FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR'S THESIS

Title:	Machine learning for financial crime detection
Student:	Stanislav Němec
Supervisor:	MSc. Juan Pablo Maldonado Lopez, Ph.D.
Study Programme:	Informatics
Study Branch:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	Until the end of summer semester 2019/20

Instructions

Financial institutions face increasing pressure from regulators to prevent financial crime: money laundering, terrorism funding and tax evasion. Part of the challenge is to design systems that produce as little false positives as possible, as this minimizes the work that human analysts need to perform. The goal of this work is:

- Survey the different data mining techniques for financial crimes.
- Implement a baseline algorithm that identifies fraudulent transactions and entities (perhaps separately).
- Propose different improvements on the baseline algorithm while reducing the number of false positives.
- Implement the improved version and compare its performance and interpretability.

References

Will be provided by the supervisor.

Ing. Karel Klouda, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague December 19, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Machine learning for financial crime detection

Stanislav Němec

Department of Applied Mathematics
Supervisor: MSc. Juan Pablo Maldonado Lopez, Ph.D.

May 16, 2019

Acknowledgements

I would like to express my sincere gratitude to the supervisor of my thesis MSc. Juan Pablo Maldonado Lopez, Ph.D. for all the advice and help.

Declaration

In Prague on May 16, 2019 . . . .

Citation of this thesis

Němec, Stanislav. *Machine learning for financial crime detection.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato práce se zabývá návrhem modelu pro detekci pokusů o finanční podvody za pomoci strojového učení. Cílem práce je vybrat a vyhodnotit základní model a po vyhodnocení jeho přesnosti jej upravit a rozšířit. Účelem úprav základního modelu je zvýšení přesnosti modelu a eliminace případů, kdy model označí běžné chování za podvodné.

Na základě rešerše existujících a používaných řešení je vybrán rozhodovací strom jako algoritmus pro základní model. Poté je provedena rešerše možných rozšíření tohoto algoritmu. Vybrané algoritmy a rozšíření, jako je zavedení *cost-sensitivity* pro rozhodovací stromy nebo shlukování rozhodovacích stromů pomocí metody AdaCost, jsou následně realizovány pomocí knihoven jazyka Python. Vybrané algoritmy jsou trénovány a testovány na simulovaných datech finančních transakcí.

Výsledky experimentální části práce ukazují, že vylepšené modely jsou úspěšnější v porovnání se základním modelem. Aplikování *cost-sensitivity* vedlo k nalezení vyváženého kompromisu mezi eliminací počtu falešných obvinění a odhalením větší části podvodů. Prototyp algoritmu AdaCost také dosáhl lepších výsledků v porovnání se základním modelem.

Přínosem této práce je vyhodnocení navržených a realizovaných úprav pro rozhodovací stromy, které mohou být zužitkovány při návrhu systémů pro detekci finančních podvodů.

Klíčová slova klasifikační algoritmy, detekce finančních podvodů, rozhodovací stromy, vylepšení rozhodovacích stromů, boosting, AdaCost, detekce praní špinavých peněz

Abstract

This work focuses on designing a machine learning model for financial crime detection. The goal of this work is to select a baseline model and apply it to the financial dataset. After evaluating it, propose extensions and improvements to it with an aim to improve its performance and reduce the number of activities falsely classified as fraudulent.

Based on the survey of existing solutions, decision tree algorithm was selected as the baseline model. Afterwards a study of possible improvements and extensions to this algorithm is carried out. Proposed improvements, such as introducing cost-sensitivity and cost-sensitive ensemble called AdaCost, are applied and evaluated using the Python programming language. The experiments are carried out using simulated money transactions.

The results of the experimental part show that the improvements applied to the baseline model were successful. The cost-sensitivity helped to find a model with a good balance between eliminating the false accusations and detecting a majority of frauds. The prototype of the AdaCost algorithm also showed better results when compared to the baseline model.

The usefulness of this work comes from the evaluation of proposed improvements to decision trees, that can be utilized while designing systems for financial fraud detection.

Keywords classification algorithms, financial fraud detection, decision tree, improvements to decision trees, boosting, AdaCost, money laundering detection

Contents

List of Figures

List of Tables

Introduction

Financial crime goes hand in hand with tax evasion and thus it causes huge financial losses for states. Most of the money connected to financial crime finances terrorist and other criminal organisations. That is why state institutes put a lot of pressure on financial institutions to monitor the activity of their customers and detect any suspicious behaviour. Considering the volume of records that bank customers produce, it is almost impossible to accomplish this task without its automation.

Systems for detecting suspicious behaviour, that are currently used by financial institutions, are mostly based on sets of rules defined by experts and consultants. The rules are based on the best practices and information collected from different institutions. The rules are not very flexible and adaptable which makes them prone to being abused and bypassed. Other problem is that the systems are producing high numbers of false positives - normal entities marked as suspicious. This creates an unnecessary backlog of work that the employees are unable to process.

Solutions based on machine learning can be continuously trained and updated with new data which makes them adaptable and eliminates the need for consultants defining new rules. The solution proposed in this work is based on decision tree algorithm (DT). DT is suitable because it is easily visualized and interpreted which provides the reasoning behind its decisions. Extensions to the decision tree, examined in this work, aim at reducing the number of false positives. The drawback of the extensions is that they also reduce the number of detected frauds and creates a trade-off between false positives and undetected frauds.

The result of this work can help financial institutions with developing or improving systems generating suspicious activity reports. It will help to implement systems that will relieve some pressure of bank analysts by eliminating some of the workload they face.

The first two chapters provide introduction to the goals of this work and to the problem of financial fraud respectively. The following chapter provides basic introduction to machine learning, the fourth chapter introduces the used dataset. The fifth chapter is a survey of existing solutions and presents theoretical foundation for algorithms selected for this work. After briefly describing utilized technologies in chapter six, the seventh chapter contains the experimental part. The experimental part follows a methodology called CRISP-DM, that provides a guideline for carrying out machine learning projects.

Chapter 1

Goals of the work

The goal of this work is to survey data mining techniques that could be used for detecting financial crime and propose and implement a model for this problem based on the gained information.

In the research part of this work I will survey different data mining techniques that are used for detecting financial crime. After gaining an overview of possible solutions, I am going to choose a baseline model for the experimental part. I will also survey possible extensions to the selected algorithm that should increase the performance of the model.

In the experimental part I will evaluate a selected baseline algorithm that identifies fraudulent entities. After evaluating the performance of the baseline model, I will apply to it the improvements chosen in the research part. The aim is to keep the number of false positives as low as possible while also reducing the number of undetected frauds. After implementing the improved model I am going to evaluate its performance and interpretability and make a comparison between all the models.

Chapter 2

Financial crime

This chapter briefly explains the terms used in this work that are connected to financial crime. It will also help to better understand the purpose of this work.

2.1 Money laundering

The purpose of money laundering is to conceal the origin of money generated by illicit business. It consists of many activities and techniques that aim at making the illegal income seem legitimate in front of controlling institutions. [1]

2.1.1 Money laundering techniques

Criminals have developed many techniques over the years. The evolution and increasing presence of technology in financial sector, such as online banking, online payment services, mobile transactions or virtual currencies, allow the criminal organizations to find novel ways to launder their illegal funds. [2]

The techniques involving banks and other financial institutions usually consist of three stages. First the cash is broken down into smaller sums and deposited in multiple accounts. Then the money is run through a series of transactions so that it is almost impossible to trace it back to it's original source. At last the money is returned to the criminals, usually in a form of expensive wares. [3]

2.2 Anti money laundering

Anti money laundering (AML) consists of regulations and procedures that aim to prevent money laundering attempts or financing criminal organisations. All entities in the financial market are obliged to follow these regulations. [4]

Financial institutions execute various strategies in order to prevent and detect the cases of money laundering. Banks use controlling systems that monitor their customers and their transactions. When the systems detect any suspicious activity, it notifies employees who then examine the situation and determine whether it should be reported to law enforcement as a financial fraud. The report is called Suspicious activity report (SAR). SARs are then investigated by law enforcement and can potentially lead to uncovering criminal offenders. [5]

Current systems used by financial institutions are mostly based on rules assembled by consultants. The rules are based on best practices and experience collected from the clients. Having a fixed set of rules creates a risk of them being exposed and thus easily circumvented. [6]

2.2.1 Demands on practical AML systems

Financial institutions need to prove that they are efficiently processing the alerts generated by their systems. To prove the efficiency they need to show that they are able to process and successfully evaluate all the generated reports. When the system produces high number of false accusations, it causes an emergence of backlogs of unprocessed alerts. This is why the system is expected to produce low number of false positives, even if it means not detecting all the frauds. [6]

Other important feature, that system for detecting suspicious transactions should have, is an ability to provide a reasoning behind its decision. When producing a SAR, there needs to be an explanation of why the activity is considered suspicious.

Chapter 3

Machine learning essentials

This chapter provides a general overview and explanation of the terms related to machine learning that are going to be used in this work.

3.1 Machine learning

Machine learning (ML) is a popular term that appears in many fields of industry with no fixed definition. Tom Mitchell [7] describes machine learning by the question this field of science tries to answer. The question is how to build computer systems and programs that automatically improve in specified tasks with experience.

Ian H. Witten and Eibe Frank [8] state that the purpose of ML is to extract information and relations from raw data and discover structures that underlie it. The information should be in comprehensible form and should be usable for given purposes.

To summarize this term I will use the common parts of most of the definitions. ML covers means (mainly algorithms) of making machines learn to make a decision or prediction based on input data and experience gained from past observation.

3.2 ML model

Machine learning model is a final product of the learning process where ML algorithm is trained on the input data. The trained model can then be used to produce an output (e.g., decision, prediction) when presented previously unseen input. [9]

A ML model can also be seen as a suitable function $f(X) = y$ that maps input data X on to an output y (decision, prediction, ...). [10]

3.2.1 Supervised learning

The goal of supervised learning is to create a model $g(\cdot)$ that tries to learn relations between an input X and an output Y. The training set for supervised learning consists of pairs of input x and known output y. The model is defined as:

$$y = g(x|\theta)$$

where θ are the parameters of the model. The parameters θ are optimized during the training phase in order to minimize the error of the prediction. In other words, to get the prediction as close to the correct output values from the training set. [11]

3.2.2 Classification

Classification is one of the supervised learning problems. The output y is called a class and it is a categorical and discrete attribute. The model is called a discriminator or classifier and its purpose is to separate the samples into given classes. The class attribute has a finite number of possible values. The case When the target class has only two values is called binary classification. [11]

3.3 CRISP-DM

Cross-Industry Standard Process for Data Mining (CRISP-DM) is a methodology that serves as a guideline for carrying out a data mining project. The process is formed of the six following stages:

Business understanding stage is about discovering how the business can benefit from data mining;

Data understanding is a stage when one takes a closer look at the data to find out what the format is and how consistent the dataset is;

Data preparation stage is for data cleansing and preprocessing;

Modeling is a stage for selecting a suitable model and training it on the prepared dataset;

Evaluation is where the performance of the model is measured;

Deployment is a step when the model is integrated with other systems and utilized to make improvements.

The stages do not necessarily need to follow the above order. It is common that one goes back to former stages or skipping some. For example when the results in evaluation stage are insufficient, the process might return to the Data Preparation stage. [12]

3.3.1 Analyzing and understanding the data

This activity corresponds to the Data Understanding stage in the CRISP-DM methodology. The goal of this phase is to have a first look at the data, analyze its properties explore it and verify its quality.

First step, before analyzing can begin, is to collect the data and load it into a chosen tool for exploring data. This can also involve integrating multiple data sources or merging multiple tables.

In the next step, the goal is to describe the loaded data. This includes determining its format, finding out how big the dataset is (number of records and attributes) and understanding what particular attributes represent.

After understanding the format and structure of the dataset, one can start with the data exploration. The outcome of this step is gaining a better insight into the data. This insight is gained by computing basic statistics, determining the relationships between individual attributes, finding the distributions of key attributes or by aggregating the data.

Other step is about verifying the quality of the dataset. That means checking whether there are some missing values or whether there are some errors in the records. [12]

3.3.2 Data preprocessing

Data preprocessing is a vital step in the ML procedure and should be one of the first steps taken. The purpose of data preprocessing is to make sure that the data is complete, correct and processable by algorithms in the further steps of the project. The preprocessing phase consists of multiple techniques sorted into categories like data cleansing, data integration and data transformation. [13]

Data cleansing aims at handling problems such as missing values, records violating some constraints or regulations or records that probably originate from an error during the data collecting.

Missing values are usually resolved by filling in the values based on some statistics of the data or based on some similar records. Or if the size of the dataset allows it, the records with missing values are discarded.

Records that violate some constraints can occur for example as mistakes amongst categorical attributes (e.g., values should be *"true"* or *"false"* and *"truth"* occurs). Another type of data violating some rules are values in a wrong format (e.g., decimal number is expected and there is a alphabetical character). Values that do not make sense in given context also fall into this category. (e.g., negative number between size measurement data. Possible ways of dealing with these faulty records are to manually correct them or to leave them out. [13]

Data integration means putting together data from various sources. [13] This involves not only joining data from different databases and files, but also converting all the data into united structure and format.

Data transformation covers multiple techniques for transforming and adjusting the data. More information regarding this topic can be found in Data Preprocessing Techniques for Data Mining [13].

Feature engineering focuses on adding new attributes to the records based on those that are already present in the dataset. Forming the new features is a good way to embed a knowledge of given topic. New attributes can be created by applying basic mathematical operations or even more complex formulas on the existing ones. Another way of introducing new attributes is to aggregate data using time periods. [14]

It is strongly recommended to **split the dataset** into two parts. The first part is called the training set and is used to train the ML model. The other part is called the test set and is used to evaluate the model after it has been trained. The splitting ratio varies, but it is common to use 70% of the data for training and save the remaining 30% for testing. It is also important to have all types of data evenly represented in the split parts. [11]

3.4 Evaluating machine learning models

When the ML model is trained, it is important to check how well it learned to perform it's task before it is applied in real life. This is where evaluation metrics come into play and help us determine how accurate the model is or how many mistakes it makes.

3.4.1 Confusion matrix and accuracy

In a case of a binary classifier with positive and negative classes, there are four possible outcomes when comparing the predicted values to the true values. To find out how the model performed, a matrix is constructed. The matrix contains sums of following cases over the observed set of data:

True Positives (TP) are cases where the model correctly predicted the positive class.

True Negatives (TN) are samples correctly predicted by the model that belong to the negative class.

False Positives (FP) are cases where the model predicted the positive class while the correct output was the negative class.

False Negatives (FN) are samples that the model labeled as the negative class but the correct label is the positive class.

It is often convenient to have a single value as a result of evaluation. One of the measures can be accuracy that is calculated by dividing correct predictions by the total number of samples, expressed by the following equation [8]:

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

Table 3.1: Visualization of the confusion matrix

	Predicted positive	Predicted negative
Original positive	TP	FN
Original negative	FP	TN

3.4.2 Recall and balanced accuracy

Balanced accuracy is an evaluation metric for classifiers that utilizes recalls obtained on all classes (both classes in case of binary classification).

Recall of a class expresses the fraction of detected test samples of given class. [8] In other words, it is obtained by dividing the number of detected samples of the class by the total number of samples of the given class. In binary classification problem (with classes marked as positive and negative), the equation for recall (for the positive class) can be expressed using terms from the confusion matrix:

$$recall_{positive} = \frac{\sum TP}{\sum TP + \sum FN}$$

Balanced accuracy is calculated by averaging the recalls obtained on each class. [15] I will use the metric for binary classification (with positive and negative classes), expressed as follows:

$$balancedAccuracy = \frac{recall_{positive} + recall_{negative}}{2}$$

It is a good metric to evaluate a model trained and tested on dataset with uneven distribution of target classes, because it punishes models that favour one class over the other.

3.4.3 Weighted f-score

Basic F-score is obtained by computing the harmonic mean H of recall and precision. With n being the number of values and x_i the values, harmonic mean is:

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Precision (with focus on positive class) of a binary classifier in a test is the number of correctly classified positives divided by the number of all samples classified as positive. [8]

$$precision_{positive} = \frac{TP}{TP + FP}$$

The F-score is then expressed as follows:

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} = \frac{2PR}{P + R}$$

where P is precision and R is recall. [16]

Weighted F-score is then computed as a weighted mean of the f-scores for each class. [17] The weight w_i for class i is equal to the number of samples from the test set belonging to the class i. With n being the number of classes and F_i the F-score for class i, weighted f-score is:

$$f_{weighted} = \frac{\sum_{i=1}^{n} w_i F_i}{\sum_{i=1}^{n} w_i}$$

3.5 Imbalanced classes

The problem of imbalanced classes occurs when the dataset used for training a classifier has an uneven distribution of the target classes. In other words one of the classes is represented by much larger portion of samples in a dataset. [18] This problem can lead to models that favor the class that contains the majority of the samples over the other classes when not taken into consideration.

Chapter 4

Description of used data

This chapter describes the dataset used in this work. It also provides information about the origin of the data and reasons why I decided to use this dataset.

4.1 Origin of the data - AMLSim

Dataset generated using the AMLSim project [19] is used in the experimental part of this work. AMLSim simulates the transactions and brings a fraud patterns into the data in two phases.

During the first phase, a network of transactions is generated and alerts signaling suspicious activities are introduced. The network represents relations, or transactions, between the accounts of customers.

The second phase uses another project called PaySim to simulate the transactions in the network generated during the first phase. The algorithm in PaySim was presented a data of real mobile transactions which it was supposed to imitate. PaySim utilizes statistics and Agent-Based simulation to create synthetic data with characteristics similar to the original data. [20]

Considering the nature and sensitivity of the data that is needed for training AML models, there is no real data publicly available. AML Models can be trained on records of labeled money transactions and using various personal information about customers [21]. Financial institutions cannot publicly share this data about customers and their transactions so there is a need for synthetic dataset simulating real life data.

4.2 Dataset description - Tables

The dataset consists of approximately 10000 generated accounts with over 500000 transactional records over 150 time steps which is equivalent to 5 months.

Each row contains step in which the transaction was carried out (one step corresponds to one day), identification of both participants of the transaction, their account balance before and after the transaction, the amount transferred and the type of the transaction. Each record also contains an indicator of whether the transaction is fraudulent. Every record, or transaction, is labeled as suspicious if it is a part of an activity that has been flagged as possibly fraudulent.

The time step corresponds to one day of real time. The currency of the transactions is unspecified but is uniform for all transactions. Types of transactions are either incoming and outgoing cash transactions or normal non-cash transaction.

Chapter 5

Survey of solutions

This chapter provides an overview of existing solutions for AML systems. The other section of this chapter introduces algorithms selected for the experimental part of this work.

5.1 Survey of existing solutions

The details of the solutions used by financial institutions are rarely publicly available. They have been using systems based on rules defined and tuned over time by experts.

An article presenting an overview of machine learning (ML) algorithms used in experiments on this topic was published by Chen et al. [21]. The authors have divided known ML algorithms into six categories, including AML typologies, Link analysis, Behavioural modelling, Risk scoring, Anomaly detection and Geographic capability. The division is based on the aspects taken into consideration by the algorithm when detecting suspicious behaviour.

AML typologies use records of past cases of money laundering to detect similar future attempts. That means that either data with labeled fraudulent activities or an assistance from a domain expert is needed to train models of this type. Chen and Mathe [22] utilized fuzzy rules to detect suspicious activities, their approach is to convert the rules created by analysts into a set of fuzzy rules and then use inference to detect suspicious transactions. Other approach is to treat this task as a classification problem and solve it using Support Vector Machine [23].

Link analysis based solutions focus on identifying connections between customers' accounts to discover relationships between them. The transactions between accounts form graphs that can be analyzed to find potentially fraudulent patterns. [24] The resulting structures or patterns are visualized to support further analyzing.

Behavioural modelling combines various information sources to construct a behavioural model of customers. New activities are then compared to the modelled behaviour.

Risk scoring solutions rank transactions or customers by a potential risk. The risk score is obtained by applying statistical models and business rules. The ranking is taken into consideration when deciding whether a report should be generated. Wang and Yang [25] presented a solution using a decision tree algorithm to rank the customers. Another approach by Liu et al. [26] analyzes sequences of transactions over time, where new activities are compared to the historical ones.

Anomaly detection models identify activities deviating from a norm. Approaches based on anomaly detection should be able to identify uncommon activities within an account.

Geographic capability tool aims at identifying fraudulent activities carried out across different geographical locations and states. It requires data about geographic locations of the participating parts.

I decided to focus on the risk scoring approach to this problem, namely algorithms involving decision trees. This approach is suitable with regard to being applied on the available dataset that contains all the needed information. The predictions produced by decision trees can also easily be interpreted and explained, thus solutions using this algorithm satisfy one of the demands on systems for detecting fraudulent activities.

5.2 Baseline model - decision tree

Decision tree (DT) or a tree structured classifier is constructed by repeatedly splitting the subsets of a dataset based on values of samples' variables. The whole dataset is split first and then the created subsets are split until a condition for stopping is met. I will focus on a binary tree structured classifier where each split creates two subsets. The samples that meet the condition set by the splitting rule are sent to the left subset (child node) and the rest to the right one. The subsets that are not split are called terminal subsets or terminal nodes. The union of terminal subsets forms the original set. Each terminal node is assigned a class label. Further I will describe the Classification and Regression Trees (CART) [27] approach for building the tree structure, because this approach is used in the scikit-learn implementation of DT. [28]
The process of constructing a DT requires the following parts:

- A set of questions Q (splitting rules) with two possible answers (True or False), where the questions are in a form of $\{Is\ x \in A?\}$, where $x \in X$, X is a set of data records and $A \subset X$.
- A function $\Phi(s, t)$ evaluating how good the split s is on node t.
- Conditions that define when to stop splitting.
- A rule that defines which class label will be assigned to each terminal node.

5.2.1 Splitting rules

Data records have standard structure when they are in form of vectors $\mathbf{x} = (x_1, x_2, \ldots, x_M)$ of fixed dimensionality M, where $x_1, ..., x_M$ are variables of either categorical or ordered type. With data having the standard structure, standardized set of splitting rules Q can be defined as:

1. The value of only a single variable is taken into consideration in each rule.
2. Q contains all questions in form of $\{\mathit{Is}\ x_i \leq c\,?\}$ for each ordered variable x_i and $\forall c \in (-\infty, \infty)$.
3. For a categorical variable x_i that is taking possible values in $V = \{v_1, v_2, \ldots, v_L\}, L \in \mathbb{N}$, Q contains questions $\{\mathit{Is}\ x_i \in v\,?\}$ for all subsets v of V.

There is a finite number of splitting rules for each ordered variable x_i restricted by the number of distinct values of x_i in the dataset. For categorical variable x_i with possible values from $V = \{v_1, v_2, \ldots, v_L\}, L \in \mathbb{N}$, the number of splitting rules is restricted by $2^{L-1} - 1$. Only a half of all the possible subsets v of V needs to be tested, beacause cases of $\{x_i \in v\}$ and $\{x_i \notin v\}$ are both tested within one rule and empty set is not tested. [27]

5.2.2 Splitting criterion

Let $\pi(i)$ be the prior probability of class i, interpreted as a probability of sample belonging to class i being presented to the tree. The proportion of samples of class i to the total number of samples, $\pi(i) = N_i/N$, is often used to estimate $\pi(i)$. $N_i(t)$ is the number of samples labeled as i present in a node t. A probability that a sample both belongs to class i and is in a node t is expressed by

$$p(i,t) = \pi(i)N_i(t)/N_i.$$

With J being the set of classes, the estimation of probability $p(t)$ of any sample getting into node t is

$$p(t) = \sum_{i \in J} p(i,t).$$

An estimated probability that case belongs to class i given that it is in node t is

$$p(i|t) = p(i,t)/p(t).$$

The function, that evaluates how good a split is, is called a split criterion. Split criterion is originally based on impurity function. Impurity function ϕ is defined on a set of all J-tuples $(p_1, p_2, \dots, p_J)$ where $p_i \geq 0, \forall i \in \{1, \dots, J\}$ and $\sum_{i=1}^{J} p_i = 1$, J is the number of classes. The function ϕ has the following properties:

1. ϕ reaches its maximum at the point $(\frac{1}{J}, \dots, \frac{1}{J})$,
2. ϕ reaches its minimum at the points $(1, 0, \dots, 0)$, $(0, 1, 0, \dots, 0)$, $\dots$, $(0, 0, \dots, 0, 1)$,
3. ϕ is a symmetric function of $p_1, \dots, p_J$.

Given the impurity function ϕ, the impurity measure $i(t)$ of a node t is defined as

$$i(t) = \phi(p(1|t), p(2|t), \dots, p(J|t)).$$

The split s on node t is evaluated by the decrease in impurity $\Delta i(s,t)$ it provides which is expressed as follows:

$$\Delta i(s,t) = i(t) - p_R i(t_r) - p_L i(t_L)$$

where p_R and p_L are the proportions of data samples sent to the right and left child node respectively. [27]

The total decrease brought by all rules containing one attribute can express the importance of the attribute, or **feature importance**.

The scikit-learn implementation of DT allows the usage of either Gini

$$i(t) = \sum_{i \in J} p(i|t)(1 - p(i|t))$$

or Entropy

$$i(t) = -\sum_{i \in J} p(i|t) \log(p(i|t))$$

as the impurity measures. [28]

5.2.3 When to stop splitting

When there are no additional restrictions, the splitting in node t is stopped when there is no split that would bring a decrease in impurity. This can be restricted by a threshold, so that the splitting will stop when there is no split that would decrease the impurity at least by an amount equal to the threshold. [27]

One way to restrict further splitting is to set the maximal depth of the tree. The depth is equal to the number of splits (nodes) on the longest way from the first split (root) to a terminal node.

Setting the minimal number of samples in a node can also be used to terminate splitting. In this case, the node is denoted as terminal if all the splits would create child nodes containing lower number of samples than the defined minimum.

5.2.4 Class assignment rule

Each terminal node $t \in \tau$ is assigned a class label $j \in \{1, \dots, J\}$ following the class assignment rule. $j(t)$ notation is used to describe the class assigned to node t. The estimated probability of misclassification for a node t is:

$$\sum_{i \in J \setminus \{j(t)\}} p(i|t).$$

The rule is formed so that it minimizes this estimate. The class assignment rule is defined as:

$$\textit{If } p(j|t) = \max_i p(i|t) \textit{ then } j(t) = j.$$

In case that multiple classes reach the maximum, one of them is chosen arbitrarily. [27]

5.2.5 Classifying new samples

The class for a new sample is predicted using the constructed tree. The sample progresses through the tree depending on the splitting rules until it arrives in a terminal node. The class label assigned to the terminal node is then used as a prediction for given sample. The progression through the tree creates a set of rules that can be used to interpret the reasoning behind the decision.

5.3 Improving the baseline model

This section contains a description of techniques and algorithms that can be applied to the DT to improve its performance.

5.3.1 K-Fold cross-validation

First step of K-Fold cross-validation is splitting the dataset X into K parts of equal size, $\chi = \{X_1, X_2, \dots, X_K\}$. It is important to preserve the distribution of the classes in each of the splits. K pairs $\{T_i, V_i\}, i = 1, \dots, K$ of training and validation sets are created. The validation set V_i is equal to the split X_i and the combination of the rest of the splits forms the training set T_i. [11]

$$V_i = X_i$$

$$T_i = \chi \setminus X_i$$

The created pairs are then used to validate a model, training sets are used in the training phase and validation sets serve as sets for evaluating the model. This approach produces K different results that can be averaged to gain a single value.

Cross-validation can be used for selecting the parameters of a model. The combinations of selected values for parameters of the model form a parameter grid. All the settings from the grid are then used to create models that are validated in the K-Fold cross-validation. The best-performing model with its parameter setting is selected as the predictor. This approach is implemented in scikit-learn and used in the experimental part. [29]

5.3.2 Weighted classes

A weight w_j can be assigned to each class j. The weight then influences the probability estimates defined in 5.2.2.

The prior probability becomes

$$\pi(j) = \frac{w_j N_j}{\sum_{i \in J} w_i N_i}.$$

This adjusts all the probability estimates dependent on the prior probability. I will describe only the final forms showing the influence on class assignment rule and on Gini impurity.[1]

The Gini impurity used in scikit-learn [28] will take the following form when class weights are introduced:

$$i(t) = \sum_{i \in J} \frac{w_i N_i(t)}{\sum_{j \in J} w_j N_j(t)} (1 - \frac{w_i N_i(t)}{\sum_{j \in J} w_j N_j(t)}).$$

The rule for class assignment is then expressed as follows:

$$\textit{If } \frac{w_k N_k(t)}{\sum_{j \in J} w_j N_j(t)} = \max_i \frac{w_i N_i(t)}{\sum_{j \in J} w_j N_j(t)} \textit{ then } j(t) = k.$$

It is clear that the introduction of class weights influences both the splitting criterion and class assignment rule. Finding suitable values for the class weights thus can influence how the predictor is built and how it assigns class labels. Finding the values and evaluating the influence on accuracy of the predictor is addressed in the experimental part 7.4.2.

Assigning higher class weight to one class also has similar effect as random re-sampling. Random re-sampling means randomly selecting samples of the class with less representatives and including them in the dataset until both classes have similar representation. [30] Re-sampling is used to deal with class imbalance.

[1]All the adjusted probabilities can be computed by substituting in the new $\pi(j)$.

5.3.3 AdaBoost

AdaBoost is a boosting algorithm. The formalization of boosting is following: A training set of pairs $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, where x_i is a sample labeled as y_i, is presented to a booster. The booster defines a distribution D_t over the set of samples on each step $t = 1, \ldots, T$, and requests a hypothesis h_t with error ε_t. The distribution D_t assigns each sample an importance in step t. After T steps, the booster combines the hypotheses and creates one prediction rule. [31]

The AdaBoost algorithm is following:

Algorithm 1 AdaBoost

```
Require: set of N labeled samples $\{(x_1, y_1), \ldots, (x_N, y_N)\}$
         distribution $D$ over the $N$ samples
         Predictor that classifies samples
         integer $T$, the number of steps
1: Initialize the weight vector: $w_i^1 = D(i), i = 1, \ldots, N$
2: for $t = 1, 2, \ldots, T$ do
3:     Set
           $$p^t = \frac{w^t}{\sum_{i=1}^N w_i^t}$$
4:     Get hypothesis $h_t : X \to [0, 1]$ from calling Predictor and providing it
       with the distribution $p^t$.
5:     Calculate the error of $h_t$: $\epsilon_t = \sum_{i=1}^N p_i^t |h_t(x_i) - y_i|$
6:     Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$
7:     Set the new weight vector as
           $$w_i^{t+1} = w_i^t \beta_t^{1-|h_t(x_i)-y_i|}$$
8: end for
9: return the hypothesis
       $$h_f(x) = \begin{cases} 0 & \textit{if } \sum_{t=1}^T (\log 1/\beta_t) h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \log 1/\beta_t \\ 1 & \textit{otherwise.} \end{cases}$$
```

5.3.4 AdaCost

AdaCost is a variant of AdaBoost that is cost-sensitive. **Cost-sensitive** models' predictions are influenced by the misclassification cost. **Misclassification cost** $C_{i,j}$ is a cost for misclassifying a sample of class i as class j. In AdaCost, the misclassification costs are used when updating the sample weights in each step. [32]

Algorithm 2 AdaCost

```
Require: set of N labeled samples with costs
        $S = \{(x_1, c_1, y_1), \ldots, (x_N, c_N, y_N)\}$;
        $x_i \in X$, $X$ is a set of data samples, $c_i \in \mathbb{R}^+$, $y_i \in \{-1, 1\}$
        Predictor that classifies samples
        integer $T$, the number of steps
1: Initialize distribution $D_1$; $D_1(i) = c_i / \sum_{j=1}^{N} c_j$
2: for $t = 1, 2, \ldots, T$ do
3:     Get hypothesis $h_t : X \to \mathbb{R}$ from training Predictor and providing it
       with the distribution $D_t$.
4:     Choose $\alpha_t \in \mathbb{R}$
5:     Update the distribution
           $$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i) \beta(i)\right)}{Z_t}$$
       ▷ where $\beta(i) = \beta(sign(y_i h_t(x_i)), c_i)$ and $Z_t$ is a normalization factor so
       that $D_{t+1}$ is a distribution
6: end for
7: return the hypothesis
           $$H(x) = sign(f(x)) \text{ where } f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$
```

I decided to use the same selection of parameters α and β as the authors of AdaCost mentioned in their work [32]. The parameter α was selected as follows:

$$\alpha = \frac{1}{2} ln(\frac{1+r}{1-r})$$

where

$$r = \sum_i D(i)u_i,\ u_i = y_i h(x_i)\beta(i).$$

And the parameter β is following:

$$\beta(sign(y_i h_t(x_i)), c_i) = (-1)sign(y_i h_t(x_i))0.5c + 0.5.$$

Chapter 6

Technologies used

This chapter provides an overview of the technologies used in the experimental part.

6.1 Python

Python is a programming language with an extensive community that is also involved in developing and extending the language. [33] The extensions are called packages or libraries and include many reliable and open source implementations of machine learning algorithms, methods for manipulating with data and visualization tools. The functionality and stability of the extensions and the fact that Python is easy to understand and use are one of many reasons why Python is commonly used by data scientists.

6.1.1 Jupyter

Jupyter is an open-source project that provides interactive environment for scientific computation across various programming languages. [34]

6.1.2 Pandas

Pandas is an open source library for Python. It provides data structures that are described as high-performance and easy-to-use. The data structures and their functionality help with analyzing various types of data.

The library provides functions for loading input data in different formats and storing them in unified data structures. The loaded data can then be viewed, filtered, sorted or transformed with the help of the functions implemented in the library. [35]

6.1.3 Matplotlib

Matplotlib is a Python library that helps creating various types of 2D plots. It is intuitive for basic use, but also provides advanced options for full control of line styles, fonts and axes properties. [36]

6.1.4 SKLearn

This Python library provides simple and efficient tools for data mining and data analysis. It implements many of the commonly used machine learning algorithms, tools for data exploration and preprocessing and means to evaluate the efficiency of trained models. [37]

6.1.5 NumPy

NumPy is a package providing tools for scientific computing in the Python programming language. [38] The functionality utilized in this work is the N-dimensional array object and ability to carry out mathematical operations on its elements.

Chapter 7

Realization

This chapter describes the practical realization of this thesis' goal, which is implementing chosen models and proposed improvements. The experimental part follows the CRISP-DM methodology, outlined in 3.3. The chapter contains evaluation of the experiments and a summary of how the experiments worked out. The source code of all the following parts can be found on the enclosed CD, its content is described in appendix B.

7.1 Data analysis

This part corresponds to the first step of CRISP-DM and is described in 3.3.1. It contains the discovery of the structure and features of the dataset.

After I loaded the data from the CSV file, I printed out five rows from the dataset to get the intuition of what the structure of the data is and what values can I expect for the attributes. The attributes were already described in 4.2.

After getting familiar with the data, I searched the dataset for missing values or some inconsistencies. There were no missing values discovered, which was expected since the data is synthetic. However, I found approximately 1500 transactions where the amount of money transferred was 0. I found out that none of these transactions were reported as part of a fraud, so I consider them insignificant and decided to leave them out.

7.2 Data preprocessing

The work described in this section corresponds to the second step of CRISP-DM methodology that is described in 3.3.2. In chapter 5 I selected the approach of scoring entities as fraudulent or not, that means that I need data aggregated per account for training the model.

First I addressed the issues discovered when analyzing the data by filtering out the unwanted records. After that, with clean and consistent dataset, I extracted all account identifiers. Then I decided to separate the transactions based on their type into two categories, cash and non-cash transactions. For each type of transactions, I collected aggregated data for every account, specifically the summed incoming and outgoing amounts of money and the number of incoming and outgoing transactions of each type. I also retrieved the mean balance for each account. I marked each account that has been involved in some fraud as suspicious, this indicator serves as the target variable (class). After the data was transformed, I printed out a few examples to see the values of new attributes.

After preprocessing the dataset, I split the data into training and test sets. The function for splitting the data implemented in scikit-learn ensures that the distribution of target variable is similar to the distribution in the original dataset.

I also visualized the distribution of the target variable. It is clear from the figure 7.1 that the distribution of the classes is uneven. This indicates the imbalanced classes problem described in 3.5.

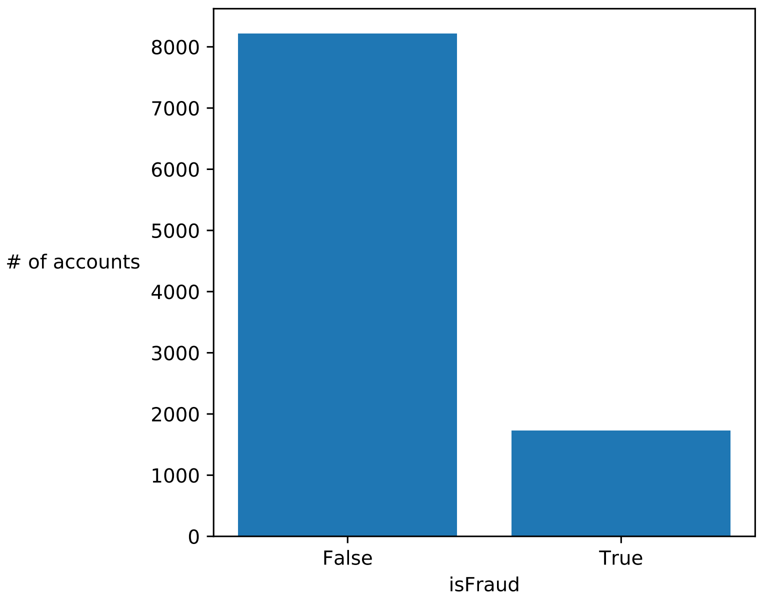

Figure 7.1: A visualization of the class imbalance

7.3 Baseline model

In section 5.2 I explain my decision to use decision tree (DT) as a baseline model. I used the scikit-learn implementation of DT for classification.

I used the training set for training the model, utilizing the scikit-learn implementation of the decision tree classifier. I selected the values of parameters for the DT with a use of the grid search cross-validation. The parameter that was tuned was splitting criterion and the possible values are gini and entropy and are described in 5.2.2. The parameters maximal depth and minimal samples in a leaf are restricted because they influence the size of the constructed tree. Restricting the size of the tree is important because smaller and simpler tree is easy to interpret and understand. The test set was used to evaluate the baseline model. Table 7.1 shows the confusion matrix constructed using the predictions for the test set produced by the trained classifier. Table 7.2 contains the overall accuracy of the baseline model, precision and recall for both classes.

It is apparent from the confusion matrix that the problem of class imbalance influenced the trained classifier. The classifier clearly favours the majority class. Although the overall accuracy is reasonably high and the number of false positives low, the number of undetected frauds is unacceptable.

Table 7.1: Confusion matrix for the baseline model

	Predicted positive	Predicted negative
Original positive	265	257
Original negative	90	2370

Table 7.2: Results of the baseline model

Measure	**Value**
Accuracy	88.36
Precision positive	74.85
Precision negative	90.22
Recall positive	50.77
Recall negative	96.34

7.4 Improved decision tree models

The evaluation of the baseline model indicates that adjustments to the baseline model are needed with regard to balancing the importance of each class.

7.4.1 Custom metric for parameter tuning

First of all I introduced a new metric to be used in cross-validation while tuning the parameters of the DT. The purpose of the custom metric is to provide a score that reflects the demands on the model. As it was stated, the model should produce low number of false positives while detecting the majority of frauds. The score, given by the metric to sample i with true label y_t and predicted label y_p, is defined as follows:

$$score_i = \begin{cases} -a & \text{if } y_t = 0 \ \& \ y_p \neq y_t \\ -b & \text{if } y_t = 1 \ \& \ y_p \neq y_t \\ 1 & \qquad \text{if } y_t = y_p \end{cases}$$

where a and b are the parameters that express how big the cost of misclassifying the given class is.

The new metric itself needed to be tuned to find the right balance between the misclassification costs. I created a set of 100 custom evaluation functions, each with different parameter settings ($(a, b) \in \{1, \dots, 10\} \times \{1, \dots, 10\}$). The tuning of the parameters was carried out by evaluating a set of randomly configured DTs with each metric. Then the numbers of false positives and false negatives produced by the models considered as best by each metric were visualized in figures 7.2 and 7.3. I decided to go with values of $a = 4$ and $b = 6$ which, according to the experiments, should provide a good balance between punishing false positives and missed frauds.

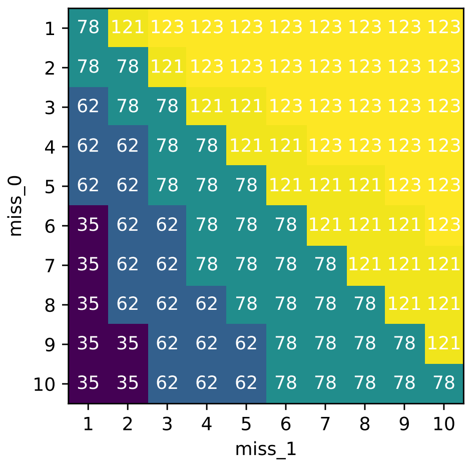

Figure 7.2: Heatmap with the number of false positives produced by models selected by different metrics

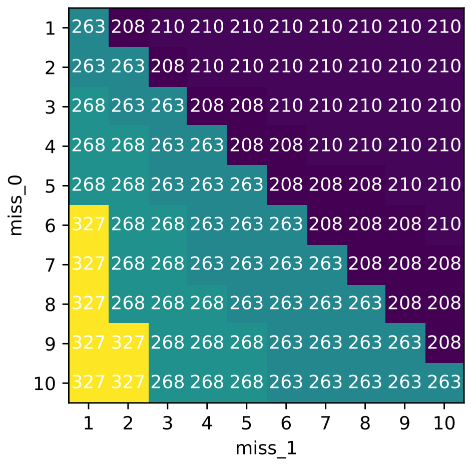

Figure 7.3: Heatmap with the number of false negatives produced by models selected by different metrics

7.4.2 Adjusting the class weights

As it is described in 5.3.2, introducing weights to the classes can help deal with the class imbalance problem as well as make the model cost-sensitive.

First I decided to balance the class importance. With this approach, each class is assigned a weight so that weighted sums of samples of a class are equal. For the binary classification problem with positive and negative classes, the weights w_0 and w_1 are assigned so that $w_0 N_0 = w_1 N_1$, where N_0, N_1 are numbers of respective classes.

Table 7.3: Results of the decision tree with balanced class weights

Measure	**Value**
Accuracy	84.00
Precision positive	52.87
Precision negative	95.13
Recall positive	79.50
Recall negative	84.96
FP	370
FN	107

As the numbers in the table 7.3 suggest, weighting the classes so that they are balanced, helped to detect more positive samples. Although the number of detected frauds increased, compared to the baseline model, the number of false positives increased as well. As it was established in 2.2.1, the number of false positives needs to be reduced. The previous facts imply that there will occur a trade-off between the number of false positives and true negatives.

The metric introduced and tuned in previous subsection is designed to select a model that provides a good balance between minimizing the number of false positives and detecting true negatives. The metric is used in cross-validation to select the class weight parameter.

Table 7.4: Results of the decision tree with class weights tuned using the custom metric

Measure	**Value**
Accuracy	88.90
Precision positive	69.94
Precision negative	92.53
Recall positive	64.18
Recall negative	94.15
FP	144
FN	187

The model with weights $w_1 = 3$ and $w_0 = 2$ for positive and negative class achieved the best results in the cross-validation. An evaluation of the models' performance is shown in table 7.4. It detected 64% of frauds while retaining acceptable level of false positives.

7.5 AdaCost

I implemented a prototype of the AdaCost algorithm described in 5.3.4. This algorithm ensembles a group of classifiers and uses the combination of their predictions to classify a sample. I decided to use decision trees with maximal depth $max\_depth = 4$ as the base predictors. The costs for missclassifying each class were selected as $c_0 = 4$ and $c_1 = 6$. The selection of the costs is based on the custom metric selected in section 7.4.1. I selected the costs to be the same as the weights in the metric because of how they reflect the demands on the fraud detection system.

An evaluation of the performance of this algorithm is in table 7.5. Although the accuracy is lower than the one of the decision tree with tuned class weights, the results are still acceptable.

Table 7.5: Results of the prototype of the AdaCost algorithm

Measure	**Value**
Accuracy	88.16
Precision positive	71.07
Precision negative	90.82
Recall positive	54.60
Recall negative	95.28
FP	116
FN	237

The drawback to this approach is that the decisions it makes cannot be interpreted as easily as in the case of decision tree algorithm. Discovering the feature importance is a method for partially uncovering the reasoning behind a decision. I utilized the feature importances returned by individual predictors and weighted them with the weights used when predicting.

7.6 Overall evaluation and comparison

Table 7.6 contains an evaluation of all the trained models. The tests were carried out using a test set consisting of 2982 entities, of which 522 was marked as suspicious. Balanced accuracy and weighted f-score are described in section 3.4 and are designed to fairly evaluate models trained on data with imbalanced class distribution. False positives are samples falsely classified as fraud and recall of the positive class is the percentage of detected frauds.

The baseline model produced a small number of false positives but it was due to the influence of the class imbalance, which resulted in bias towards the majority class. This is also confirmed by the recall of the positive class. The recall expresses that only a half of the fraudulent entities was detected.

Balancing the class weights eliminates the problem of class imbalance. It resulted in increasing the percentage of detected frauds to almost 80%, but it also increased the number of false positives. The number of false positives is now over 12% of all tested samples which means a lot of additional needless reports that is hard to process.

The usage of tuned class weights for a decision tree resulted in a good balance between the number of produced false positives and detected frauds. Model with tuned class weights detected 15% more fraudulent entities while retaining the number of false positives under 5%.

The prototype of the AdaCost algorithm shows promising results with an increase in percentage of detected frauds and keeping a low number of false positives when compared to the baseline model. There is a lot of parameters of the model that can be adjusted and potentially improve the performance of the model. Adjusting and designing these parameters requires further study of the algorithm and will be subject to future research. The drawback to this approach is the loss of clear interpretability of the decisions, substituted by only providing the feature importance.

Table 7.6: Results of all trained models, table shows balanced accuracy, weighted f-score, number of false positives and recall of the positive class

model	balanced acc.	w. f-score	FP	recall pos.
Baseline - DT	73.55	87.45	90	50.77
DT - balanced weights	82.23	85.16	370	79.50
DT - tuned weights	79.16	88.71	144	64.18
AdaCost	74.94	87.53	116	54.60

Conclusion

The first goal of this work was to survey existing solutions for detecting financial fraud. The following goals were to select, study and evaluate a baseline model based on the information gained in the previous survey, then survey and propose possible improvements to the baseline model, apply and evaluate them. The final goal was to compare the baseline model with the improved models.

I carried out the survey of solutions based on machine learning to the problem of detecting financial fraud. Based on the survey and demands on the fraud detection systems, especially the interpretability of the decisions, I chose a decision tree algorithm as the baseline model that can be trained on available data.

After a study of the selected algorithm I applied it on the available dataset. Then I evaluated the model and surveyed possible extensions to the model that could improve its performance, mainly balancing the trade-off between false positives and undetected frauds.

The proposed improvements were custom metric that reflects the demands on the system and helps to select class weights. Weighting the classes is another extension to the decision tree, which introduces cost-sensitivity. The final improvement was to ensemble multiple decision trees using cost-sensitive boosting.

The proposed extensions were applied and evaluated in the experimental part. The results of the evaluation show that introducing cost-sensitivity helps to get a reasonable trade-off between false positives and undetected frauds. The prototype of the AdaCost algorithm produced promising results and a small improvement when compared with the baseline model.

Further study of parameters for AdaCost algorithm and ensuring compatibility of the implemented prototype with components from the scikit-learn is a subject to future work. Future work could also be based on a cooperation with a financial institution, that would provide a real dataset for more realistic evaluation of the models. The models could also be compared with solutions used by the institution to discover new shortcomings. New improvements and adjustments could be proposed to eliminate the shortcomings and make the model ready for production use.

Bibliography

[1] LEVI, Michael and REUTER, Peter; Money Laundering; *Crime and Justice* [online]; 2006, 34.1: 289-375 [accessed on 28.03.2019]; doi:10.1086/501508; Available from: https://www.journals.uchicago.edu/doi/full/10.1086/501508.

[2] Investopedia; What methods are used to launder money?; *Investopedia* [online]; 20.02.2015 [accessed on 28.03.2019]; Available from: https://www.investopedia.com/ask/answers/022015/what-methods-are-used-launder-money.asp.

[3] KENTON, Will; Smurf; *Investopedia* [online]; 20.03.2019 [accessed on 28.03.2019]; Available from: https://www.investopedia.com/terms/s/smurf.asp.

[4] Czech National Bank; Money Laundering; *Czech National Bank* [online]; ©2019 [accessed on 28.03.2019]; Available from: https://www.cnb.cz/en/supervision-financial-market/legislation/money-laundering/.

[5] National Crime Agency; Suspicious Activity Reports; *National Crime Agency* [online]; [accessed on 28.03.2019]; Available from: https://www.nationalcrimeagency.gov.uk/what-we-do/crime-threats/money-laundering-and-terrorist-financing/suspicious-activity-reports.

[6] MALDONADO LOPEZ, Juan Pablo; Supervisor of this thesis; Prague 30.04.2019.

[7] MITCHELL, Tom M.; *Machine Learning*; London: McGraw-Hill Pub. Co., 1997; ISBN 9780071154673.

[8] WITTEN, Ian H. and FRANK, Eibe; *Data Mining: Practical machine learning tools and techniques*; Elsevier Science, 2005; ISBN 9780080477022.

[9] Amazon Web Services, Inc. or its affiliates; Training ML Models; *Amazon Web Services (AWS) - Cloud Computing Services* [online]; ©2019; [accessed on 30.03.2019]; Available from: `https://docs.aws.amazon.com/machine-learning/latest/dg/training-ml-models.html`.

[10] BROWNLEE, Jason; Supervised and Unsupervised Machine Learning Algorithms; *Machine Learning Mastery* [online]; 16.03.2016; [accessed on 30.03.2019]; Available from: `https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms`.

[11] ALPAYDIN, Ethem; *Introduction to Machine Learning*; MIT Press, 2009; ISBN 9780262303262.

[12] Smart Vision Europe; What is the CRISP-DM methodology?; *Smart Vision Europe* [online]; ©2018; [accessed on 05.04.2019]; Available from: `https://www.sv-europe.com/crisp-dm-methodology/`.

[13] KOTSIANTIS, Sotiris B., KANELLOPOULOS, Dimitris, PINTELAS, Panagiotis E.; Data preprocessing for supervised leaning; *International Journal of Computer Science*, 2006, 1.2: 111-117.

[14] Elite Data Science; Feature Engineering; *Elite Data Science* [online]; ©2016-2018; [accessed on 03.04.2019]; Available from: `https://elitedatascience.com/feature-engineering`.

[15] scikit-learn developers; sklearn.metric.balanced_accuracy_score; *scikit-learn* [online]; ©2007-2019; [accessed on 15.04.2019]; Available from: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn.metrics.balanced_accuracy_score`.

[16] SASAKI, Yutaka, et al.; The truth of the F-measure; *Teach Tutor mater* [online], 2007, 1.5: 1-5 [accessed on 15.04.2019]; Available from: `https://www.cs.odu.edu/~mukka/cs795sum10dm/Lecturenotes/Day3`.

[17] scikit-learn developers; sklearn.metrics.f1_score; *scikit-learn* [online]; ©2007-2019; [accessed on 15.04.2019]; Available from: `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score`.

[18] SUN, Yanmin, KAMEL, Mohamed S., WANG, Yang; Boosting for learning multiple classes with imbalanced class distribution; In: *Sixth International Conference on Data Mining (ICDM'06)* [online]; IEEE,

2006. p. 592-602 [accessed on 10.04.2019]; Available from: http://people.ee.duke.edu/~lcarin/ImbalancedClassDistribution.pdf.

[19] IBM; AMLSim; *GitHub* [online]; updated on: 23.01.2019; [accessed on: 02.03.2019]; Available from: https://github.com/IBM/AMLSim#amlsim.

[20] LOPEZ-ROJAS, Edgar, ELMIR, Ahmad, AXELSSON, Stefan; PaySim: A financial mobile money simulator for fraud detection; In: *28th European Modeling and Simulation Symposium, EMSS, Larnaca*; Dime University of Genoa, 2016; p. 249-255.

[21] CHEN, Zhiyuan et al.; Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review; *Knowledge and Information Systems* [online]; 2018, 57.2: 245-285 [accessed on 18.02.2019]; ISSN 0219-3116; Available from: https://link.springer.com/article/10.1007/s10115-017-1144-z.

[22] CHEN, Yu-To, MATHE, Johan; Fuzzy computing applications for anti-money laundering and distributed storage system load monitoring [online]; 2011 [accessed on 10.04.2019]; Available from: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/37118.pdf.

[23] KEYAN, Liu, TINGTING, Yu; An improved support-vector network model for anti-money laundering; In: *2011 Fifth International Conference on Management of e-Commerce and e-Government*; IEEE, 2011; p. 193-196.

[24] DREŻEWSKI, Rafał, SEPIELAK, Jan, FILIPKOWSKI, Wojciech; System supporting money laundering detection; *Digital Investigation* [online]; 2012, 9.1: 8-21; [accessed on 10.04.2019]; Available from: https://www.sciencedirect.com/science/article/pii/S1742287612000230.

[25] WANG, Su-Nan, YANG, Jian-Gang; A money laundering risk evaluation method based on decision tree; In: *2007 International Conference on Machine Learning and Cybernetics* [online]; IEEE, 2007; p. 283-286; [accessed on 10.04.2019]; Available from: https://ieeexplore.ieee.org/abstract/document/4370155.

[26] LIU, Xuan, ZHANG, Pengzhu, ZENG, Dajun; Sequence matching for suspicious activity detection in anti-money laundering; In: *International Conference on Intelligence and Security Informatics*; Springer, Berlin, Heidelberg, 2008; p. 50-61.

[27] BREIMAN, Leo, et al.; Classification and Regression Trees; 1984.

[28] scikit-learn developers; Decision Trees; *scikit-learn* [online]; ©2007-2019; [accessed on 01.05.2019]; Available from: `https://scikit-learn.org/stable/modules/tree.html`.

[29] scikit-learn developers; sklearn.model_selection.GridSearchCV; *scikit-learn* [online]; ©2007-2019; [accessed on 02.05.2019]; Available from: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.GridSearchCV`.

[30] JAPKOWICZ, Nathalie; The class imbalance problem: Significance and strategies; In: *Proc. of the Int'l Conf. on Artificial Intelligence* [online]; 2000; [accessed on 20.04.2019]; Available from: `https://pdfs.semanticscholar.org/907b/02c6322d0e7dff6b0201b03e3d2c6bc1d38f.pdf`.

[31] FREUND, Yoav, SCHAPIRE, Robert E.; A decision-theoretic generalization of on-line learning and an application to boosting; *Journal of computer and system sciences*, 1997, 55.1: 119-139.

[32] FAN, Wei, et al.; AdaCost: misclassification cost-sensitive boosting; In: *Icml*; 1999; p. 97-105.

[33] Python Software Foundation; *Python* [online]; ©2001-2019; [accessed on 30.11.2019]; Available from: `https://www.python.org/about/`.

[34] Project Jupyter; About Us; *Project Jupyter* [online]; ©2019; [accessed on 13.05.2019]; Available from: `https://jupyter.org/about`.

[35] PyData; Package overview; *pandas* [online]; [accessed on 02.04.2019]; Available from: `http://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html`.

[36] The Matplotlib development team; *Matplotlib: Python plotting* [online]; ©2012 - 2018; [accessed on 03.05.2019]; Available from: `https://matplotlib.org/index.html`.

[37] scikit-learn developers; *scikit-learn: machine learning in Python* [online]; ©2007-2019; [accessed on 02.04.2019]; Available from: `https://scikit-learn.org/stable/index.html`.

[38] NumPy developers; *NumPy* [online]; ©2019; [accessed on 03.04.2019]; Available from: `https://www.numpy.org/`.

Appendix A

Glossary

AML Anti money laundering

CART Classification and Regression Trees

CSV Comma separated values

CRISP-DM Cross-Industry Standard Process for Data Mining

DT Decision tree

FNs False negatives

FPs False positives

ML Machine learning

SAR Suspicious activity report

TNs True negatives

TPs True positives

Appendix B

Content of the enclosed CD

```
readme.txt ................... Brief description of the content of the CD
 └─ src ..................... Directory with sources for the practical part
     └─ html ........ Directory contains notebooks with outputs in HTML
     │   ├─ data_analysis.html .......................................
     │   ├─ data_preprocessing.html ..................................
     │   ├─ baseline_model.html ......................................
     │   ├─ custom_metric_tuning.html ................................
     │   ├─ dt_cost_sensitivity.html .................................
     │   ├─ model_ensemble_AdaCost.html ..............................
     │   └─ evaluation.html ..........................................
     └─ jupyter_notebooks ....... Directory containing Jupyter notebooks
         ├─ data ......................... Directory contains the datasets
         ├─ models .................... Directory contains trained models
         ├─ graphics ................... Directory contains visualizations
         ├─ data_analysis.ipynb ......................................
         ├─ data_preprocessing.ipynb .................................
         ├─ baseline_model.ipynb .....................................
         ├─ custom_metric_tuning.ipynb ...............................
         ├─ dt_cost_sensitivity.ipynb ................................
         ├─ model_ensemble_AdaCost.ipynb .............................
         └─ evaluation.ipynb .........................................
└─ written ............................... The written part of the thesis
    ├─ thesis .................... The source of the thesis in LaTeX format
    └─ thesis.pdf ....................... Text of the thesis in PDF format
```