UNIVERSITY OF TARTU
Faculty of Social Sciences
School of Economics and Business Administration

Kseniia Kasianova

DETECTING MONEY LAUNDERING USING HIDDEN MARKOV MODEL

Master's thesis

Supervisor: Kaur Lumiste (PhD)

Tartu 2020

Name and signature of supervisor Kaur Lumiste …….

Allowed for defense on 04.06.2020

(date)

I have written this master's thesis independently. All viewpoints of other authors, literary sources and data from elsewhere used for writing this paper have been referenced.

……..

(signature of author)

Abstract

Recent money laundering scandals, like the Danske Bank and Swedbank's failure to mitigate money laundering risks (Kim, 2019), have made "anti money laundering" (AML) a much discussed topic. Governments are making AML regulations tougher and financial institutions are struggling to comply, one of the requirements is to actively monitor financial transactions to detect suspicious ones.

Most of the financial industry applies simple rule-based methods for monitoring. This thesis provides a practical model to detect suspicious transactions using the hidden Markov model (HMM). The use of HMM is justified, because the criminal nature of a transaction is hidden to the financial institution, only transaction parameters can be observed. By using past data, a model is built to detect if current transaction is suspicious or not. The model is assessed with artificial and real transactions data. It was concluded that this model performs better than a classical k-means clustering algorithm.

Keywords: money laundering, hidden Markov model, transaction monitoring, k-means clustering.

Content

1 Introduction

The problem of money laundering has become more and more crucial in the last years. Big scandals like in the Danske Bank (RiskScreen, 2019) and the Wachovia Bank (Wyler, 2011) money laundering cases increased interest in this topic. Money laundering is a process that takes illegally obtained finances and puts it through a cycle of transactions and various accounts in a bank (or between banks) in order for it to appear to be from a legitimate source. The general idea is to hide traces of the illegal money so that it would not lead back to its source. Hiding legitimately acquired money to avoid taxation also qualifies as money laundering.

According to the Financial Action Task Force, large sums of money are laundered every year, posing a threat to the global economy and its security (FATF, 2020). There are various ways to launder money, for example, usage of shell companies for fictitious business activity; usage of fraudulent record keeping; and the purchase of real estate using cash. Such activities can occur globally. The sophistication of money laundering activities depends on the transactions' sequence that is used to hide the relationship between dirty money and its origin.

To deal with this problem, anti-money laundering (AML) detection procedures and requirements are forced upon financial institutions by different governmental institutions. For example, in Estonia gathering information about suspicions of money laundering or terrorist financing is the aim of the Financial Intelligence Unit (FIU), an independent structural unit of the Estonian Police and Border Guard Board (Estonian Police and Border Guard Board, 2020). The Prosecutor's Office, Security Police, Tax and Customs Board and courts in Estonia take over from FIU to identify the criminal activities and enforce Estonian AML laws. The laws and regulations are overseen by the Ministry of Finance in (Estonian Ministry of Finance, 2019). Many governments abide by recommendations or directives issued by international organisations. Examples of such sources of AML regulations are The Financial Action Task Force's Recommendations (FATF, 2020) and European Union's Fourth and Fifth Anti-Money Laundering Directives (European Commission, 2019).

It can be said that two parties are involved in AML process:

Government

- implement measures to prevent or mitigate money laundering;
- implement laws and regulations;
- criminalize money laundering;
- confiscate funds;
- coordinate internationally.

Financial institutions

- implement measures to prevent or mitigate money laundering;
- implement policies;
- record keeping;
- report suspicious activity;
- coordinate with law enforcement.

In order to be compliant with regulations, financial institutions have to implement measures to prevent or mitigate money laundering, this includes risk assessment, customer due diligence and actively try to detect money laundering. One of the practical implications of these efforts is transaction monitoring, where a set of financial institutions' customers' transactions are evaluated by different rules and labelled as *suspicious* or *normal*. Many financial institutions apply rule-based monitoring systems - if a set of fine-tuned thresholds are exceeded or certain parameters are triggered, then a transaction is investigated further. There are some examples of state-of-the-art methods being used, like machine learning methods and artificial intelligence (Comply Advantage, 2019; TransferWise, 2019). Current thesis proposes a novel approach for detecting suspicious transactions.

The problem with rule-based approach is that it might create a lot of false alerts, which will take AML specialists a lot of time to resolve. Sometimes really suspicious cases are even missed. So, the improvement of a model for detecting money laundering is crucial in the modern world.

In this thesis, the hidden Markov model (HMM) was considered to detect transactions, which could be connected to money laundering. This method is suitable by the very definition of the method. The type of transaction (suspicious or not) was assumed as a *hidden* (unobservable) random variable, that is dependent only on the previous value (Markov property). Different transactions' characteristics (e.g. time of transaction, currency, directions, amount, number of transactions in the last 7 days and so on) were

used to define an *observable* variable. Based on the observable variable HMM predicts the hidden state using defined probabilities between them.

One difference between HMM and rule-based algorithms is that in HMM, state of current transaction depends on the state of previous transaction, but in rule-based system states of transactions are independent. Moreover, in HMM we can use the same rules, but as a combination of them, not separately as in rule-based systems. The other difference is that HMM provides a probabilistic approach. Besides, HMM is more personalized than usual rule-based system.

In short, the goal of this thesis is to introduce HMM, build a model to detect suspicious transactions and test this model on artificial and real data. To check if this novel model provides better results, the performance of HMM is compared to classical k-means clustering. Clustering is a basic algorithm for anomaly detection and k-means clustering method is the most widely used one.

The thesis is organized as follows: Section 2 reviews general methods for anomaly detection, methods to detect money laundering in a form of suspicious events and different applications of the hidden Markov model. Section 3 focuses on methodological issues – introduces HMM and algorithms that are needed for practical application, and k-means clustering. In Section 4 describes the empirical study and Section 5 discusses the results followed by conclusions. The empirical study was carried out in R software (R Core Team, 2018) with packages *data.table, HMM.*

2 Literature review

2.1 Introduction to money laundering

According to Madigner (2011) money laundering is a process of making dirty money appear to be clean, but laundered funds are never totally clean. Cox (2012) noted that money laundering refers both to the use of a cash business such as a launderette to facilitate the mingling of legal and illegal funds, and also to the generic process of disguising the original proceeds of the funds - a process more normally referred to as layering.

Usually, the process of money laundering is divided into three stages:

- placement: money is transferred into the system without banks or authorities recognizing it;

- layering: money is concealed within multiple layers of transactions;

- integration: seemingly clean money is steered into the economy.

Cox (2012, p. 15) described such stages as: "The initial proceeds enter the banking system at a perceived point of weakness (the placement phase) and then the funds are moved around such that the initial source of the funds is disguised (the layering phase). The funds are eventually reintegrated into the mainstream banking system as clean funds (the integration phase)."

According to the Financial Action Task Force (FATF, 2020), there are three primary methods of laundering money:

1. Via financial institutions and nonbank financial institutions. This includes the placement and structuring of deposits of tainted money into banks, wiring or layering the dirty money to multiple accounts in multiple banks in multiple jurisdictions to confuse the paper trail, and then using the laundered money by integrating it into the economy by way of purchasing high-value properties and goods.

2. Bulk cash smuggling is the physical smuggling of illicit cash from one jurisdiction to another where it will be more readily accepted for deposit.

3. Trade-based money laundering. This also includes underground financial systems, because historically and culturally most are settled on the misuse of international trade, including customs fraud.

Here we focus on the first point – anti-money laundering (AML) detection procedures and solutions for suspicious transactions in financial institutions, e.g. retail or commercial banks. AML refers to a set of laws, regulations, and procedures intended to prevent criminals from disguising illegally obtained funds as legitimate income. Financial institutions are required to monitor customers' transactions and report on anything suspicious. AML solutions, being part of the overall fraud control, automate and help to reduce the manual work of a screening/checking process.

Usually, banks are controlled by different AML compliance organizations. The most important AML regulations are established in a such sources:

- The Financial Action Task Force's (FATF) Recommendations;
- The United States' Bank Secrecy Act (BSA);
- European Union's Fourth and Fifth Anti-Money Laundering Directives;
- Hong Kong Monetary Authority's Guideline on Anti-Money Laundering
- Counter-Financing of Terrorism and Monetary Authority of Singapore's Notices on the Prevention of Money Laundering and Countering the Financing of Terrorism.

The main rule of all these documents, is that banks and other financial institutions are obligated to establish AML compliance programs with internal controls. For example, the FATF recommendations require carrying out due diligence procedures when transaction amount exceeds 15000 USD/EUR or international transactions exceeding 1000 USD/EUR, while BSA requires financial institutions to report every transaction in the sum of 10000 USD or more to the US authorities.

The topic of AML is important and actual as recent media coverage has shown (e.g. the Danske Bank money laundering scandal (RiskScreen, 2019), Swedbank's failure on AML controls (Kim, 2019)). Because of this, there is ample literature that provide analysis on the topic and many different software solutions have been developed in recent years (Ayasdi AML (Ayasdi.com 2020), Guardian analytics (GuardianAnalytics.com 2020), SAS Anti-Money Laundering (SAS.com 2020), Comply Advantage (Comply Advantage, 2019) etc).

2.2 Overview of general methods for anomaly detection

One of the practical implications of money laundering is transaction monitoring, where a set of financial institution's customer transactions are evaluated by different rules and labelled as *suspicious* or *normal*. For example, a person could be labelled as

suspicious if he has more than 10 transactions to different persons with the amount of every transaction bigger than 1000 EUR. Then suspicious transactions can be suspended and are investigated in more detail, e.g. financial institutions may ask the customer to confirm the legality of income. Suspicious transactions are grouped in a suspicious activity report which financial institutions should send to a committee or governmental organisation (e.g. in Estonia to the Financial Intelligence Unit).

In general, the methods to detect money laundering are part of a family of methods to detect anomalies — seeking out patterns in data which are not expected according to previous behaviour.

A big comparative analysis of different methods for anomaly detection in various areas was done by Chandola et al. (2009). Their analysis includes neural networks-based methods, Bayesian networks, support vector machines-based methods, rule-based methods, nearest neighbour analysis, clustering, statistical anomaly detection techniques, information theoretic techniques and spectral analysis. For each category of anomaly detection techniques a unique assumption was identified, regarding the notion of normal and suspicious data.

Another overview of network anomaly detection techniques was provided by Ahmed et al. (2016). They described such techniques as clustering, classification, information theory and statistics. Authors concluded that clustering and information theory-based techniques are better than others. Clustering techniques are computationally efficient and have specific targets for denial of service (DoS) attack detection, while the information theory-based techniques have no specific attack target.

One of the closest areas to money laundering detection is detecting credit-card fraud. Credit-card fraud occurs when criminals steal credit cards or use a lost one for online or offline payments. The main idea behind these related methods is that there is a sharp change of activity from normal to suspicious. Such methods could be found in Maes et al. (2002), Perols (2011), Sahin et al. (2013), and Awoyemi et al. (2017).

Maes et al. (2002) used artificial neural networks and Bayesian belief networks to detect credit-card fraud. As a result, they found that Bayesian belief networks give better accuracy and results in much smaller learning and executing time compared to artificial neural networks.

Perols (2011) analysed the quality of logistic regression and support vector machine methods compared to artificial neural network and stacking, and found the first ones are also good.

In the study of Sahin et al. (2013), a new cost-sensitive decision tree approach was developed. The performance of cost-sensitive decision tree approach was compared with more commonly used classification models on a real data set of credit card transactions. As a conclusion, such cost-sensitive decision tree algorithms perform better than the other existing methods for detecting credit card fraud.

Awoyemi et al. (2017) investigated the performance of naive Bayes, k-nearest neighbour and logistic regression on highly skewed credit card fraud data. The comparative results show that k-nearest neighbour method performs better than naive Bayes and logistic regression techniques.

2.3 Methods to detect money laundering in a form of suspicious events

The literature in money laundering detection research can be divided into two parts: theoretical evaluation of different methods and practical implications, mostly on artificial data. It is also worth mentioning that it is impossible to detect money laundering directly, only some suspicious activities, which could be a part of the money laundering process. Moreover, only law enforcement organisations have the rights to conduct a full investigation and make a case if money laundering actually happened.

Chen et al. (2018) provided an overview of different machine learning techniques for anti-money laundering including supervised and unsupervised techniques. They evaluated fuzzy rules, frequent pattern algorithm, support vector machine, radial basis function, some clustering and hybrid approaches, few methods of link analysis and behavioural modelling. Moreover, risk scoring methods and anomaly detection algorithms also were presented in that paper. Based on the findings from existing literature reviews, they found that most methods use similar attributes such as the amount received, the amount withdrawn, and the debit/credit transaction frequency within certain time windows, e.g. daily, weekly, and monthly. However, some authors used additional attributes such as risk value, the individuals' salary information, and the senders/receivers individual account history as part of their methods.

Chen et al. (2018) mentioned that it is also crucial to remember, that there are such problems as real data insufficiency to test the effectiveness of methods and the detection of suspicious transactions only relies on discovered transaction. The effectiveness of a machine learning algorithm is largely influenced by the unique characteristics of financial money laundering data. It is, therefore, crucial to understand the strengths and the limitations of each algorithm when applied to money laundering problems.

Different authors provide various classifications on AML methods. For example, Rohit and Patel (2015) divided AML methods into such groups: rule-based approach, clustering-based approach, classification-based approach and model-based approach. According to their analyses, it can be concluded that: 1) An artificial intelligence approach for AML starting to replace rule-based AML systems; 2) Unsupervised learning with a small set of training data is suitable enough for building data mining based solutions for AML.

Cao and Do (2012) combined data mining techniques and human's analyst ability to create an effective method to detect money laundering in Vietnam. They used the clustering algorithm CLOPE invented by Yiling Yang, Xudong Guan and Jinyuan You (2002). The main idea of the algorithm is based on the realistic situation from real-life data in string data type. They concluded that CLOPE is a suitable algorithm for money laundering detection. But the system cannot run standalone absolutely, it must be based on the ability of analysts in analysing data and providing a set of rules (criteria set) to validate clusters after clustering.

Umadevi and Divya (2012) proposed Transaction Flow Analysis (TFA) System to detect money laundering, which includes clustering and frequent pattern mining. There are few main parts of this TFA system such as: bank transaction importer; application of detecting money laundering algorithms (frequent pattern and transaction mining algorithms), which executes distributive box and collective box; transaction clustering and detecting suspicious clusters using laundering methods and roles of the offender. Finally received frequent patterns and clusters can be visualized in schema and timeline diagrams. It helps AML specialists to find suspicious transactions. But this system was tested only with a small generated amount of data - 100 accounts.

Finally, it could be concluded that there is no one specific method which will solve the problem of money laundering in all cases, so it is crucial to continue investigation of other algorithms.

2.4 Hidden Markov model and its application

Despite the extensive list of methods, there are other ways to determine money-laundering, which are not so widely used, but still can provide good results. For future analysis hidden Markov model was chosen.

Markov chain is a type of random process, where the probability of being in some state depends only on the previous state, and not on what happened before. Hidden Markov models are a class of probabilistic models that allow us to predict a sequence of unknown (hidden) variables from a set of observed variables. So, it is a statistical model in which the modelled system is expected to be like a Markov process with some unobserved states. In general, the HMM is defined by 5 different components: states, state probabilities, initial probabilities, transition probabilities and emission probabilities.

HMM can be summarized as a double stochastic process with the two following aspects:

1. The first process is a finite set of states, where each of them is generally associated with multidimensional probability distribution. The transition between the different states is statistically organized by a set of probabilities called transition probabilities.

2. In the second process, in any state, an event can be observed. That means we observe and analyse the event without knowing which state generated it. So, the states are called “hidden” as they are hidden to the observer.

HMM is mostly used in such problems as recognition of speech (Varga and Moore, 1990, Schuller et al., 2003), handwriting (Chen et al., 1994), gesture recognition (Yang and Xu, 1994), part-of-speech tagging, musical score following (Raphael, 1999), partial discharges and bioinformatics, but this method is also used for detecting credit card fraud (e.g. Li et al., 2009, Mhamane and Lobo, 2012, Singh and Pandit, 2015) . In the case of money laundering, the unobserved state is a type of transaction (*suspicious* or *normal*).

By Stamp (2004) hidden Markov model can be defined as a general version of the mixed model with the hidden variables, which control the mixture element to be selected

for each observation. Moreover, these hidden (latent) variables are related through a Markov process, which means that they depend on one previous value.

Jadhav and Bhandari (2013) pointed that they received a comparatively small number of False Positives (transactions classified as suspicious, but they are not) from their HMM based fraud detection system. This was especially noticeable in case of real life transactions.

Li et al. (2009) used HMM and genetic algorithms to detect hidden group members in financial transaction networks. Hidden group detection problem was solved by maximum likelihood approach which consisted of the finding the hidden group with the maximal likelihood of transactions, which were observed. In order to do a simulation experiment, the group structure of the synthetic financial network was generated. Different relations put various effects on the group structure, so the authors consider two classes of relations: field relation and transaction relation. As a result, it was concluded that the genetic algorithm is efficient for this specific optimization problem. But this paper tests their model only on a small amount of artificial data and the effectiveness of the proposed method decreases with bigger network size.

Mhamane and Lobo (2012) explained how Internet banking fraud could be detected using HMM. By their definition, a fraudulent user is the unauthorized user who does not a legal Internet banking account in a bank; who makes use of authorized users' Internet banking accounts to do transaction. He obtains the password of a customer by conducting a cyber-attack. Authors considered the unobserved state as a kind of purchase (travel ticket, movie ticket or book purchase). They proposed a model which firstly should be trained and then it will be in the detection and prevention phase, where the system looks for the deviation in the expected and actual outcome and fraud is recognized. To calculate HMM parameters (state and transition probabilities) authors used Baum-Welch algorithm. The paper is mostly theoretical, with no empirical simulation or application.

Singh and Pandit (2015) provided a system, which firstly creates the behavioural pattern of all user using HMM. Afterwards, if the transaction is not accepted by the given model then they consider it as a security threat or fraud and send an alert to the user to verify.

3 Methodology

In this part, an overview of theoretical methods is provided. Firstly, the basis of hidden Markov model (HMM) is introduced - Markov chains. Secondly, HMM is explained using equations, an example, and charts. As an integral part of applying HMM in practice, Viterbi algorithm and Baum-Welch algorithm are presented. Thirdly, a benchmark method – the k-means clustering is explained. Finally, evaluation methods are given in order to compare HMM and k-means clustering results.

3.1 Hidden Markov model

To give a full overview of HMM, few topics should be mentioned: Markov chain, Viterbi algorithm and Baum-Welch algorithm. This section relies heavily on HMM theory given in Rabiner (1989).

3.1.1 Markov chain

The base of the HMM is formed by Markov chains. A Markov chain is a type of random process, where the probability of being in some state depends only on the previous state, and not on what happened before that. Let us assume here that time is discrete, meaning that measurements are done at certain time points. Then, formally a discrete Markov chain has the following components and properties.

Let $\boldsymbol{S} = \{S_1, S_2, \dots, S_N\}$ be a set of N possible hidden states in the model. The process starts from a random state, the distribution of the initial states is given by $\pi = \{\pi_1, \pi_2, \dots, \pi_N\}$, where π_i is the probability that the Markov chain will start in state S_i. If state S_j has $\pi_j = 0$, then that means state S_j cannot be an initial state of the random process. Since π is a distribution, then always $\sum_{i=1}^{n} \pi_i = 1$.

The Markov chain is initiated and starts to "wander". We denote the time instants associated with state changes as $t = 1, 2, 3, \dots$, and denote the actual realised state at time t as q_t. The Markov property, that the actual state q_t, at time t, is only dependent on the previous state q_{t-1}, is expressed as

$$P\left(q_t = S_j \middle| q_{t-1} = S_i, q_{t-2} = S_k, \dots, q_1 = S_l\right) = P\left(q_t = S_j \middle| q_{t-1} = S_i\right) \quad (1)$$

Probabilities of moving from one state to the other are given by transition probability matrix A:

$$A = \{a_{ij}\} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{pmatrix}, \quad (2)$$

where $a_{ij} = P(q_t = S_j | q_{t-1} = S_i)$ is a probability of moving from state i to state j. Probabilities a_{ij} have the properties: $a_{ij} \geq 0$; and $\sum_{j=1}^{N} a_{ij} = 1, \forall i$, meaning that row-probabilities sum up to 1.

Here it is assumed, that the random process produces observable states q_t at each instant of time t, but what if the true state is hidden and only some indications of the hidden state are revealed? For example, the fact if a financial transaction is of illicit nature, is hidden for the financial institution. Only the customer, receiver and transactional data – parameters of the transaction – are observable.

3.1.2 Definition of hidden Markov model

A hidden Markov model unites both observed events (parameters of transaction) and hidden events (type of transactions – *suspicious* or *normal*) that we think of as causal factors in our probabilistic model. Let us denote the set of M possible observed events as $V = \{v_1, v_2, \dots, v_M\}$ that hidden states S produce, often V is refered to as alphabet. The HMM starts to "wander", the hidden sequence of $Q = q_1, q_2, \dots, q_T$ emits a sequence of T observations $O = O_1, O_2, .., O_T$. Moreover, we need likelihoods (emission probabilities) that the hidden state at time t, $q_t = S_i$, produced an observed event $O_t = v_k$, i.e.

$$B = \{b_i(v_k)\} = \begin{pmatrix} b_1(v_1) & b_1(v_2) & \cdots & b_1(v_M) \\ b_2(v_1) & b_2(v_2) & \cdots & b_2(v_M) \\ \vdots & \vdots & \ddots & \vdots \\ b_N(v_1) & b_N(v_2) & \cdots & b_N(v_M) \end{pmatrix}, \quad (3)$$

where $b_i(v_k) = P(O_t = v_k | q_t = S_i)$. Matrix B is called an emission probability matrix and specifies the observation event probability distribution. Note tha $\sum_{k=1}^{M} b_i(v_k) = 1, \forall i$, meaning that row probabilities sum up to 1.

Besides the Markov property, HMM has another assumption: the probability of an output observation O_i is defined only by the state that produced the observation q_i and not by any other states or any other observations (so called output independence property):

$$P(O_i | Q, \mathrm{O}) = P(O_i | q_i). \quad (4)$$

Generally, three main problems of hidden Markov model are defined according to Rabiner (1989):

- Likelihood: Using such parameters of HMM as $\lambda = (A, B)$ and given observation sequence O, determine the likelihood $P(O|\lambda)$.
- Decoding: Using such parameters of HMM as $\lambda = (A, B)$ and given observation sequence O, find the best sequence of hidden states Q.
- Learning: Using given observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

In this work we are interested in decoding for evaluating this method in general, but first an example of a HMM is provided.

So the general idea is that we don't know (cannot observe) state q_t for time t, but we can observe some other variable O_t that takes some value v_k based on the state q_t. A set of parameters $\lambda = (\pi, A, B)$ completely specifies a HMM, where A and B are defined by (2) and (3), correspondingly. A graphic depiction of a HMM is given in Figure 1.

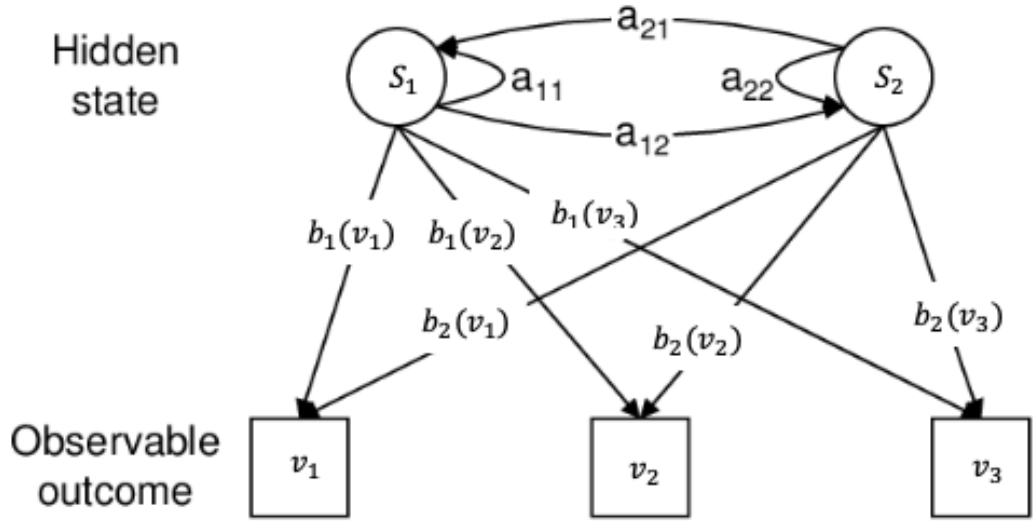

Figure 1. General view of HMM, where S_1 and S_2 are hidden states, v_1, v_2, v_3 are observable states, $\{a_{ij}\}$ are transition probabilities, $b_i(v_k)$ are emission probabilites. (Source: author's depiction)

The HMM in Figure 1 generates a sequence of hidden states $Q = q_1, q_2, \dots, q_T$ and for each a corresponding observations, i.e. $O = O_1, O_2, .., O_T$, where each observation O_i is one element from V.

3.1.3 Example of a HMM

A simple example of a HMM is a situation between two friends, Alice and Bob, who live in different cities, but talk every evening and Bob always says what he did this day – stay at home or have a walk. The choice of this activity is fully determined by weather on a given day, which could be sunny or rainy. Alice does not know weather for sure, but she knows general trends in Bob's behaviour. Every day Alice tries to guess what the weather was, based on what Bob tells her he did each day.

Alice cannot observe weather, so it is a hidden variable for her. Instead she knows Bob's activity every day, which is the observed variable for Alice. In our terms $S = \{S_1 = "rainy"; S_2 = "sunny"\}$ and possible values for $V = \{"stay", "walk"\}$. The girl also knows some probabilities. First of all, she assumes that on the very first day the probability of rain is 0.6 and the probability of sun is 0.4. These are initial probabilities: $\pi = \{\pi_1 = 0.6; \pi_2 = 0.4\}$.

The weather of every next day depends only on the weather of one previous day, which is a property of the Markov chain. Alice knows these probabilities between two weather states: if today is rainy, then tomorrow will be rainy with the probability 0.7 and sunny with the probability 0.3; if today is sunny, then tomorrow will be rainy with the probability 0.4 and sunny with the probability 0.6. These probabilities are our transition probabilities:

$$A = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}.$$

Alice knows Bob's behaviour: if it is rainy, he will stay home with the probability 0.8 and go for a walk with the probability 0.2. If the day is sunny, then Bob will stay home with the probability 0.4 and go for a walk with the probability 0.6. Formally, these are our emission probabilities:

$$B = \begin{pmatrix} b_1(stay) & b_1(walk) \\ b_2(stay) & b_2(walk) \end{pmatrix} = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}.$$

So knowing all these probabilities, the weather yesterday and Bob's activity today, Alice will predict what the weather was today using the model on Figure 2.

For initializing HMM in the empirical study, R function *initHMM* from package *HMM* was used (see Appendix D).

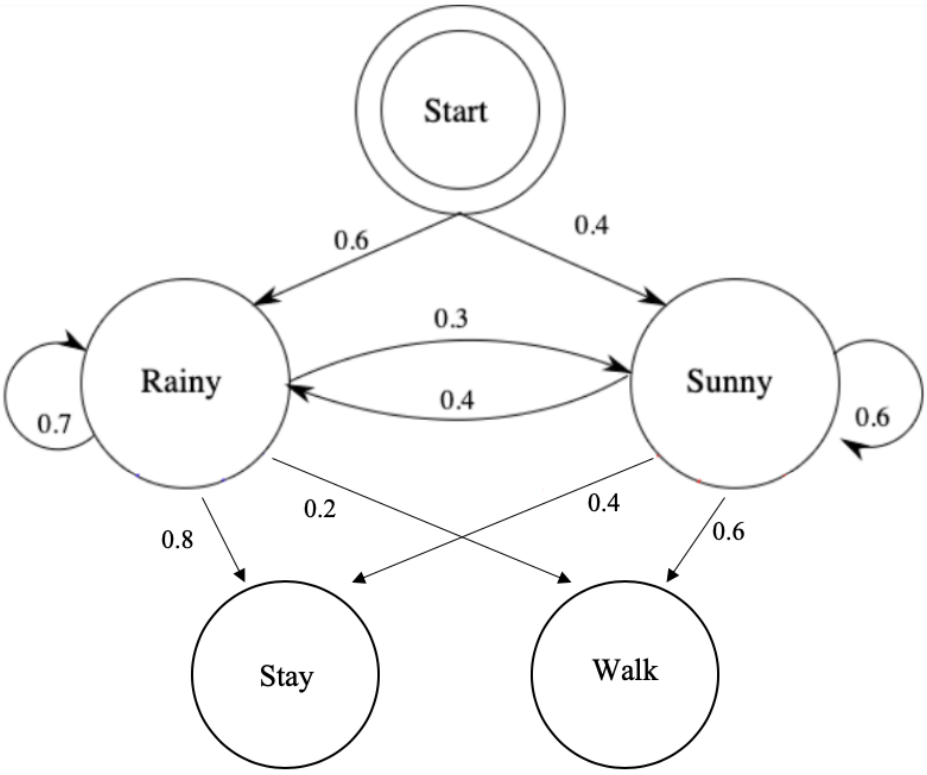

Figure 2. Example of a HMM.

(Source: author's depiction)

3.1.4 Viterbi algorithm

For any model, including HMM, which have hidden variables, decoding task is the process of determining which sequence of variables is the underlying source of some sequence of decoding observations. The definition of decoding in the context of HMM is: given input $\lambda = (A, B)$ and a sequence of observations $O = O_1, O_2, .., O_T$, find the most probable sequence of states $Q = q_1, q_2, \dots, q_T$.

The most widely used decoding algorithm for HMM is the Viterbi algorithm (Jurafsky and James 2014). This algorithm is a kind of dynamic programming. The algorithm first fills in the values in $\lambda = (\pi, A, B)$ using the recursive definition. It then uses the identity described before to calculate the highest probability for any sequence. So, this algorithm takes the most likely path.

A more detailed explanation of the Viterbi algorithm can be found in Appendix A.

The application of Viterbi algorithm in the empirical part was done in functions *makeViterbimat* and *get_states* (see Appendix D).

3.1.5 Baum-Welch algorithm

If transition and emission probabilities (matrices A and B) for a HMM are not known, then these have to be trained using the observation sequence O and the set of possible states S in the model, i.e. the learning problem mentioned in the section 3.1.2.

Baum-Welch algorithm (Baum, 1972) – the same as forward-backward algorithm is the most widely used algorithm for HMM training. This algorithm is a variation of the Expectation-Maximization (EM) algorithm (Dempster et al., 1977). The main point of the algorithm is the opportunity to estimate (improve the emission probabilities B and the transition probabilities A of the HMM.

If we know the state occupation probability (the state distribution at time t), we can derive the emission probability and the transition probability. If we know these two probabilities, we can derive the state distribution at time t. EM is an iterative algorithm, in which firstly initial estimate for the probabilities are computed and secondly these estimates are used for defining a better estimate of probabilities, and so on, iteratively improving the probabilities that it learns.

Baum-Welch algorithm starts with an estimate for the transition and observation probabilities and then uses these previously estimated probabilities to improve such probabilities. It is done by calculating the forward probability for an observation and after it such probability mass should be divided among all the possible paths that facilitate to this forward probability.

Detailed formulas of Baum-Welch algorithm can be found in Appendix B.

In the empirical part, Baum-Welch algorithm was applied by function *baumWelch* from package *HMM*.

3.2 k-means clustering algorithm

To conclude if the results of HMM are good enough, another basic model for comparison is needed. Based on the goal of the analysis clustering algorithm was chosen. Clustering algorithms are procedures for partitioning data into groups or clusters such that the clusters are distinct, and members of each cluster belong together. In our case the aim of the clustering algorithm is to divide transactions into 2 groups – *normal* and *suspicious*.

The k-means clustering algorithm is one of the most widely used clustering algorithms, so it was chosen. According to Hartigan and Wong (1979), the aim of the k-means algorithm is to divide M points in N dimensions into K clusters so that the within-cluster sum of squares is minimized.

The way k-means algorithm works is as follows:

1. Specify number of clusters K.
2. Initialize centroids (the most representative point within the group) by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters is not changing.
4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

The approach k-means follows to solve the problem is called Expectation-Maximization, which was mentioned before as a generalisation of the Baum-Welch algorithm. The E-step is assigning the data points to the closest cluster. The M-step is computing the centroid of each cluster. In a more formal way it is presented below.

The objective function is:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} \|x_i - \mu_k\|^2 \quad (5)$$

where $w_{ik} = 1$ for data point x_i if it belongs to cluster k; otherwise, $w_{ik} = 0$, $\|...\|$ is distance. Also, μ_k is the centroid of x_i's cluster.

It's a minimization problem of two parts. We first minimize J w.r.t. w_{ik} and treat μ_k as fixed. Then we minimize J w.r.t. μ_k and treat w_{ik} as fixed. Technically speaking, we differentiate J w.r.t. w_{ik}first and update cluster assignments (E-step). Then we differentiate J w.r.t. μ_k and recompute the centroids after the cluster assignments from the previous step (M-step). Therefore, E-step is:

$$\frac{\partial J}{\partial w_{ik}} = \sum_{i=1}^{m} \sum_{k=1}^{K} \|x_i - \mu_k\|^2 \implies w_{ik} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \left\|x_i - \mu_j\right\|^2 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

In other words, assign the data point x_i to the closest cluster judged by its sum of squared distance from the cluster's centroid.

And M-step is:

$$\frac{\partial J}{\partial \mu_k} = 2\sum_{i=1}^{m} w_{ik}(x_i - \mu_k) = 0 \implies \mu_k = \frac{\sum_{i=1}^{m} w_{ik} x_i}{\sum_{i=1}^{m} w_{ik}} \quad (7)$$

Which translates to recomputing the centroid of each cluster to reflect the new assignments.

To apply this algorithm in R, function *kmeans* from package *stats* was used, which is based on Hartigan and Wong (1979) version of specification.

3.3 Quality assessment metrics

To compare the results of two models, quality assessment metrics are needed. To compare HMM and k-means clustering algorithms we use Precision, Sensitivity, and F-Score.

Definitions of precision and sensitivity require a confusion matrix – a 2x2 table that cross-checks predictions with actual values. It is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. A 2x2 confusion matrix has 2 states for actual values – e.g. positive (True) and negative (False) – and 2 states for predicted values. The result is a table with 4 different combinations of predicted and actual values:

- true positives (TP): These are cases in which we predicted positive and it is true;
- true negatives (TN): We predicted negative and it's true;
- false positives (FP): We predicted positive and it's false (also known as "Type I error");
- false negatives (FN): We predicted negative and it is false (also known as "Type II error").

General form of confusion matrix is presented in Figure 3.

Precision (Positive predictive value) shows out of all the positive classes we have predicted correctly, how many are actually positive:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (8)$$

Sensitivity (Recall, True positive rate) shows out of all the positive classes, how much we predicted correctly:

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (9)$$

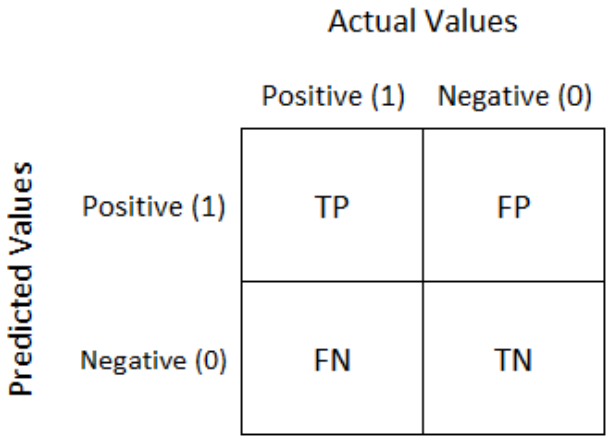

Figure 3. Confusion matrix

It is difficult to compare two models with low precision and high recall or vice versa. So, to make them comparable, we use F-Score. F-score helps to measure Sensitivity and Precision at the same time. It uses harmonic mean in place of arithmetic mean by punishing the extreme values more:

$$\text{F-Score} = \frac{2 * \text{Sensitivity} * \text{Precision}}{\text{Sensitivity} + \text{Precision}}. \quad (10)$$

4 Empirical study

This section consists of two parts. Firstly, the general description of data is provided, and three study cases are introduced with different data setups. Secondly, the description of the practical application of HMM and k-means clustering for every study is given.

4.1 Data

Two different databases were used to test HMM approach to detect suspicious financial transactions:

- artificial data - a database produced by Salv Technologies;
- real (anonymized) transaction data from a Baltic financial institution.

The artificially generated database contains randomly generated users, transactions, user action logs and meta info. One of the main features of this database is that some transactions are assumed to be suspicious, added to usual data at random time moments. That is why we used this database to build and test HMM. The second database with real data also had transactions which were found suspicious by AML specialists, so the available data gave rise to the following setup:

– **Study 1**. Firstly, the model was built based on 'training' data from the artificial database: 5568 transactions of 15 persons spread over 6 years, where 229 (4,1%) of them were assumed to be suspicious.

– **Study 2.** Secondly, the previously built model was tested on a higher amount of artificial data ('test' data): 252330 transactions of 16705 persons spread over 6 years, where 91 (0.04%) transactions by 30 persons were suspicious.

– **Study 3.** The model was updated and tested on the database with real data, where 0.36% persons were marked as suspicious by AML over a period of 5 months. Those persons made 0.01% out of all transactions. Overall, 103342 transactions of 1368 users were used.

Study 1 simulates a situation where we would have a small portion of initial data that is used to train the classifying models. The extracted dataset contains only users that have proven suspicious activity (for example, labelled by AML specialists). The data is used to catch the underlying signals of a suspicious transaction. Later it would be applied to future incoming transaction data, simulated in Study 2.

With artificial data you know what you are trying to catch, so Study 3 aims to apply the considered classification methods on real data.

Next transaction characteristics were used in the model:

- date of the transaction;
- direction of transaction (incoming and outgoing);
- amount in EUR;
- currency of the transaction;
- counterparty's country.

An example of the dataset of Study 1 can be found in Table 1.

Table 1. Data sample for Study 1

Transaction id	User id	Datetime of transaction	Incoming/ outgoing	Currency	Amount in EUR	Counterparty country
4768048	11097	2014-10-14 09:13:03	O	EUR	730.91	EE
6051363	41676	2016-01-20 03:22:21	I	EUR	7895.88	EE
19339305	40791	2020-03-15 21:19:40	I	CHF	234.54	EE

4.2 Study setup

The idea is that the hidden state in the model is the type of transaction: $normal$ or $suspicious$. An observable variable was taken to have 2 values: low_risk and $high\_risk$, which are defined using the auxiliary variable 'score'.

Variable 'score' is a numerical variable, which is based on combining different characteristics of transaction, for example, one such characteristic is checking if the counterparty country of transaction in a high-risk country. The list of high-risk countries (see Appendix C) includes high-risk and other monitored jurisdictions from FATF (FATF, 2020) and list of offshore countries created by the International Monetary Fund (International Monetary Fund, 2019) and European Commission (European Commission, 2019). The higher the value of variable 'score', the riskier transaction. The components of 'score' are taken by rules, which are usually used in rule-based method to detect money laundering and were built using domain knowledge (FATF, 2020).

Study 1 and 2

The best score composition was manually founded by the highest value of F-score for artificial training data of Study 1, reported in Table 2. The same score composition was also used for Study 2.

Table 2. Composition of 'score' for Study 1 and 2

Characteristics of transaction	Score increase
Time of transaction earlier than 7 AM	15
Time of transaction later than 9 PM	15
amount_in_eur>=1000 and amount_in_eur<5000	5
amount_in_eur>=5000 and amount_in_eur<10000	10
amount_in_eur>=10000 and amount_in_eur<20000	15
amount_in_eur>=20000	20
Counterparty country is high-risk country	5
In last 3 days for incoming transactions: sum(amount_in_eur)>=5000 and sum(amount_in_eur)<10000	5
In last 3 days for incoming transactions: sum(amount_in_eur)>=10000 and sum(amount_in_eur)<20000	10
In last 3 days for incoming transactions: sum(amount_in_eur)>20000	15
In last 3 days for outgoing transactions: sum(amount_in_eur)>=5000 and sum(amount_in_eur)<10000	5
In last 3 days for outgoing transactions: sum(amount_in_eur)>=10000 and sum(amount_in_eur)<20000	10
In last 3 days for outgoing transactions: sum(amount_in_eur)>20000	15
Number of transactions is last 7 days >=3 and <5	5
Number of transactions is last 7 days >=5 and <10	10
Number of transactions is last 7 days >=10	15

After calculating score for every transaction, observable variable was defined by the following rules and max(score) is the maximum score for each person:

- if score < max(score) $\cdot \frac{2}{3}$, then observable variable for this transaction is low_risk;
- if score >= max(score) $\cdot \frac{2}{3}$, then observable variable for this transaction is $high\_risk$;
- if max(score) = 0, then all transactions for this person get observable variable as low_risk.

Table 3 gives an example of a few transactions with auxiliary variables and calculated variable 'score'.

Table 3. Example of calculations for Study 1

Id	User id	Datetime	Direc-tion	Currency	Amount in EUR	Counterparty country	Susp	Incoming sum in 3 days	Outgoing sum in 3 days	Transactions in last 7 days	Score	Observation
15370819	48645	2020-01-19 19:02:55	I	JMD	13243.00	JM	1	13243.00	0.00	1	25	high_risk
15272885	48645	2020-01-20 23:13:32	O	EUR	59.19	EE	0	13243.00	59.19	2	25	high_risk
15370820	48645	2020-01-21 12:34:59	I	COP	212303.00	CO	1	225546.00	59.19	3	35	high_risk
14548136	48645	2020-01-09 15:31:10	O	EUR	85.69	EE	0	0.00	85.69	1	0	low_risk
14567528	48645	2020-01-09 19:49:09	O	EUR	5.37	EE	0	0.00	91.06	2	0	low_risk

HMM was used for every person separately, but the probabilities are the same for every person. The model was initialised with the following setup:

- hidden states: $S = \{S_1 = "normal" , S_2 = "suspicious"\}$;
- possible observable values: $\{"low\_risk" , "high\_risk"\}$;
- initial probabilities: $\pi = \{\pi_1 = 0.9; \pi_2 = 0.1\}$;
- transition probabilities: $A = \{a_{11} = 0.9; a_{12} = 0.1;\ a_{21} = 0.1; a_{22} = 0.9\}$;
- emission probabilities:
  $B = \{b_1(low\_risk) = 0.8;\ b_1(high\_risk) = 0.2;\ b_2(low\_risk) = 0.1;\ b_2(high\_risk) = 0.9\}$.

Study 3

Based on real historical data in Study 3, the composition of score was updated and can be found in Table 4.

Table 4. Composition of 'score' for Study 3

Characteristics of transaction	Score increase
Time of transaction earlier than 7 AM	15
Time of transaction later than 9 PM	15
Counterparty country is high-risk country	10
Currency of transaction is not EUR or USD	10
In last 3 days for incoming transactions: sum(amount_in_eur)>=5000 and sum(amount_in_eur)<10000	5
In last 3 days for incoming transactions: sum(amount_in_eur)>=10000 and sum(amount_in_eur)<20000	10
In last 3 days for incoming transactions: sum(amount_in_eur)>20000	15
In last 3 days for outgoing transactions: sum(amount_in_eur)>=5000 and sum(amount_in_eur)<10000	5
In last 3 days for outgoing transactions: sum(amount_in_eur)>=10000 and sum(amount_in_eur)<20000	10
In last 3 days for outgoing transactions: sum(amount_in_eur)>20000	15
Number of incoming transactions is last 7 days >=3 and <5	5
Number of incoming transactions is last 7 days >=5 and <10	10
Number of incoming transactions is last 7 days >=10	15
In last 3 days: sum of outgoing transactions >= 90% of sum of incoming transaction	15

A new rule compared with Table 3 was added: current transactions is more suspicious if sum of outgoing transactions in last 3 days is higher than at least 90% of sum of incoming transaction in last 3 days. This rule was added to improve the performance of the model.

Next changes in defining observable variable and probabilities are based on the higher F-score for data we have.

The logic for defining observable variable was changed:

- if score <= max(score) - 5, then observable variable for this transaction is low_risk;
- if score > max(score) - 5, then observable variable for this transaction is $high\_risk$;
- if max(score)=0, then all transactions for this person get observable variable as low_risk.

Some updates in probabilities:

- transition probabilities:
  $A = \{a_{11} = 0.8; a_{12} = 0.2;\ a_{21} = 0.1; a_{22} = 0.9\}$;
- emission probabilities:

$$B = \{b_1(low\_risk) = 0.9;\ b_1(high\_risk) = 0.1;\ b_2(low\_risk) = 0.01;\ b_2(high\_risk) = 0.99\}.$$

After defining scores and observation variable for all Studies, HMM was initiated for every person with the specific values for every Study given above. Then the Baum-Welch algorithm was used to find better transition and emission probabilities and finally, the Viterbi algorithm was used to predict the state for every transaction.

Later k-means clustering algorithm was used for the same dataset and based on variable 'score' transactions of every person were grouped in 2 clusters - *normal* and *suspicious*.

The results of the proposed HMM model was compared to k-means clustering algorithm for Studies 1-3. Quality assessment metrics – Precision, Sensitivity and F-score were used for the comparison.

5 Results

For every Study case, both HMM and k-means clustering algorithms were used for every person. For Study 1 and 2 the composition of 'score' variable is described in Table 2, for Study 3 - in Table 4.

Then for every person Precision and Sensitivity were calculated and as a general evaluation of both models an average of these indexes was taken. Based on them F-score was calculated for both methods and for every Study case. The results for all Studies can be found in Table 5.

In all Studies Sensitivity is higher than Precision for both models, but for k-means clustering the difference between these two factors is bigger. So, for k-means method compared to HMM Precision is always much lower and Sensitivity slightly bigger. Comparing Study 1 and Study 2 it should be mentioned, that for higher amount of data (Study 2) for HMM Precision decreased, and Sensitivity increased. Because of this, F-score of HMM for Study 2 smaller compared to Study 1, but still better than that of k-mean clustering. Study 3 shows better results compared to Study 1 and 2, because model fits suspicious data closer and as a consequence of this Sensitivity is 100%. Still, Precision is higher for HMM, than for k-means. Based on the F-score it can be concluded that HMM performs better than k-means clustering algorithm for detecting suspicious transactions in all Studies.

Table 5. Comparison of models for all Studies

	Study 1		**Study 2**		**Study 3**	
	HMM	**k-means**	**HMM**	**k-means**	**HMM**	**k-means**
Precision	52%	22%	40%	24%	67%	47%
Sensitivity	61%	87%	70%	77%	100%	100%
F-score	0.56	0.35	0.53	0.36	0.81	0.64

For Study 2, 86% of all persons were defined as *normal* by HMM and they are really not suspicious. We also got 8.1% of all transactions as *suspicious* by model, but marked *normal* within the dataset. These transactions should be investigated by an AML specialist, to be sure they are really not suspicious. For Study 3, among persons which have only normal transactions, 6.4% of transactions were marked as *suspicious* by the HMM so they should also be checked by an AML specialist.

6 Conclusions

The goal of the thesis was to introduce, build and test HMM for detecting suspicious transactions. For testing, artificial and real data was acquired, and as a benchmark, k-means clustering was chosen. All of the thesis goals were achieved.

The thesis provides an overview of machine learning methods for detecting money laundering among transactions and gives a detailed summary of hidden Markov model. Based on this method the practical model was built in R software. The model was calibrated and tested using more than 250 thousand transactions of artificial data and more than 100 thousand transactions of real data. To conclude about the quality of the model, it was compared with k-means clustering using Precision, Sensitivity and F-score. Finally, it was shown that HMM provides better results for detecting suspicious transactions compared with k-means clustering in a sense of higher F-score.

Based on such results the model was proposed to a Estonian anti-money laundering company, which offers an AML platform (incl. transaction monitoring). HMM could replace or enhance its current method for detecting suspicious transactions – a rule-based approach. It is also possible to add current existing rules into the composition of the 'score' variable.

One of the limitations of HMM is the variable 'score', which is based on the general understanding of money laundering logic. Moreover, this variable is not dynamic and should be review every year or few years. To improve HMM, it could be useful to increase the number of observed variable levels and add rules from rule-based approach into the composition of variable 'score'.

Appendices

Appendix A. Viterbi algorithm

Jurafsky and James in 2014 explain that idea of the Viterbi algorithm is going through the observation sequence from left to right with the aim to fill out a lattice. The lattice consists of cells, which are called $r_t(j)$ and implicate the probability that the HMM currently is in state j after reviewing the first t observations with the route through the sequence of the most probable states $q_1..q_{t-1}$, given the automaton λ. Cells $r_t(j)$ are defined by recursively taking the most probable path that could bring into this cell. In a formal way, each cell presents the probability $r_t(j) = P(q_1..q_{t-1}, O_1, O_2..O_t, q_t = j|\lambda)$.

It should be noted that to represent the most probable path we use the maximum value over all possible previous state sequences $\max_{q_1,..,q_{t-1}}$. Using the general logic of dynamic programming algorithms, Viterbi algorithm fills each cell recursively. Firstly, the probability of being in every state at time $t-1$ is calculated, and then, using this information, the Viterbi probability is calculated by choosing the most probable of the extensions of the paths that brings us to the current cell. With a given state q_j at time t, the value $r_t(j)$ is computed as

$$r_t(j) = r_{t-1}(i)a_{ij}b_j(O_t), \quad \text{(A.1)}$$

where $r_{t-1}(i)$ is the previous Viterbi path probability from the previous time step; a_{ij} is the transition probability from previous state q_i to current state q_j and $b_j(O_t)$ is the state observation likelihood of the observation symbol O_t given the current state j.

So, the formal definition of the Viterbi recursion could be given in the next way:

1. Initialization:

$$r_1(j) = \pi_j b_j(O_1) \quad 1 \le j \le N \quad \text{(A.2)}$$

$$rt_1(j) = 0 \quad 1 \le j \le N \quad \text{(A.3)}$$

2. Recursion

$$r_t(j) = r_{t-1}(i)a_{ij}b_j(O_t)\ ;\ 1 \le j \le N, 1 < t \le T \quad \text{(A.4)}$$

$$bt_t(j) = \arg\max_{i=1,..,N} r_{t-1}(i)a_{ij}b_j(O_t)\ ;\ 1 \le j \le N, 1 < t \le T \quad \text{(A.5)}$$

3. Termination:

The best score: $P^* = r_T(i)$;

The start of back trace: $q_T^* = \arg\max_{i=1,..,N} r_T(i)$

Appendix B. Baum-Welch algorithm

As Jurafsky and James in 2014 mentioned, firstly, backward probability should be defined. The backward probability β is the probability of seeing the observations from time $t+1$ to the end, given that we are in state i at time t (and given the automaton λ):

$$\beta_t(i) = P(O_{t+1}, O_{t+2}..O_T|q_t = i, \lambda) \quad \text{(B. 1)}$$

For calculation of these probabilities, the induction algorithm is used:

1. Initialization:

$$\beta_T(i) = 1, \qquad 1 \le i \le N \quad \text{(B. 2)}$$

2. Recursion

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)\,, \qquad 1 \le i \le N, 1 \le t < T \quad \text{(B. 3)}$$

3. Termination:

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j b_j(O_1) \beta_1(j) \quad \text{(B. 4)}$$

Below it is mentioned how the transition probability a_{ij} and observation probability $b_i(O_t)$ from an observation sequence can be calculated using the forward and backward probabilities, even though the real path taken by the model is not observed (hidden).

Firstly, $\hat{a}_{ij}$ should be estimated as a kind of simple maximum likelihood estimation:

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \quad \text{(B. 5)}$$

To compute the numerator, next intuition should be used: assume that some estimate of the probability that a given transition $i \to j$ was taken at a particular point in time t in the given observation sequence. If this probability for each particular time point t is known, then the sum over all time up to time point t will be the estimate for the total count of $i \to j$ transitions.

In a formal way, the probability ξ_t could be defined as the probability of being in state i at time t and state j at time $t+1$, given the observation sequence and the model:

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j|O, \lambda) \quad (B. 6)$$

To compute ξ_t, firstly it should be computed a probability which is similar to ξ_t, but differs in including the probability of the observation; note the different conditioning of O from (B.6):

$$\text{not-quite-}\xi_t(i,j) = P(q_t = i, q_{t+1} = j, O|\lambda) \quad \text{(B. 7)}$$

There are different probabilities that are used to calculate not-quite-ξ_t: the transition probability for the arc in question, the α probability before the arc, the β probability after the arc, and the observation probability for the symbol just after the arc, where arc is used in a sense of curve between two points (cells). These four probabilities are multiplied together to produce not-quite-ξ_t in the following way:

$$\text{not-quite-}\xi_t(i,j) = \alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j) \quad \text{(B. 8)}$$

To compute ξ_t from not-quite-ξ_t, the laws of probability and dividing by $P(O|\lambda)$ should be used, since:

$$P(X|Y,Z) = \frac{P(X,Y|Z)}{P(Y|Z)} \quad \text{(B. 9)}$$

The probability of the observation given the model is simply the forward probability of the whole statement (or alternatively, the backward probability of the whole statement):

$$P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j) \quad \text{(B. 10)}$$

So, the final equation for ξ_t is

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j)\,\beta_t(j)} \quad \text{(B. 11)}$$

The expected number of transitions from state i to state j is then the sum over all t of ξ. For the estimate of a_{ij} in (B.5), one more thing is needed: the total expected number of transitions from state i. It can be calculated by summing over all transitions out of state i. So, the final formula for $\hat{a}_{ij}$ will be such:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)} \quad \text{(B. 12)}$$

Moreover, a formula for recomputing the observation probability also needed. This is the probability of a given symbol v_k from the observation vocabulary V, given a state j: $\hat{b}_j(v_k)$. Next formula should be used:

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j} \quad \text{(B. 13)}$$

For this, $\gamma_t(j)$ - the probability of being in state j at time t, is needed:

$$\gamma_t(j) = P(q_t = j|O, \lambda) \tag{B.14}$$

Once again, it could be computed by including the observation sequence in the probability:

$$\gamma_t(j) = \frac{P(q_t = j, O|\lambda)}{P(O|\lambda)} \tag{B.15}$$

The numerator of (B.15) is the product of the backward probability and the forward probability:

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)} \tag{B.16}$$

Next step is computing b. To get the numerator, the sum $\gamma_t(j)$ for all time steps t in which the observation O_t is the symbol v_k should be used. To calculate the denominator, the sum $\gamma_t(j)$ over all time steps t should be found. Finally, it will be the percentage of the times that it was in state j and has symbol v_k (the notation $\sum_{t=1\ s.t.O_t=v_k}^{T}$ means "sum over all t for which the observation at time t was v_k"):

$$\hat{b}_j(v_k) = \frac{\sum_{t=1\ s.t.O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{B.17}$$

So now formulas from (B.12) and (B.17) should be used for re-estimation the transition probabilities A and observation probabilities B from an observation sequence O, assuming that we already have a previous estimate of A and B.

These re-estimations create a basis of the iterative forward-backward algorithm. This algorithm starts with an initial estimate of the HMM parameters $\lambda = (A, B)$. Like other cases of the EM (expectation-maximization) algorithm, the forward-backward algorithm has two iterative steps: the expectation step, or E-step, and the maximization step, or M-step.

In the E-step, 2 variables should be computed: the expected state occupancy count γ and the expected state transition count ξ from the earlier A and B probabilities. In the M-step, γ and ξ are used to recompute new A and B probabilities.

Appendix C. List of high-risk countries

Country code	Full name
AF	Afghanistan
AI	Anguilla
AG	Antigua and Barbuda
AW	Aruba
PT-20	Azores
BS	Bahamas
BH	Bahrain
BB	Barbados
BY	Belarus
BZ	Belize
BM	Bermuda
BA	Bosnia and Herzegovina
BN	Brunei Darussalam
BF	Burkina Faso
KH	Cambodia
KY	Cayman Islands
CF	Central African Republic
CG	Congo
CK	Cook Islands
CI	Cote d'Ivoire
CU	Cuba
CW	Curacao
KP	Democratic People's Republic of Korea (DPRK)
DJ	Djibouti Republic
DM	Dominica
DO	Dominican Republic
EC	Ecuador
ER	Eritrea
ET	Ethiopia
GH	Ghana
GI	Gibraltar
WG	Grenada
GT	Guatemala
GG	Guernsey
DW	Guinea Bissau
GY	Guyana
HK	Hong Kong
IR	Iran
IQ	Iraq
IM	Isle of Man
JA	Jamaica
JE	Jersey
KE	Kenya
LA	Lao People's Democratic Republic
LB	Lebanon
LR	Liberia
LY	Libya
MO	Macao
PT-30	Madeira

Country code	Full name
MV	Maldives
MH	Marshall Islands
MU	Mauritius
MS	Montserrat
MZ	Mozambique
MM	Myanmar (Burma)
NA	Namibia
NR	Nauru
NC	New Caledonia
NU	Niue
PK	Pakistan
PW	Palau
PS	Palestine State of
PA	Panama
RU	Russian Federation
SH	Saint Helena, Ascension and Tristan da Cunha
KN	Saint Kitts and Nevis
PM	Saint Pierre and Miquelon
VC	Saint Vincent and the Grenadines
WS	Samoa
RS	Serbia
SC	Seychelles
SX	Sint Maarten
SO	Somalia
SS	South Sudan
LK	Sri Lanka
SD	Sudan
SZ	Swaziland (Eswatini)
SY	Syria
PF	Tahiti (French Polynesia)
TL	Timor-Leste
TO	Tonga
TT	Trinidad and Tobago
TN	Tunisia
TC	Turks and Caicos Islands
UG	Uganda
UY	Uruguay
VU	Vanuatu
VE	Venezuela
VG	Virgin Islands, British
VI	Virgin Islands, U.S.
YE	Yemen

Appendix D. R code

```
library(HMM)
library(data.table)
#defining functions for Viterbi algorithm
makeViterbimat <- function(sequence, transitionmatrix, emissionmatrix)
{
 sequence <- toupper(sequence)
 numstates <- dim(transitionmatrix)[1]
 v <- matrix(NA, nrow = length(sequence), ncol =
dim(transitionmatrix)[1])
 v[1, ] <- 0
 v[1,1] <- 1
 for (i in 2:length(sequence))
 {
 for (l in 1:numstates)
 {
  statelprobnucleotidei <- emissionmatrix[l,sequence[i]]
  v[i,l] <- statelprobnucleotidei * max(v[(i-1),] *
transitionmatrix[,l])
 }
 }
 return(v)
}

get_states <- function(sequence, transitionmatrix, emissionmatrix)
{
 states <- rownames(theemissionmatrix)
 v <- makeViterbimat(sequence, transitionmatrix, emissionmatrix)
 mostprobablestatepath <- apply(v, 1, function(x) which.max(x))
 return(mostprobablestatepath)
}

View(all_dd) #our dataset
all_dd=data.table(all_dd)
users=unique(all_dd$user_id)

all_dd[,score:=0] #creating empty score column
all_dd[as.ITime(date_created)<=as.ITime('07:00:00'),score:=score+15 ]
all_dd[as.ITime(date_created)>=as.ITime('21:00:00'),score:=score+15]
```

```
all_dd[amount_in_eur>=1000 & amount_in_eur<5000, score:=score+5]
all_dd[amount_in_eur>=5000 & amount_in_eur<10000, score:=score+10]
all_dd[amount_in_eur>=10000 & amount_in_eur<20000, score:=score+15]
all_dd[amount_in_eur>=20000, score:=score+20]

all_dd[counterparty_country %in% c('AF','AI','AG','AW','PT-
20','BS','BH','BB','BY','BZ','BM','BA','BN','BF','KH','KY','CF','CG','
CK','CI','CU','CW','KP','DJ','DM',
'DO','EC','ER','ET','GH','GI','WG','GT','GG','DW','GY','HK','IR',
'IQ','IM','JA','JE','KE','LA','LB','LR','LY','MO','PT-30','MV','MH',
'MU','MS','MZ','MM','NA','NR','NC','NU','PK','PW','PS','PA','SC','RU',
'WS','RS','SX','SO','SS','LK','SH','KN','PM','VC','SD','SZ','SY','PF',
'TL','TO','TT','TN','TC','UG','UY','VU','VE','VG','VI','YE'),
score:=score+5]
all_dd[currency!='EUR'& currency!='USD', score:=score+10]

#creating table where we will put summary of the model for every
person
results=data.table(users)
results[,det_sups:=-1]
results[,det_sups_really_susp:=-1]
results[,really_susp:=-1]
results[,all_transactions:=-1]


#building HMM for every person
for (j in users) {
 dd=all_dd[user_id==j]
 #calculatig additional transaction characteristics
 dd[,sum_3in:=0] # sum of amount in eur in last 3 days for incoming
transactions
 dd[,sum_3out:=0] # sum of amount in eur in last 3 days for outgoing
transactions
 dd[,count_7:=0] # count of transaction in last 7 days
 for (i in 1:nrow(dd)) {
 dd$sum_3in[i]=sum(dd[difftime(dd$date_created[i], date_created,
units='days')<=3 & (date_created<=dd$date_created[i]) & type=='I',
amount_in_eur ])
```

```
 dd$sum_3out[i]=sum(dd[difftime(dd$date_created[i], date_created,
units='days')<=3 & (date_created<=dd$date_created[i])& type=='O',
amount_in_eur ])
 dd$count_7[i]=length(dd[difftime(dd$date_created[i], date_created,
units='days')<=7 & (date_created<=dd$date_created[i]), amount_in_eur
])
 }

 dd[sum_3in>=5000 & sum_3in<10000, score:=score+5]
 dd[sum_3in>=10000 & sum_3in<20000, score:=score+10]
 dd[sum_3in>=20000 , score:=score+15]

 dd[sum_3out>=5000 & sum_3out<10000, score:=score+5]
 dd[sum_3out>=10000 & sum_3out<20000, score:=score+10]
 dd[sum_3out>=20000 , score:=score+15]

 dd[count_7>=3 & count_7<5, score:=score+5]
 dd[count_7>=5 & count_7<10, score:=score+10]
 dd[count_7>=10 , score:=score+15]

 mgr=dd[,max(score)] # maximum score for person

 #creating observable variable
 dd[score<mgr*2/3 | mgr==0, observation:='low_risk']
 dd[score>=mgr*2/3 & mgr>0, observation:='high_risk']

 #defining parameters for HHM
 states    <- c("normal", "susp") # define the names of the states
 normprobs   <- c(0.9, 0.1)    # set the probabilities of switching
states, where the previous state was "normal"
 suspprobs   <- c(0.1, 0.9)    # set the probabilities of switching
states, where the previous state was "susp"
 thetransitionmatrix <- matrix(c(normprobs, suspprobs), 2, 2, byrow =
TRUE) # create a 2 x 2 matrix
 rownames(thetransitionmatrix) <- states
 colnames(thetransitionmatrix) <- states
 observations   <- c("low_risk", "high_risk") # define the alphabet of
observations
```

```
 normstateprobs <- c(0.8, 0.2) # set the values of the probabilities,
for the normal state
 suspstateprobs <- c(0.1, 0.9) # set the values of the probabilities,
for the susp state
 theemissionmatrix <- matrix(c(normstateprobs, suspstateprobs), 2, 2,
byrow = TRUE) # Create a 2 x 2 matrix
 rownames(theemissionmatrix) <- states
 colnames(theemissionmatrix) <- observations
 myseq<- dd$observation # create a vector of obervable variable

 #initialization HMM
 hmm = initHMM(c("normal","susp"),observations,
    transProbs=thetransitionmatrix,
    emissionProbs=theemissionmatrix)
 # Baum-Welch algorithm for updating transition and emission
probabilities
 bw = baumWelch(hmm,myseq,5)
 thetransitionmatrix<-bw$hmm$transProbs
 theemissionmatrix<-bw$hmm$emissionProbs
 # using Viterbi algorithm to predict state for every transaction
 res=get_states(myseq, thetransitionmatrix, theemissionmatrix)

 def_susp=c(which(res %in% c(2)))
 dd_s=dd[def_susp] # taking transactions which defined as suspisious
 results[users==j,det_sups:= sum(res!=1)] # number of transactions
which defined as suspisious
 results[users==j,det_sups_really_susp:= nrow(dd_s[susp>0])] # number
of really suspisious transaction among defined as suspisious
 results[users==j,really_susp:= nrow(dd[susp>0])] # number of really
suspisious transaction
 results[users==j,all_transactions:= nrow(dd)] # number of all
transactions
}

results[,pres:=det_sups_really_susp/det_sups] # precision for every
person
results[,sens:=det_sups_really_susp/really_susp] # sensitivity for
every person
results[ mean(pres)] # average precision among all persons
```

```
results[ mean(sens)] # average sensitivity among all persons

#using k-means clustering for every person
for (j in users) {
 dd=all_dd[user_id==j]
 #calculatig additional transaction characteristics
 dd[,sum_3in:=0] # sum of amount in eur in last 3 days for incoming
transactions
 dd[,sum_3out:=0] # sum of amount in eur in last 3 days for outgoing
transactions
 dd[,count_7:=0] # count of transaction in last 7 days
 for (i in 1:nrow(dd)) {
 dd$sum_3in[i]=sum(dd[difftime(dd$date_created[i], date_created,
units='days')<=3 & (date_created<=dd$date_created[i]) & type=='I',
amount_in_eur ])
 dd$sum_3out[i]=sum(dd[difftime(dd$date_created[i], date_created,
units='days')<=3 & (date_created<=dd$date_created[i])& type=='O',
amount_in_eur ])
 dd$count_7[i]=length(dd[difftime(dd$date_created[i], date_created,
units='days')<=7 & (date_created<=dd$date_created[i]), amount_in_eur])
 }

 dd[sum_3in>=5000 & sum_3in<10000, score:=score+5]
 dd[sum_3in>=10000 & sum_3in<20000, score:=score+10]
 dd[sum_3in>=20000 , score:=score+15]

 dd[sum_3out>=5000 & sum_3out<10000, score:=score+5]
 dd[sum_3out>=10000 & sum_3out<20000, score:=score+10]
 dd[sum_3out>=20000 , score:=score+15]

 dd[count_7>=3 & count_7<5, score:=score+5]
 dd[count_7>=5 & count_7<10, score:=score+10]
 dd[count_7>=10 , score:=score+15]

 print(unique(dd$user_id))
 Cluster <- kmeans(dd$score, 2)
 print(table(Cluster$cluster, dd$susp)) #susp as row (0, 1), clusters
as columns (1,2)
}
```

References

Ahmed, M., Mahmood, A.N., and Hu, J. (2016). A survey of network anomaly detection techniques, *Journal of Network and Computer Applications*, 60, pp.19-31.

Awoyemi, J.O., Adetunmbi, A.O., and Oluwadare, S.A. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis, *International Conference on Computing Networking and Informatics*, pp. 1-9.

Ayasdi.com 2020, computer software. Available at: https://www.ayasdi.com/applications/anti-money-laundering/

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities III: Proceedings of the 3rd Symposium on Inequalities*, pp. 1–8.

Cao, D.K., Do, P. (2012). Applying Data Mining in Money Laundering Detection for the Vietnamese Banking Industry. In: JS. Pan, SM. Chen, N.T. Nguyen, ed., *Intelligent Information and Database Systems*. Berlin: Springer, pp. 207-216.

Cassara, J. (2015). *Trade-Based Money Laundering: The Next Frontier in International Money Laundering Enforcement*. Wiley.

Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey, *ACM computing surveys*, 41(3), pp.1-58.

Chen, M.Y., Kundu, A., and Zhou, J. (1994). Off-line handwritten word recognition using a hidden Markov model type stochastic network, *IEEE transactions on Pattern analysis and Machine Intelligence*, 16(5), pp.481-496.

Chen, Z., Dinh, L., Khoa, V., Nazir, A., Teoh, E.N., Karupiah, E.K., and Lam, K.S. (2018). Machine learning techniques for anti-money laundering (AML) solutions in suspicious transaction detection: a review. *Knowledge & Information Systems,* 57, pp. 245–285.

Coghlan, A. (2011). *A Little Book of R For Bioinformatics.* E-book library [online]. Available at: https://a-little-book-of-r-for-bioinformatics.readthedocs.io/ en/latest/index. html (Accessed: 16 April 2020)

Comply Advantage. (2019). *Ever watchful – the AI and machine learning promise* [Online]. Available at: https://complyadvantage.com/blog/ever-watchful-ai-machine-learning-promise/

Cox, D. (2012). *Handbook of anti-money laundering*. John Wiley & Sons, Ltd.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977) Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1), pp. 1–21.

Estonian Ministry of Finance. (2019). *Money Laundering and Terrorist Financing Prevention Act* [Online]. Available at: https://www.riigiteataja.ee/en/eli/ee/Riigikogu/act/525032019005/consolide

Estonian Police and Border Guard Board. (2020). *Estonian Financial Intelligence Unit* [Online]. Available at: https://www2.politsei.ee/en/organisatsioon/rahapesu-andmeburoo/

European Commission. (2019). *Anti-money laundering and counter terrorist financing* [Online]. Available at: https://ec.europa.eu/info/business-economy-euro/banking-and-finance/financial-supervision-and-risk-management/anti-money-laundering-and-counter-terrorist-financing_en

FATF. (2020). *Financial Action Task Force* [Online]. Available at: http://www.fatf-gafi.org/

GuardianAnalytics.com 2020, computer software. Available at: https://guardiananalytics.com/

Hartigan, J., Wong, M. (1979). A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society*, 28(1), pp. 100-108.

International Monetary Fund (2019). *Past IMF Staff Assessments on Offshore Financial Centers* [Online]. Available at: https://www.imf.org/external/NP/ofca/OFCA.aspx

Jadhav, S.N., Bhandari, K. (2013). Anomaly Detection Using Hidden Markov Model, *International Journal of Computational Engineering Research*, 3(7), pp. 28-35.

Jurafsky, D., James H. M. (2014). *Speech and Language Processing*. 2nd edn. Harlow: Pearson Education.

Kim, R. (2019). *Analysis: Danske Bank, Swedbank, and Global AML Failure* [Online]. Available at: https://news.bloomberglaw.com/bloomberg-law-analysis/analysis-danske-bank-swedbank-and-global-aml-failures

Li, Y., Duan, D., Hu, G., and Lu, Z. (2009). Discovering Hidden Group in Financial Transaction Network Using Hidden Markov Model and Genetic Algorithm. In: *Sixth International Conference on Fuzzy Systems and Knowledge Discovery*. Tianjin, pp. 253-258.

Madinger, J. (2011). *Money Laundering: A Guide for Criminal Investigators*. 3rd edn. CRC Press.

Maes, S., Tuyls, K., Vanschoenwinkel, B., and Manderick, B. (2002). Credit card fraud detection using Bayesian and neural networks, *Proceedings of the 1st international naiso congress on neuro fuzzy technologies*, pp. 261-270.

Mhamane, S.S., Lobo, L.M. (2012). Use of Hidden Markov Model as Internet Banking Fraud Detection. *International Journal of Computer Applications,* 45(21), pp. 5-10.

Palshikar, G., Apte, M., and Baskaran, S. (2014). Analytics for detection of money laundering. Technical Report: Tata Consultancy Services Limited.

Perols, J. (2011). Financial Statement Fraud Detection: An Analysis of Statistical and Machine Learning Algorithms, *A Journal of Practice & Theory*, 30(2), pp. 19-50.

R Core Team (2018). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. Available at: https://www.R-project.org/.

Raphael, C. (1999). Automatic segmentation of acoustic musical signals using hidden Markov models, *IEEE transactions on pattern analysis and machine intelligence*, 21(4), pp. 360-370.

RiskScreen (2019). *Estonia orders Danske Bank to shut down following money laundering scandal* [Online]. Available at: https://www.riskscreen.com/kyc360/news/estonia-orders-danske-bank-to-shut-down-following-money-laundering-scandal/.

Rohit, K.D., Patel, D.B. (2015). Review On Detection of Suspicious Transaction In Anti-Money Laundering Using Data Mining Framework. *International Journal for Innovative Research in Science & Technology*, 8(1), pp. 129-133.

Sahin, Y., Bulkan, S., and Duman, E. (2013). A cost-sensitive decision tree approach for fraud detection, *Expert Systems with Applications*, 40(15), pp. 5916-5923.

SAS.com 2020, computer software. Available at: https://www.sas.com/en_us/software/anti-money-laundering.html

Schuller, B., Rigoll, G., and Lang, M. (2003). Hidden Markov model-based speech emotion recognition, *International Conference on Acoustics, Speech, and Signal Processing*, 2, pp. 401-404.

Singh D., Pandit R. (2015). Credit Card Fraud Detection Using Hidden Markov Model. *International Journal of Scientific & Engineering Research*, 6(1), pp. 1488-1491.

Stamp, M. (2018). *Introduction to Machine Learning with Applications in Information Security*. New York: Chapman and Hall/CRC

TransferWise. (2019). Comply with me! How we keep TransferWise safe [Online]. Available at: https://transferwise.com/gb/blog/we-keep-transferwise-safe

Umadevi, P., Divya, E. (2012). Money laundering detection using TFA system. In: *Software Engineering and Mobile Application Modelling and Development*. Chennai, pp. 1-8.

Varga, A., Moore, R.K. (1990). Hidden Markov model decomposition of speech and noise, *International Conference on Acoustics, Speech, and Signal Processing*, pp. 845-848.

Wyler, G. (2011). How A Major U.S. Bank Laundered Billions In Mexican Drug Money [Online]. Available at: https://www.businessinsider.com/how-wachovia-laundered-billions-in-mexican-drug-money-2011-4.

Yang, J., Xu, Y.S. (1994). Hidden Markov model for gesture recognition. Master thesis, The Robotics Institute Carnegie Mellon Univ. Pittsburgh, Pennsylvania 15213.

Yang, Y., Guan, X., and You, J. (2002). CLOPE: A Fast and Effective Clustering Algorithm for Transactional Data, *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 682-687.

Yasaka, N. (2017). Data mining in anti-money laundering field. *Journal of Money Laundering Control*, 20(3), pp. 301-310.

Non-exclusive licence to reproduce thesis and make thesis public

I, Kseniia Kasianova,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, **Detecting money laundering using Hidden Markov Model**, supervised by Kaur Lumiste.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

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

Kseniia Kasianova
25.05.2020