Be Reasonably Supposed to Have Arisen from Random Sampling.” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50(302). doi: 10.1080/14786440009463897.

Pedregosa et al. 2011. “Scikit-Learn: Machine Learning in Python.”

Plotly Technologies Inc. 2015. “Collaborative Data Science.”

Quinlan, Philip T. 1998. “Structural Change and Development in Real and Artificial Neural Networks.” *Neural Networks* 11(4). doi: 10.1016/S0893-6080(98)00033-1.

R. Bellman. 1978. *An Introduction to Artificial Intelligence : Can Computers Think?* San Francisco: Boyd & Fraser Pub. Co.

R. Quinlan, M. Kaufmann, and S. L. Salzberg. 1994. *Programs for Machine Learning*. Vol. 16.

Ray Kurzweil, and Diane Jaroch. 1990. *The Age of Intelligent Machines*. Cambridge, Mass: MIT Press.

Rosenblatt, F. 1958. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” *Psychological Review* 65(6). doi: 10.1037/h0042519.

van Rossum, Guido and Drake Jr, Fred L. 1995. “Python Reference Manual.”

Rutkowski, Leszek, Maciej Jaworski, Lena Pietruczuk, and Piotr Duda. 2014. “The CART Decision Tree for Mining Data Streams.” *Information Sciences* 266. doi: 10.1016/j.ins.2013.12.060.

S. J. Russell, and P. Norvig. 1995. *Artificial Intelligence : A Modern Approach*. Prentice-Hall.

Sun, Yanmin, Andrew K. C. Wong, and Mohamed S. Kamel. 2009. “Classification of Imbalanced Data: A Review.” *International Journal of Pattern Recognition and Artificial Intelligence* 23(04). doi: 10.1142/S0218001409007326.

T. M. Mitchell. 1997. *Machine Learning*. Singapore: McGraw-Hill.

Talavera, Luis. 2005. “An Evaluation of Filter and Wrapper Methods for Feature Selection in Categorical Clustering.”

Wirth, Rüdiger, and Jochen Hipp. 2000. *CRISP-DM: Towards a Standard Process Model for Data Mining*.

Xu, Yun, and Royston Goodacre. 2018. “On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning.” *Journal of Analysis and Testing* 2(3):249–62. doi: 10.1007/s41664-018-0068-2.