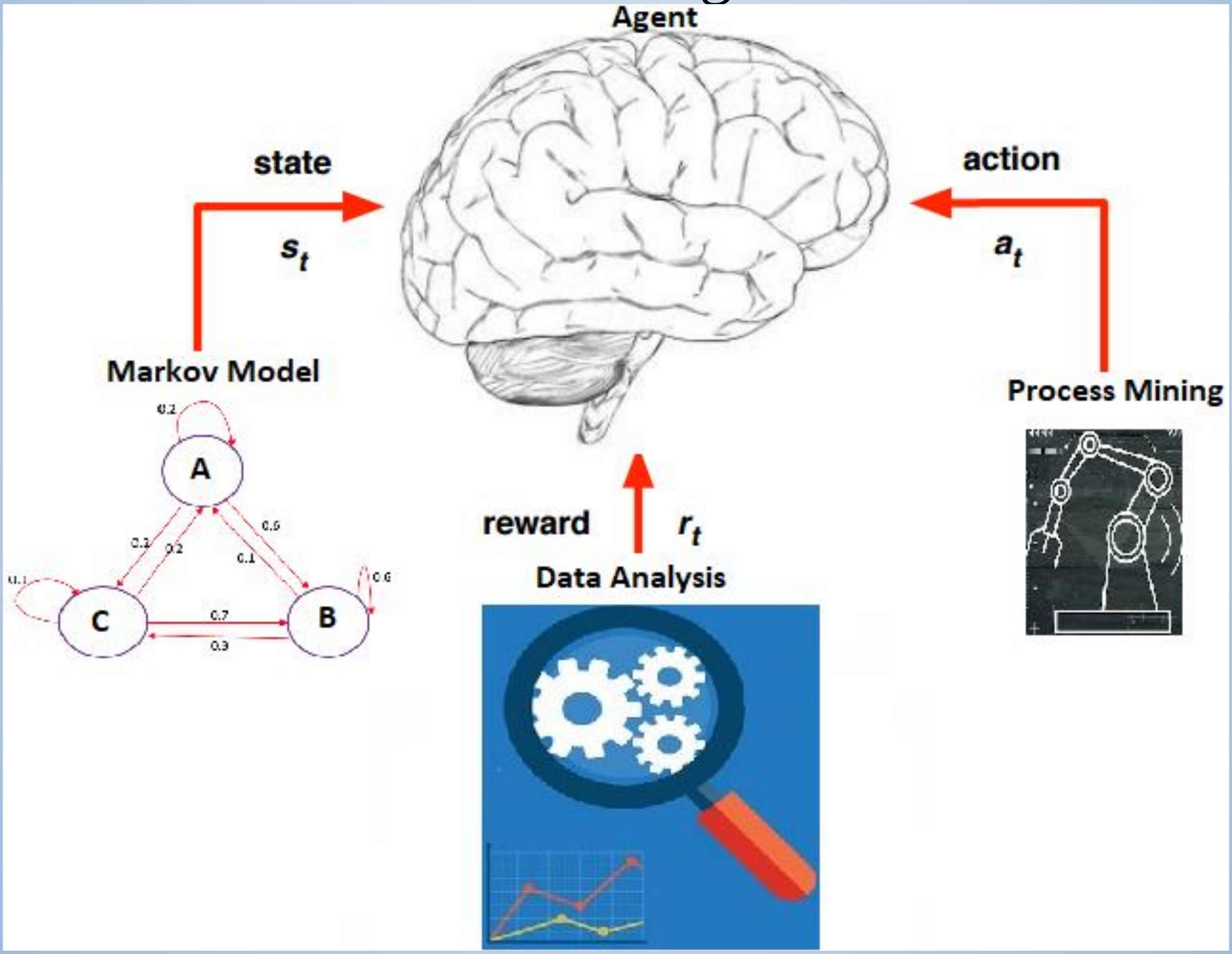
A Master Thesis on

Predictive Business Process Monitoring with Markov Models and Reinforcement Learning



Supervisor Professors Alexandros Bousdekis and Georgios Miaoulis
Department of Informatics and Computer Engineering, University of West Attica, Athens, Greece

Athens, 2021

UNIVERSITY OF WEST ATTICA

SCHOOL OF ENGINEERING

DEPARTMENT OF INFORMATICS AND COMPUTER ENGINEERING

Predictive Business Process Monitoring with Markov Models and Reinforcement Learning

Members of the Committee of Inquiry including the Rapporteur

The thesis / diploma thesis was successfully examined by the following Examination Committee:

NAME SURNAME	DIGITAL SIGNATURE
BARDIS Georgios	Georgios Bardis Digitally signed by Georgios Bardis Date: 2021.11.23 19:09:54 +02'00'
BOUSDEKIS Alexandros	Alexandros Bousdekis Digitally signed by Alexandros Bousdekis Date: 2021.11.22 14:19:22 +02'00'
VOULODIMOS Athanasios	Athanasios Voulodimos Digitally signed by Athanasios Voulodimos Date: 2021.11.23 18:37:26 +02'00'

[illegible]

Abstract

Informed decision making is at the forefront of many disciplines. Despite the domain we can always make preferable and educated decisions. This can lead to more insight, formulation of new and improved policies, always based on quality data. Specifically in the business domain, an organization can have more consistent results, transparency throughout the whole business process, improved performance and increased revenue.

With the usage of process mining techniques, behaviors of processes can be analyzed and improved. Also modeling uncertainty through the usage of stochastic models can be an additional factor in the attempt to either or both improve and understand existing process and behaviors. The trend of automatic core functions in complex system such as an organization is even more relevant with the rapid advancement of artificial intelligence and more specifically machine leaning. A common stochastic model is Markov decision process (MDP) and MDPs are used to model reinforcement learning environments. With the goal of combining process mining, stochastic models particularly MDPs and reinforcement learning we aim to answer the question.

In this thesis we strive to answer the question: Can predictive process monitoring be automated and if so at what level. Using a Business Process Intelligence challenge dataset and more specifically the 2017 Offer event log dataset. We attempt to further research the usage and application of these tools in the monitoring of business processes.

Keywords: predictive business monitoring, process mining, stochastic models, Markov chain, Markov decision process, reinforcement learning

Acknowledgements

First and foremost we would like to thank our supervisor Mr. Alexandros Bousdekis who guided, supported and helped us throughout all the research and development, always in time, even on his vacation days.

We would also like to thank Mr. Georgios Miaoulis for providing us with such an informative and interesting project, which through research and implementation helped us grow and become a better version of ourselves.

We would like to thank our friends and families, for supporting us through all our efforts and difficulties in studying and completing our university and thesis.

An honorable mention our dear friend Mario who supported us.

Table of contents

List of figures

List of Tables

Table of Acronyms

AI	Artificial Intelligence
ANN	Artificial Neural Networks
BPM	Business Process Management
BPMN	Business Process Model and Notation
DB	Database
DFG	Directly-follows graph
DQN	Deep Q-Network
LOC	Local Outlier Factor
MC	Monte Carlo
MDP	Markov Decision Process
ML	Machine Learning
PCA	Principal Component Analysis
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
SARSA	State–action–reward–state–action
SL	Supervised Learning
SVM	Support-vector machine
TD	Temporal Difference
XES	eXtensible Event Stream
XML	Extensible Markup Language

1.1 Introduction

Business process monitoring has the purpose of measuring and analyzing performance, improving certain processes and identifying critical points to activities. To accomplish this task usually software for Business Process Modeling is used. Specifically for processes, one can use process mining techniques. These techniques allow for the extraction of process model according to specific data alongside useful metrics, activity and performance-oriented information. The models produced do not take into account the uncertainty of the environment. In order to further increase the understanding and gain a different point of view, we can model the uncertainty of processes using stochastic models. Stochastic models are a tool for estimating probability distributions and predicting potential outcome. Specifically, a Markov decision process is set of mathematical tools for modeling decision making. It finds many uses in many disciplines like robotics and economics. It is also a core part of solving optimization problems. Optimizing solutions and automatic tasks through the usage of data and models is a core topic of machine learning. With machine learning we aim to create useful models that try to mimic human behavior. Reinforcement learning (RL) is a subset of machine learning. RL's main goal is to train an agent in choosing the optimal decisions in specific environments. Environments try to mimic scenarios in the real-world to simulate the interaction with the agent. With the combination of these three tools above we try to answer the following questions:

- Can we automate predictive process monitoring?
- Can the stochastic models help as further improvement in business processes?

1.2 Subject and Objectives of the Thesis

This thesis main subject is discovering new ways to analyze and predict business processes. Using data and process analysis with the use of process mining algorithms in order to have better data and process behaviors so the predictive analysis can provide better results. Investigating and experimenting with the addition of stochastic models, such as Markov Models, in order to emulate real world's randomness and decision making of individuals who execute the processes. This way it will be easier to detect inefficiencies such as design or operative mistakes whilst also have a better understanding and analysis of the process's results. Finally, with the use of Reinforcement learning an agent is trained to interact with the environment, which is set up as the business process, in order to find best ways to traverse through it while also trying to automate this process analysis.

1.3 Structure of the Diploma Thesis

In this thesis our first step was discovering the business process that was provided by the selected data set. Using different mining algorithms and selecting the one with the highest representation accuracy of the data, along with minimum complexity and unknown layers. After its decided which mined business process would be the one to choose, the next step is to analyze the correlation between the data and each state. With this information combined with the probabilities for each state based on the instances the next step is to create a Markov Model, known as Markov Chain. As the simplest form of a Markov Model, the Markov Chain simple feature was cycling through the business process's states based on the probabilities that were calculated based on all individual instances on each state. Since the Markov Chain is complete, as it is the baseline for the Markov Decision Process, the next step is to create a Markov Decision Process and add the element of decision. Now it is not solely random to decide the transition from a state to another but knowledge is added in the form of values for each state along with a rewarding system for each transition. In the last stage of this thesis, a reinforcement learning algorithm was used known as Q-Learning which does not use the probabilities to transition between states as seen before on the Markov Decision Process but "learns" based on the reward system how to traverse and find optimal paths between states. Deep Q-Network was also used as an extension in the reinforcement learning, which adds an agent that is trained in the specified environment and learns to transition in the business process using the best possible paths.

2.1 Business Process Management

2.1.1 BPM

Business Process Management is an emerged tool that's been adapting and evolving along with analytic tools and software. Understanding the importance of Business Process Management (BPM) might be easier compared to what actually is BPM and how you can distinguish it in its many appliances. For some, BPM is only about technology, for many it's about optimization based on methodologies such as TQM, Lean or Six Sigma [1]. For others, BPM is a way to design, monitor and execute processes, but in reality, BPM is a combination of all the above. [2]

The need of management for business processes came up simply, as anyone can understand, from the need of efficiency as well increase of profits since results and income are among the main concerns of a business. BPM gives organizations a way to combine its own needs along with customers' needs, organization's goals along with monitoring and execution of business processes.

When applied properly, BPM increases efficiency and productivity whilst reducing costs and errors. BPM is a way for any business to secure safe execution of procedures, protocols, resources and capital management but also provides an easy way to evaluate and optimize its own processes [3].

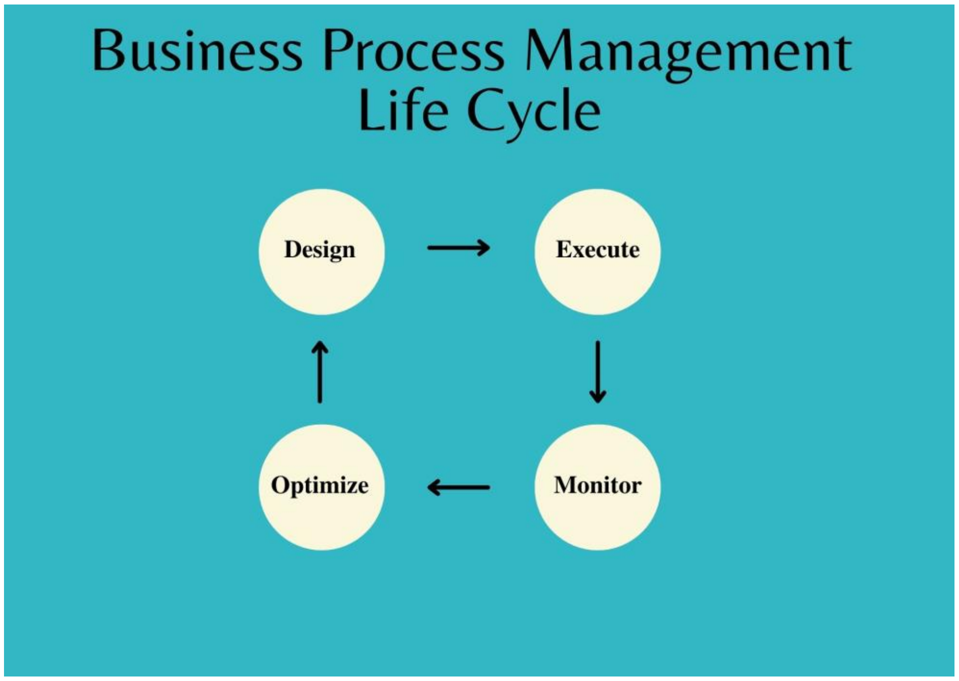


Figure 1 BPM Life Cycle

- **Design**: Designing new processes or identifying existing ones. Map the work flow between production and Execution.
- **Execute**: Enact processes designed in the previous step. This step can be performed manually or via automation tools.
- **Monitor**: Supervise processes' performance.
- **Optimize**: Utilize information from monitor phase to increase efficiency of processes. [4]

Despite all benefits, many businesses have had experienced difficulties implementing process management. In order to be able to implement BPM, there are some principles that business processes must follow, to be qualified as high-performance processes.

Process design: Designing a process is the first and most important step in order to have a high-performance process. Breaking down in points as of who, by whom, when and how so the process's steps are well defined, easy to understand and execute. Also, a good business process must have a well-defined starting point, ending point and a finite number of steps. The process must be designed in a way that can be run an indefinite number of times.

Process metrics: Process metrics consist ways to analyze the business goals along with customer needs. Supervising and keeping a balance between resources is key so we do not decline in one area when improving another.

Process performers: As mentioned previously in Process design, no matter how easy to understand and execute some processes might be, production line must consist of well-trained personnel with knowledge about business processes.

Process infrastructure: Appropriate tools must be provided on the production line in order to minimize errors and avoid mistakes (or scenarios which personnel has to act in his own judgment).

Process owner: In order to keep track of a process workflow it is important a Manager to be set with the responsibility and knowledge to guide the production line as well as adjusting resources based on chosen metrics.

It is very important that all principles mentioned above must be well designed and defined so business processes to be considered high-performance processes. [5]

2.1.2 Tools for BPM

As mentioned above, the importance of Business Process Management comes with difficulties which created the need for tools to improve ways of design and manage business processes. BPM tools are used to design a systematic approach and optimization of processes. Main uses are modeling, implementation and automatization of business workflows with main goal of improving performance while minimizing errors, inefficiencies and miscommunications.

Business Process Management software allows for non-IT specialists to build business workflows with an emphasis on connecting different systems. BPM tools provide visualization of modeling and design which makes easier even for non-technical users to design and test processes. A range of capabilities include:

- **Workflow management**: Users design, test and execute workflows to manage interactions between data, systems and employees.
- **Business rules engine**: Users can create a set of business rules and conditions.
- **Form generator**: This feat helps users without programming skills to build web-forms easily.
- **Collaboration**: The BPM tools provide discussion threads, ideas management and decision management on processes or workflows.
- **Analytics**: One of the most important steps for evaluation, which provides metrics and KPIs along with standard and custom reports.
- **Integrations**: Key integrations help businesses to use data between systems and interfaces.

Benefits of BPM include:

- **Simplifying operations**: reducing complexity also reduces errors and mistakes from employees.
- **Reducing inefficiencies**: helps minimizing the time of process completion.
- **Increasing accountability**: having a clear cut of who's doing what makes easier to monitor mistakes and amend of processes.
- **Cutting costs**: brining down costs increases profit.
- **Improving communication with customers**: customer feedback is very important for business.
- **Improving business agility**: no business can stay immutable, agility is key to competitiveness.

Most modern BPM tools provide cloud support, while many tools are still on-premises for specific reasons. Moreover, a variety of products offer both cloud and on-premises. Reasons for using on-premises products include:

- Keeping critical systems in-house,
- Focus on data security,
- Constrains on handling and storing sensitive data.

One of the well-known tools for BPM is ***Kissflow***. Kissflow is a software that enables businesses to reinvent their existing processes for digital optimization. Since it's a no-code development tool, it allows business employees to automate process flows, enforce business rules and provide solution for specific difficulties or problems that might occur without damaging the process' efficiency.

Reasons why Kissflow is favored, is because it's not restricted to structured and repeatable business processes but also supports cases and projects. Some of the well-known features include user-friendly dashboards, templates for custom reports and advanced design for forms and workflows. It can also integrate with other software solutions and standard productivity software.

Another tool that is up to the top for Business Process Management is ***ProcessMaker***. ProcessMaker is a BPM and workflow management software in one. It allows you to design, automate and deploy business processes and workflows. One thing that's unique about ProcessMaker is

1. open-source option,
2. on-premise option,
3. Cloud option.

Very few tools offer all three.
ProcessMaker is used by businesses, organizations, governments and educational institutes. Process design is done with BPMN v2.0 notations.

2.1.3 Process Simulation in BPM tools: BPMN

BPMN stands for Business Process Model and Notation and it is a language that appeared in 2004, developed by Business Process Management Initiative (BPMI), while later adopted as standard by the Object Management Group (OMG). The reason BPMN was developed, is to provide a standard way of graphical representation of business processes, setting graphic symbols with a specific meaning and the ability of combinations. The reason this was done, is to assist and provide a notation so different elements are easily distinguishable from each other and easy to understand by modelers. BPMN includes five specific categories:

1. **Flow objects**: are graphic elements to represent business processes, include three groups. Events, activities and gateways.
2. **Data**: is the information needed for the activities and includes four groups. Data object, collections of data, input and output data.
3. **Connection objects**: presents the way objects are linked and performance order of activities. There are three types of connections, sequence flows, message flows and associations.
4. **Swimlanes**: include two groups, pool and lane. Pools establish actors in processes and usually are sub-divided in lanes.
5. **Artifacts**: include two groups, group and notes which are used to supply additional information about the process.

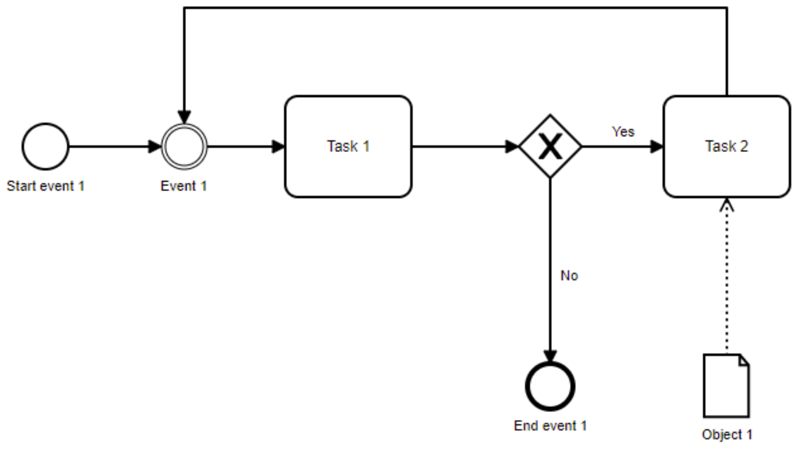


Figure 2 BPMN Business Process Diagram

BPMN was designed for graphical representation of business processes with no concerns about simulations. Since simulation is key to evaluate business processes, a set of extensions were defined to the BPMN language to allow process simulations. Elements needed to simulate a process are:

- Probabilistic distribution is needed to simulate a process since a process may be triggered many times during a period of time.
- Availability of resources is very important element in order to simulate a business process.
- Probabilistic duration is also needed in order to calculate and monitor the execution of a process.
- Multiple processes may be active simultaneously so division of resources has to be taken into account.
- Branches are divided after a gateway (decision point) so it needs to be represented by a probability.

While others argue, for a BPM tool to be fully operational in simulations, other aspects must be supported as well, such as:

- It is important to assign a stochastic value of unavailability to resources since it is not always certain that all resources can be used at all times.
- Intervals of unavailability for resources between completed or running tasks is important to be set.
- Prioritization of work resources is needed because FIFO approach is not realistic in real time execution of processes.
- Activity priority is very important, meaning activities with higher priority must have resources always available to them, even if it means other activities must be postponed or stopped temporarily.

Not long ago, in order to simulate a modeled business process, an IT specialist had to re-model and adjust the process to the specific language of the simulation tool. Such work is unnecessary doubled work, but more tools have started to emerge that allow simulation of business processes which are modeled in BPMN.

One common thing about these tools is that none of them originated from simulation tools vendors but from BPM tool vendors. So practically, simulation was an addition to BPM tools. Tools that deserve mention in the simulation addition for BPMs are:

- **Visual Paradigm**: When Visual Paradigm first came up, the only way to model a business process was with the use of UML (Unified Modeling Language). Only recently, this tool started to support BPMN, and with the use of the Enterprise version it provides simulation of business processes. One of the downsides for this tool is a missing feature that's important for simulations, the work cost to a resource.
- **BPSim**: BPSim is a tool developed by Trisotech, a company dedicated developing BPM tools. Setting aside the usage difficulties of this tool, it is very well equipped with assigning specific values and a wide variety of simulations which cover many real cases.
- **Bizagi**: Bizagi is considered one of the best BPM tools. Along with a very user-friendly interface, Bizagi also provides a very easy way to model BPMN diagrams with detailed representation of resources that business processes use. A feature worth mentioning is the What-if Analysis function. This feature allows the designer to clone any scenario and change any parameters where bottlenecks are found. The report produced compares original and cloned scenarios, providing a very comfortable way to monitor and re-design a business process.
- **BonitaSoft**: BonitaSoft is a very easy to use tool for graphical representations for business processes in BPMN, designing is very simple, with drag and drop elements in a canvas.
- **BIMP**: BIMP is known as a business process simulator, available to any user, running in an online server. BIMP, unlike other tools it does not support business process modelling. In order to simulate a business process, one must first design the process in a modeling tool with the feature to export as BPMN 2.0 or VSDX. Absence of such a feature, is a huge drawback for this tool, as is needed to use two different tools to design and simulate business processes.[6]

2.2 Process Mining

Process Mining uses a variety of techniques from both data science and process management. These techniques are used on event data most commonly called event logs in an attempt to make sense of the data. In more detail, we analyze the event data in order to find and analyze bottlenecks, compare process variants, discover processes, check compliance and suggest improvements [7].

There are three types of process mining that can be applied on an event log. These techniques can give us a different insight of the data.

First, process discovery, this technique takes as an input an event log with attributes such as case id (a unique identifier to recognize the case to which activity belongs) activity name (a textual description of the activity executed) and timestamp and produces a process model. The process model is a graphical representation of the process (Petri nets, BPMN diagrams, activity diagrams, etc.) [8].

Second, conformance checking, is a technique that compares the behavior of process model and the recorded event log of the same process. For this reason, conformance checking is used to detect, locate and explain deviations the differences between the event log and the process model. [8] [18]

Third, enchantment, with this technique the goal is to improve the existing model. One way to achieve that is to modify the model to bet fit the data, this is called repair. Another way is by cross-corelating the processes model with the log in order to get a different view of the model. [8]

2.2.1 Process mining techniques

Process Discovery is a data driven technique that uses an event log as an input and produces a process model without any a-priori information. This technique seeks to produce a process model that is as close as possible to the event data. As any method that uses data, the event logs could suffer from noise, incorrect data, partial data and so on. This problem makes process discovery perhaps the most important of the three techniques as it has to solve the data problems. Common use cases for process discovery include.

- **Structure of the process:** Understand the structure of an unknown process or discover what the process looks like in practice.
- **Routing probabilities:** Gain a better grasp of the process by examining the decision paths between choice points.
- **Most frequent paths:** Determine the path with the highest number of cases.
- **Distribution of cases:** Looking at the distribution of cases along possible routes find common and uncommon behavior. [17]

Conformance Checking takes as an input a process model and an event log and aims to find the differences between the behavior described in the event log and the process model. Two questions that we try to answer with this technique is whether the log data fits the model or the process model is an acceptable representation of the observed process. Two of the metrics that we can use are Fitness and Appropriateness. ***Fitness:*** a value that describes how well a model fits the log. ***Appropriateness:*** a combined metric of "structural appropriateness" and "behavioral appropriateness". Structural appropriateness refers to the simplicity and minimal structure of the model and behavioral appropriateness analyzes the balance between overfitting and underfitting. [7] [16]

Enhancement uses the recorded data to improve or extend a process model with the intent of adding a new perspective to the existing model. With the use of additional information like timestamps we can obtain information like bottlenecks, thruput times, slowest activities, longest waiting times, case arrival rate, Chronological order of events, resource utilization rate and cycles [17] [18].

2.2.2 Event logs

Without event logs we cannot apply any of the technique mentioned above. An event log can be viewed as a collection of cases. A case can be thought of as a trace/sequence of events. There are three minimum requirements to describe an event.

1. Case id: An identifier for events that refer to a specific case.
2. Activity: Refers to a step in a process (i.e., Create, Request, Approve)
3. Timestamp: Indicates when each of the activities took place.

Depending on how we want to analyze a case different column can represent several possible mappings.

For example, let's say that we have an event log with transaction data that contains the following columns cashier id (a unique id for the cashier that executed the transaction), customer id, payment method (cash or credit card), date time (timestamp) payment amount and product. We can use different columns depending on what information we are trying to extract. We can use the cashier id as the case id and payment method as an activity to measure the frequency of the cash vs credit card payments or we can take the product as a case id and the customer id as an activity. It all depend on what questions we are trying to answer. Some event logs have a timestamp for start of an activity and a timestamp for the end. With that information we can measure duration of an activity. Additional information that may be recorded can help with the understanding of the stature of a process.

Event data can be recorded in various formats, DB systems spreadsheets, csv files and many more. The standard for process mining is the extensible Event Stream (XES). The purpose of this standard is to provide a common scope and general syntax and semantics for the event data [19].

2.2.3 Process Mining Tools

Common tools used for process mining are α-algorithm, heuristic mining and inductive miner. These three tools can not only be used for process discovery but also for conformance checking. That is because the tools mentioned above can produce metrics such as precision, fitness, generalization, simplicity in addition to some form of a graphical representation.

1 Process models

Petri net is based on general net theory and is used for graphical representation, description and analysis of concurrent processes which can be found in distributed systems [12]. The representation is high level and compact [13]. There are four basic elements in a Petri net, Places represented by circles, Transition represented by rectangles, arcs/edges represented by arrows and tokens represented by smaller filled circles.

Business Process Model and Notation (BPMN): Is a widely used standard for representing business process models [14]. It allows for an easy-to-understand process models with capabilities of flat control flow perspective [15].

Directly-follows graph (DFG): A simple way of representing a process model each node is an activity. Activities are connected with arrows or arcs to show relationship between them [11]. In Contrast to BPMN notation cycle times and numbers are directly visualized on the graph.

2 Process mining algorithms

In more detail **α-algorithm** was the fist process discovery algorithm ever proposed. It has as an input an event log and constructs a Petri net explaining the behavior recorded in the log without any additional information [10]. α-algorithm examines the event log and find ordering relations between activities. Using the relations, it builds a footprint matrix. Then it converts the footprint matrix to a Petri net α-algorithm can distinguish 4 types of relationships between activities.

1. Direct Succession: x > y translates to if and only if
2. Causality: $x \rightarrow y$ if x > y and not y > x
3. Parallel x || y if x > y and y > x
4. Choice x # y if neither x > y and y > x

Example of traces footprint matrix and petri net.

Trace

< x, y, z, w >

<x, z, y, w>

	x	y	z	w
x	#	→	→	#
y	←	#	||	→
z	←	||	#	→
w	#	←	←	#

Table 1 Footprint matrix α-algorithm

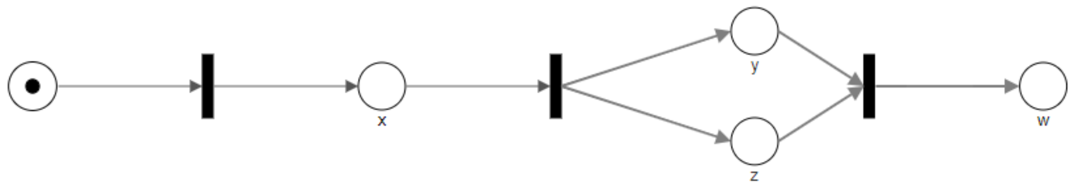


Figure 3 Petri net example

α-algorithm also has some limitations. It cannot detect short loops and it cannot distinguish between required and implicit places and this could cause additional non-essential places in the discovered Petri net [20].

Heuristic Miner is an algorithm that calculates frequencies of the activities, it can detect short loops and it can also detect skipping activities. It is best used in high frequent behavior. This as a result has infrequent paths not incorporated into the model. Using heuristic mining on low frequent behavior has the risk to also catch noise [21]. Heuristic miner uses a footprint matrix that holds the frequency of the relationship between activities and then builds a dependency matrix using the formula:

Where A: activity $\frac{|A1>A2|-|A2>A1|}{|A1>A2|+|A2>A1|+1}$

Using the following trace, we can compute the 2 matrices.

10 < x, y, z, w >

25 <x, z, y, w>

	x	y	z	w
x	0	10	25	0
y	0	0	10	10
z	0	25	0	25
w	0	0	0	0

Table 2 Footprint matrix heuristic miner

	x	y	z	w
x	0	0.9	0.96	0
y	-0.9	0	-0.41	0.96
z	0.96	0.41	0	0.9
w	0	0	0	0

Table 3 Dependency matrix

The calculated values are called heuristics values and they refer to the strength of the correlation between activities. The heuristic values can range from [-1, 1]. The higher the value the stronger the correlation between the activities. In heuristic miner we can set a parameter called Dependency threshold, this sets a limit so the algorithm holds only strong connections. Usually anything above 90%.

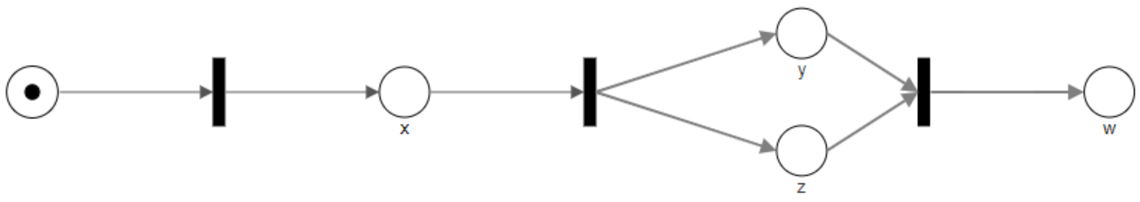


Figure 4 Petri net Dependency threshold 90%

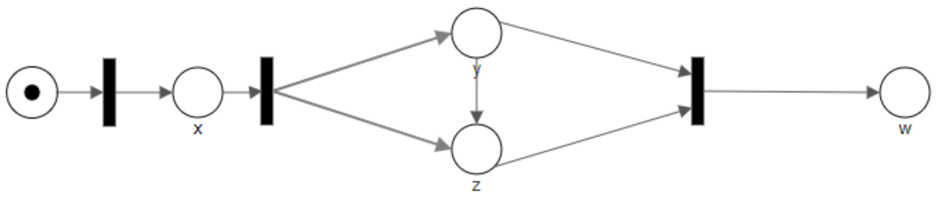


Figure 5 Petri net Dependency threshold 40%

Inductive Miner has many variants it is flexible and guarantees soundness and fitness of the input log [9].

This algorithm is an improvement of the α-algorithm and heuristics miner. Inductive miner ensures soundness using process trees. The process tree is created by repeatedly splitting the event log. It finds the most likely split. For example, we have the event log

< x, y, z, w >

< x, z, y, w >

We can see that we always start with activity x and end with activity w.

< x, y, z, w >

< x, z, y, w >

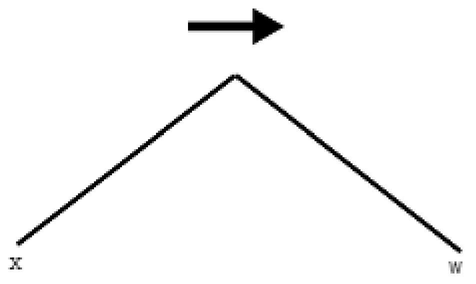


Figure 6 Process Tree first split

Then we analize the more complex behavior of the event log

<y, z>

<z, y>

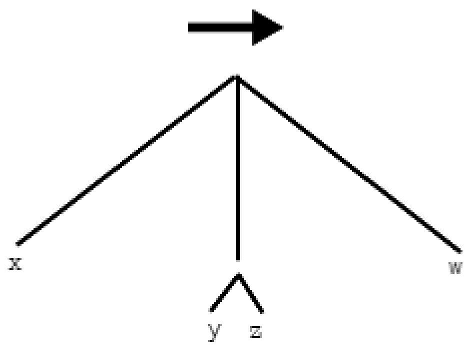


Figure 7 Process Tree final split

Application of these tools provides us also useful statistics and a process model of the case. Metrics such as fitness, precision generalization and simplicity.

- Fitness: Refers to how well the captured model represents the behavior of the event data
- Precision: Measures the allowed behavior of the model that is not seen in the event log
- Generalization: This metric estimate if the model is capable of replicating behavior of future traces.
- Simplicity quantifies the complexity of the process model.

2.2.4 Goals of process mining

Process mining has many applications in many domains from Finance, Sales, Trading, IT Services, Management and many more. Regardless the application domain, the goal is to make data-driven decisions by identifying inefficiencies, like bottlenecks and waiting times, gain a deeper understanding of the processes and build models that represent what is really happening. Process mining has the purpose of discovering, monitoring and improving processes. It Intends to increase efficiency increase automation and understanding of organizations. It is a versatile tool that can be used both individually and in collaboration with existing ones.

2.3 Markov Models

Markov Processes are probabilistic models which focus on the concept of states and state transitions. To define a complex system with the help of a Markov model, it is important to firstly specify each state in the process and every probability of transitions.

Markov Models are stochastic models used to model pseudo-random change systems. It is labeled as pseudo-random because choices are based on probabilities. There are many types of Markov Models, **Markov Chains** are the simplest type of Markov Models. Markov Chains traverses the states in systems with random variables that change according to time. [22]

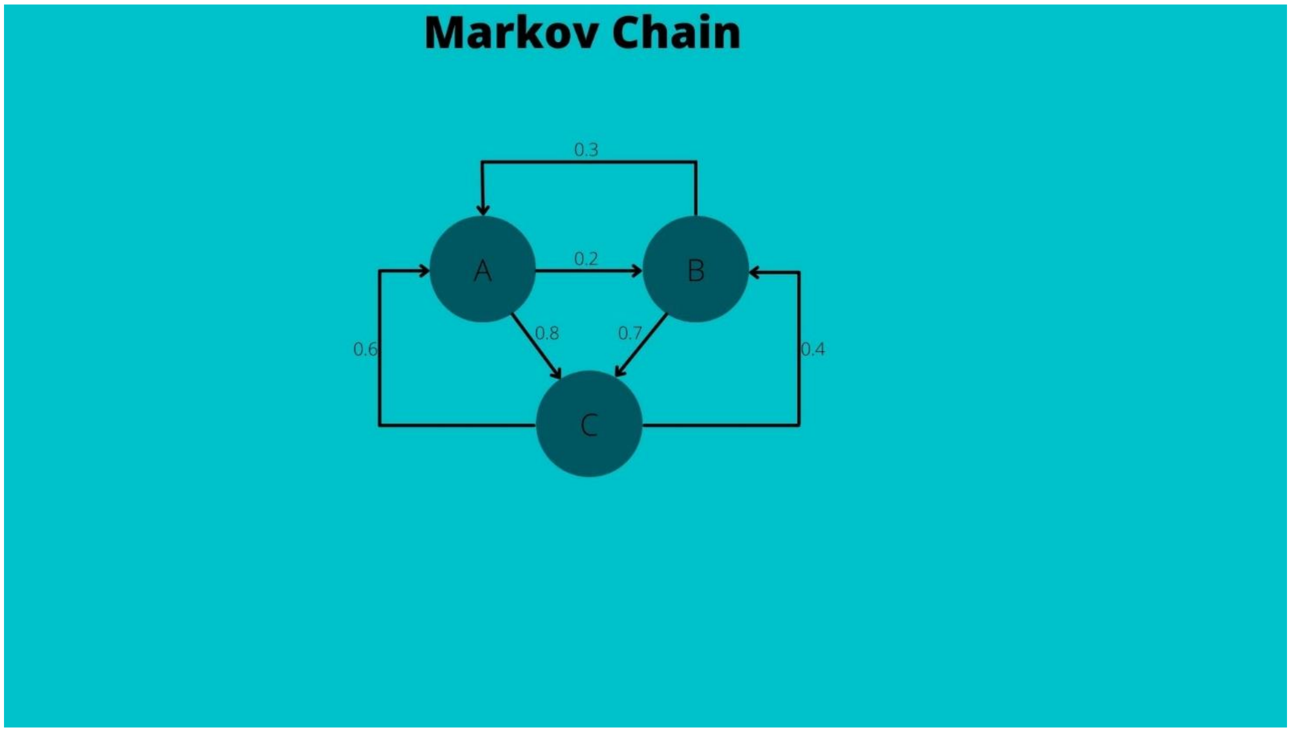


Figure 8 Markov Chain example

Markov Chain example's probability matrix is P:

	a	b	c

a	0	0.2	0.8
b	0..3	0	0.7
c	0.6	0.4	0

Table 4 Markov Chain example probability matrix

Another type of Markov Model are **Hidden Markov Models**, which are similar to Markov chains but the key factor that differentiates them are the hidden states. Hidden states can't be seen directly but only from processes that use or depend on them [23]. One common use of Hidden Markov Models is speech recognition. For the application of Hidden Markov Models, mostly speech processing, both stochastic and deterministic. There also are cases, which a hidden semi-Markov model (HSMM), a statistical model with similar structure as a hidden Markov model with difference that the unobservable process is a semi-Markov rather than a Markov, meaning the probability of being a change in the hidden state depends on the amount of time that has elapsed since the transition to the current state. Hidden Markov models are defined as:

- X_n and Y_n as discrete-time stochastic processes with $n \geq 1$. The pair (X_n, Y_n) is considered a Hidden Markov Model only if
- X_n is a Markov Process of which behavior is not directly observable.
- $P(Y_n \in A \mid X_1 = x_1, x_2, \ldots , X_n = x_n) = P(Y_n \in A \mid X_n = x_n)$.

The goal is to compute the probability of a particular output sequence, which requires summation of all possible state sequences.[24]

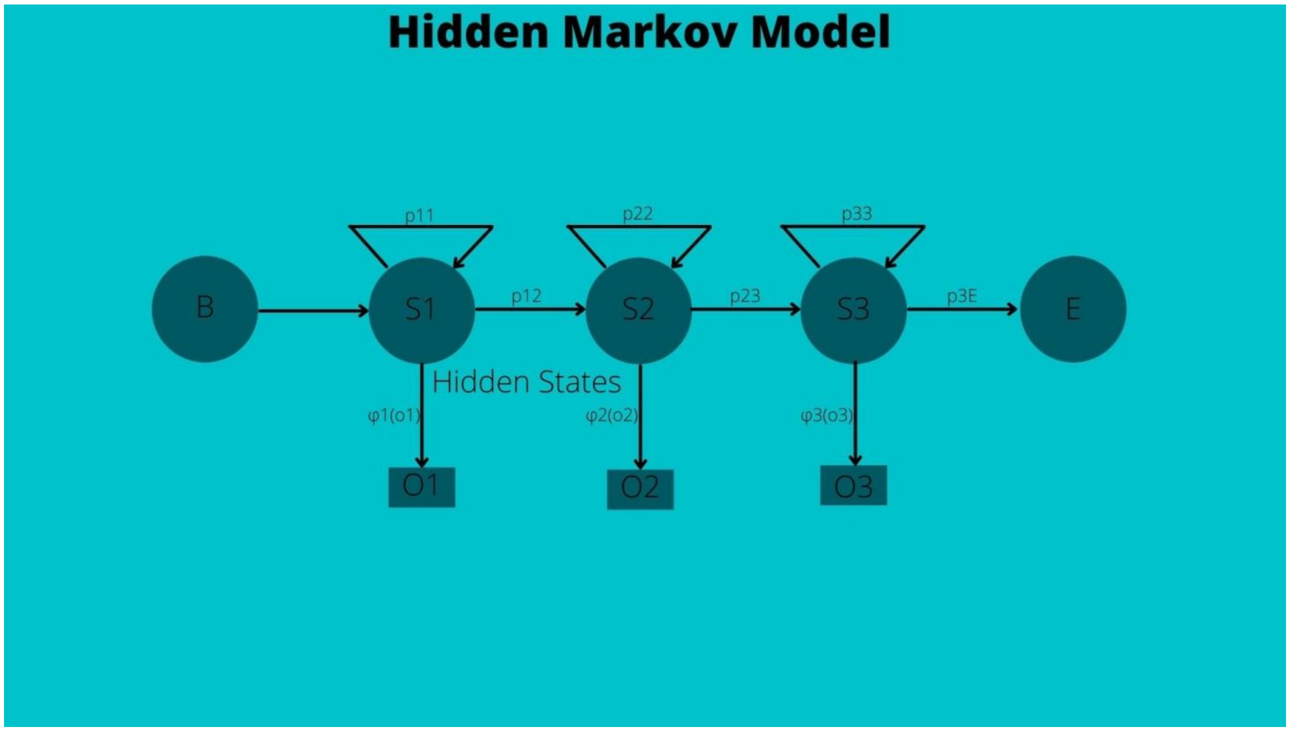


Figure 9 Hidden Markov Model example

In addition, **Markov Decision Processes** (MDP) are discrete-time stochastic control processes. It provides a decision making model in problems where outcomes are partly random and under the control of a decision maker. A Markov Decision Process consists a 4-tuple (S, A, P_a, R_a), where:

- S consists the states that form the state space,
- A consists the actions which are the action space. There can also be A_S which contains certain actions that are available on state S,
- $P_a(s,s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability which with the action *a* in state *s* at time *t* will lead to state *s'* at time *t + 1*,
- R_a is the reward after completing a transition from state *s* to *s'*, with the action *a*.

The number of states and actions can be finite or infinite. Some processes with infinite states and actions can be reduced to finite state and action processes. The main purpose of Markov Decision Processes is to find a good policy which represents the optimal path of the process.

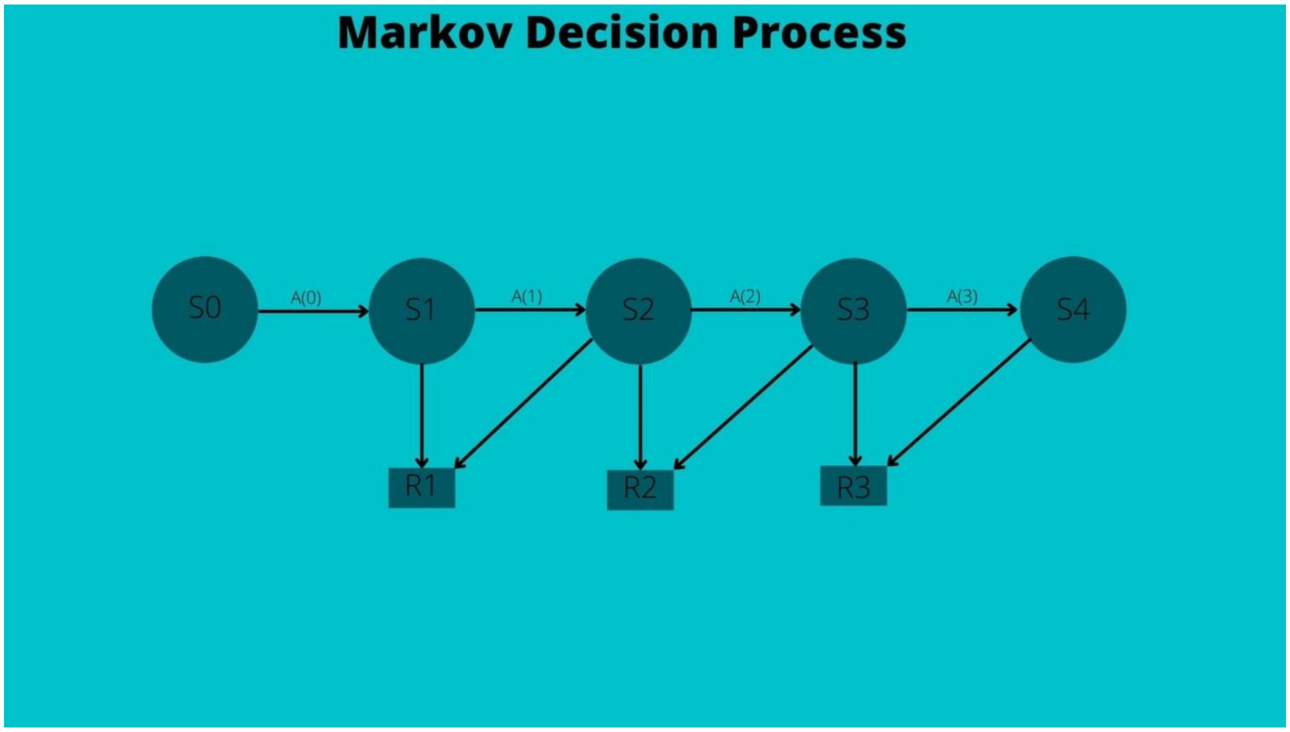


Figure 10 Markov Decision Process example

Furthermore, 1 (POMDP) are Markov Decision Processes which the states of the system are only partially observed. **Partially Observable Markov Decision Processes** are known as NP complete, and only recently they've been used on controlling agents or robots.[25] Discrete-time POMDP models consist a 7-tuple(S, A, T, R, Ω, O, γ) where:

- S consists a set of states
- A consists a set of actions,
- T consists a set of conditional transition probabilities between states,
- R: $S \times A \rightarrow \mathbb{R}$ is the reward function.
- Ω consists a set of observations,
- O consists a set of conditional observation probabilities,
- $\gamma \in [0,1]$ is the discount factor.

The goal for the agent is to pick actions in order to maximize the expected future discounted reward $E[\sum_{t=0}^{\infty} \gamma^{t} * r\_t]$ where r_t is the reward earned at time *t*. At each time period t each time period *t*, when the environment is in a state $s \in S$, the agent chooses an action $a \in A$ which causes a transition to state *s'* with the probability T(s' | s, a). Also, the agent receives an observation $o \in \Omega$ which depends on *s'*, *a* with probability O(o | s', a) and in the end the agent receives a reward *r* = R(s, a). This process is repeated until the goal is met.

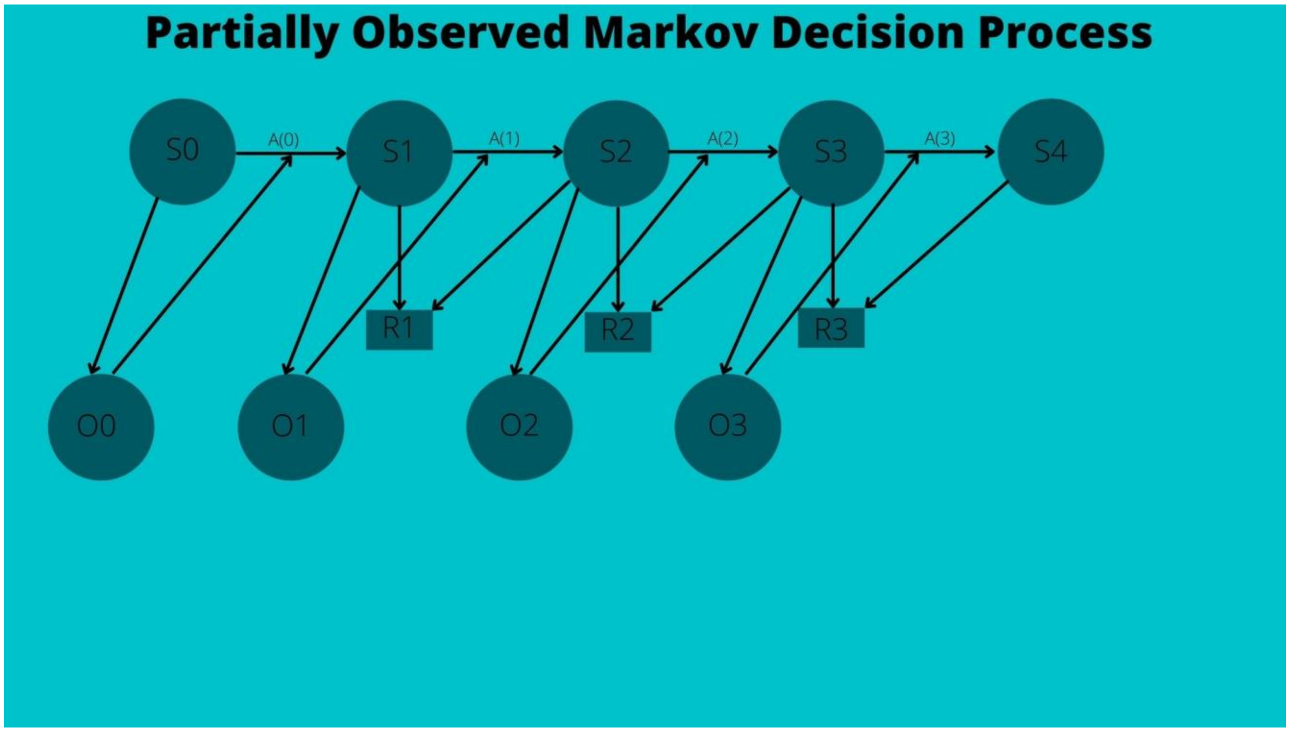


Figure 11 Partially Observable MDP example

Moreover, **Markov Random Fields** (MRF), or Markov networks, are undirected graphical models and is considered by some to be Markov chains in multiple dimensions. A Markov network represents dependencies the same way as a Bayesian network, with the only difference being that Bayesian networks are directed and acyclic, while Markov networks are undirected and sometimes might be cyclic. In a Markov Chain a state depends only on the previous state in time, whereas in Markov Networks a state depends on its neighbors. Markov random fields are random fields which fulfill the Markov properties and are defined as:

- An undirected graph G = (V, E)
- Random set of variables $X = (X_v)_{v \in V}$
- Markov Properties
    - Pairwise: Two non-adjacent variables are conditionally independent given all other variables, $X_u \perp\!\!\!\perp X_\upsilon \mid X_{V \setminus \{u,\upsilon\}}$.
    - Local: A variable is conditionally independent of all other variables given its neighbors, $X_u \perp\!\!\!\perp X_{V \setminus N[u]} \mid X_{N(u)}$. Where N(u) is the set of neighbors *u* and $N[u] = u \cup N(u)$.
    - Global: Any two subsets of variables are conditionally independent given a separating subset, $X_A \perp\!\!\!\perp X_B \mid X_S$. Where every path from *A* to *B* passes through *S*.

The Global Markov property is the strongest among the properties, following second is the Local Markov property and last the Pairwise one. [26]

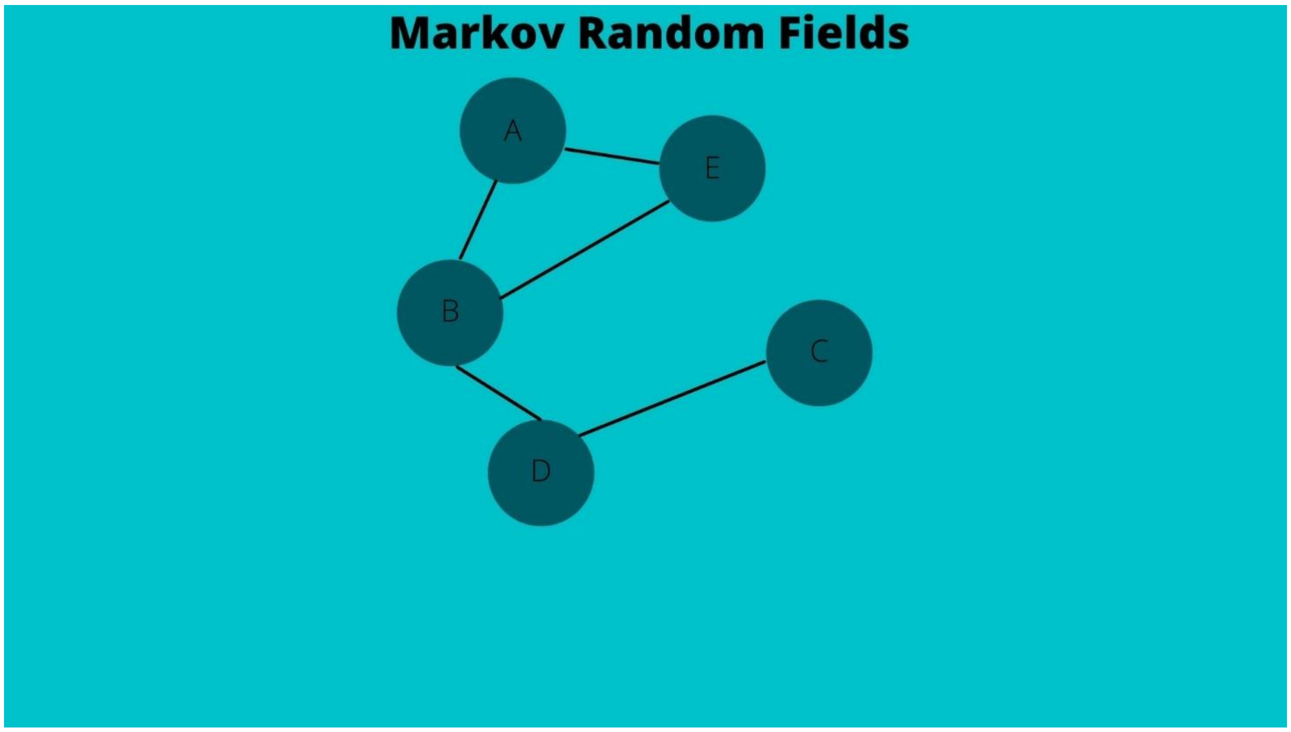


Figure 12 Markov Random Fields example

Some other honorable mentions of Markov Models are Hierarchical Markov models which are used to analyze abstract human behavior [27], Tolerant Markov Models as probabilistic-algorithmic Markov Chain Model [28] and Markov-chain forecasting models that are used as forecasting methods such as wind power with discretizing time series [29].

2.4 Machine Learning

Machine Learning (ML) is a subset of Artificial intelligence (AI) designed with the purpose that an agent can resemble human like intelligence. [34] As humans learn by the surrounding environment, observation of behavior, examples and analogy this field of study tries to make an agent build models based on data. ML has three main subcategories.

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In Supervised learning (SL) the machine learns form examples from labeled data sets. In more detail we provide the input-output pairs (X, Y) and we try learn the mapping function. The goal is to approximate the function so that it works on new data and we can make accurate predictions [33]. It is most commonly used for Classification and Regression problems. Popular algorithms include Linear regression, random forest and Support Vector machines (SVMs).

In contrast to SL, Unsupervised Learning uses an unlabeled training set. Namely you have only input X and no corresponding output. The purpose is to better understand the data by identifying features and/or find a model that represents the data like a probability distribution [33]. Common uses are Clustering, Association and Anomaly detection. Some of the methods used consist of k-means DBSCAN, principal component analysis (PCA) techniques, Local Outlier Factor (LOC), etc.

The basic idea of Reinforcement Learning is that an agent can interact with an environment in order to maximize a reward function. In relation to SL, RL does not need labelled data, an agent must explore the environment and find balance between exploration and exploitation. Reinforcement Learning has many uses in Games, Robotics, Trading, Recommendation Systems, etc. Some commonly used algorithms are Q-learning, SARSA and DQN. [30] [31] [32] [33]

2.4.1 Reinforcement Learning Methods

The typical way to model a problem in reinforcement learning if by defining three sets:

1. States of the environment, S
2. Actions the agent can perform, A
3. Short term rewards, R

In Reinforcement learning the agent learns to make sequential decisions through training. The goal of the agent is to maximize the reward throughout an episode. An episode is a sequence of state, action and rewards that ends at a terminal state. The process of training uses a reward system. The aim of the reward system is to adequately reward the agent for the performance of right actions or punish it for choosing wrong ones. Models of RL can be describes as MDPs. RL is convenient for problems with extensive duration like board games backgammon checkers and Go [36] [37]. Before moving on to the solution methods, we need to mention some terminology.

- Policy (π): is the strategy the agent follows to dictate the next action given the current state.
- Discount factor γ (gamma): a number between [0, 1) that determines the importance of the future rewards. If the discount factor reaches or surpasses 1 in a problem without a terminal state or in case that we cannot reach the terminal state. The undiscounted rewards can become infinite [33]. If the discount factor is 0 the agent cares only for the short-term reward hence it becomes short-sighted.
- Value function (V): The expected long-term reward with discount.
- Learning rate: a factor that demines the rate of overriding old knowledge with newly acquired one.

At the very start of training most of the times the action – reward process is trial and error. When the agent accumulates knowledge regarding state-action pairs as well as rewards for each pair, it is possible to get stuck into certain actions, even if they are not the optimal ones. This is the exploration vs exploitation problem; it is a well-studied problem in finite space MDPs [35]. Exploration allows for further experimentation with the hope of improving current knowledge. Exploitation chooses the greedy action the one with the most reward. If the environment has not been thoroughly explored this can lead to sub optimal policies and results.

1 ε-greedy

One method to combat this is the ε-greedy. Where ε is a parameter controlling the exploration vs exploitation rate. In more detail ε the probability of choosing explore vs exploit. We have the following function where:

$0 < \varepsilon < 1$ and p is a random probability distribution $action = \begin{cases} random\ action\ (a)\ p < \varepsilon \\ max\ Q(a) \quad else\ 1 - \varepsilon \end{cases}$

2 Value methods

Value-based methods are based on MDP theory and they require a full model of the environment. In this approach the agent's goal is to find a policy that optimizes the expected return. If the policy produces the best expected return from any initial state is considered to be optimal. The value function $V_\pi(s)$ is defined as the expected long-term return of the current state s following the policy π. Value function V is expressed as:

$$V_\pi(s) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\, | s_t = s, \pi\right]$$

- E: expected value operator
- γ: discount factor
- π: policy

The optimal value function returns the maximum expected reward from all other value functions.

It is expressed as:

$$V_*(s) = \max_\pi V_\pi(s)$$

3 Policy methods

In Policy methods we try to find the best policy without the use of a value function. The agent learns directly from the policy. That means we try to optimize the policy π directly. There are two types of policies, deterministic and stochastic. A deterministic policy maps states to actions for a given state return an action. Deterministic policies are used in environments with no uncertainty. In Stochastic policy the output is a probability distribution over the actions. Stochastic policies are most commonly used.

Stochastic Policy: $\pi(a|s) = P(A_t = a|S_t = s)$

Monte Carlo Methods

Monte Carlo (MC) method is a mathematical technique that is used for modeling the uncertainty of a system. In contrast to value methods, Monte Carlo methods do not require full knowledge of the environment. MC methods require only experience. They sample and average return for each state-action pair. In the classical policy we alternate between policy evaluation and policy improvement [36]. We start with a policy π_0 then we evaluate and get $q_{\pi 0}$ and then we improve to get the new policy π1. These steps are continued until we converge to the optimal policy.

Temporal difference methods

Temporal difference (TD) learning is a combination of dynamic programing and Monte Carlo. In TD methods the agents can directly learn from experience without a model of the environment. Also, they can learn from bootstrapping meaning they can update based on partial learning.[36] In the other methods the agent updates the rewards values when the reach the terminal state. But in TD instead of calculating all the feature reward it is trying to calculate the immediate reward. To achieve that TD uses discounted return.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

Reward at time t is the combination of future discounted rewards. This makes future reward have less value. Another important tool is the TD error (δ_t). It is the difference between optimal reward V_t^* and current predicted value V_t.

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

The value function for the TD methods can be written us:

$$V(S_t) \leftarrow \underbrace{V(S_t)}_{Value} + \underbrace{a}_{learning\, rate} \underbrace{[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]}_{TD\ error}$$

Q-learning

Q-learning is a reinforcement learning algorithm. It is model free algorithm and it is based on temporal difference. In the case of Q-leaning the function Q that is action-value directly approximates the optimal function q^*, regardless of the policy that is being followed. To perform q-learning we need a q-table. Q-table is a matrix of size $state \times action$. The q-table stores the maximum expected future reward for the actions at each state. We can either initialize the q-table with a policy that we are trying to improve or no policy. After the initialization we chose an action by using the ε-greedy method. Then we perform the chosen action we evaluate the observed outcome and reward. Last, we update the q-table. To carry out this procedure the algorithm uses the Q-function.

$$Q(S_t, A_t) \leftarrow \underbrace{Q(S_t, A_t)}_{old\ value} + \underbrace{a}_{learning\ rate} \underbrace{[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)]}_{new\ value\ with\ TD}$$

Q-learning is sufficient for small state spaces. The performance of the algorithm drops significantly as the state space becomes more complex.

Deep Q-learning

With Deep Q-learning we go beyond simple reinforcement learning. Deep Q-learning is an algorithm that combines Deep Leaning (DL) and RL. A simplistic way to think what deep q-learning does, is to replace the q-table with an artificial neural network (ANN). In the case of deep q-learning we do not use value iteration, instead we use a function approximator to find an estimate of the optimal Q-function. ANNs are used because they are excellent at approximating function. The combination of deep neural networks and q-learning is called DQN. To solve this problem, we need no explain the concepts of experience replay and target network. In experience replay the agent stores experience into a memory. When a memory reaches a certain threshold, the agent learns from it. The agent learns from randomly selecting uniformly distributed samples of the stored memory. It learns from batches. The selections are random to avoid biased decisions. The target network provides stability to the training. The target-Q values that will be utilized to compute the loss for each action during training are generated by this second network. The updating of the target network should be frequent and slow. Deep Q-leaning requires large amount of data to perform better than other techniques. [38][39][40]

2.4.2 Challenges of reinforcement learning

Reinforcement learning has many challenges. The most important perhaps, is the creation of the simulation environment. The performance of all the methods used, the adequate exploration depends on the environment. Another important challenge is finding a local optimum and getting stuck there. Even more challenges arise for real world problems. Let us consider a navigation problem of a multi-agent system like a car or a robot in the real world. A main concern is the exploration of the environment, but in the real world the agent and the environment can be damaged. So, we need to consider both the safety and cost problems of the agent and the environment. In order to do so we need to restrict the freedom of movement and interaction with the environment. Things do not always go as simulated. A simulation can miss certain elements of the environment or make them behave in a different way. There is always a gap between reality and simulation. Another problem is the acquisition of sufficient data and observation. An environment can be restrictive by nature example of this can be medical operations and trials. Despite the challenges reinforcement learning is promising field with a lot to offer.

3 Markov Models in Predictive Monitoring of Business Processes

3.1 Dataset Analysis

Most Datasets are in XES format. XES is an XML-based format, with the acronym eXtensible Event Stream. In the XES standard, there are four guiding principles:

- **Simplicity**: Information is represented in the simplest possible way, XES logs should be easy to parse and generate.
- **Flexibility**: XES standard is capable of capturing even logs from any background, no matter the application of the observed process.
- **Extensibility**: Extension of the standard should be as transparent as possible while maintaining backward and forward compatibility.
- **Expressivity**: XES should encounter as little loss information as possible, so it is striving for a generic format which elements are strongly typed with human-interpretable semantics to them.

The dataset must also be converted into a csv/xls file in order to have a better understanding of the dataset and its elements. The next step is to analyze, separate and distinguish the elements of the dataset, along with the states that are used in the business process.

After the analysis of each element's contribution in the process execution, our next step lies in the process mining tools in order to have a better understanding of the business process's structure.

3.2 Process Mining Tools

As mentioned in the process mining section one of the first steps is process discovery. To help with the understanding of the model three algorithms and two types of visualization were used. The algorithms used are α-miner, inductive miner and heuristic miner along with metrics, as for visualization Petri Nets and Directly follows graphs were used.

The metrics used to analyze and choose which mining algorithm suits better the case are

- **Log Fitness**: Representation of capture model regarding the event data.
- **Precision**: Measures the unseen behavior of the model in the event long.
- **Generalization**: Estimation of replicating capability for future traces.
- **Simplicity**: Quantification of the process model's complexity.

In order to decide based on metrics which algorithm gives us the best results, it is important to understand which the importance order of those metrics is. ***Log fitness*** is a key factor in deciding which algorithm modeled process fits best the event log, as the name indicates. Aside fitness, ***Generalization*** comes second as replication and use of the modeled process for future traces and execution tests are helpful for a better understanding of the process.

Based on the metrics' results along with visualized processes of each corresponding algorithm, we can decide which algorithm has the best approximation of the business process.

.

3.3 Statistical Analysis

Usage of statistics can provide us with useful insight to analyze and evaluate performance, predict future behavior and create more efficient practices. Through the discovery and process of the data both in XES and csv formats we can produce metrics for this particular case. It is very important to calculate frequency of activities because it will be used later to calculate the transition probabilities used in the Markov Chain as well as in the Markov Decision Process.

3.4 Process model - modification

After exploring data, it is important to check if all cases reach a final state, if not it would prove problematic in the calculation of the Markov model probabilities. It is suggested to create a new state for all cases that do not reach terminal states and terminate them in that state. Probabilities used on the Markov Models are calculated using all individual instances on each state divided by the complete number of instances: $P(s) = \frac{instances(s)}{instances}$.

3.5 Markov Chain

Business Process Management as great as it is in design, analysis and optimization of business processes it lacks the factor of uncertainty. Uncertainty can be caused by many things, varying from system failures to individual mistakes that can cost a lot of time and resources to organizations. A model that can manage uncertainty is the Markov model, more specifically a Markov Chain. Markov Chains models uncertainty by adding a new element, the probability of change.

In order to create a Markov Chain, we used an object-oriented programming approach to create a Markov Chain Class, named ***MarkovChainClass***.

One of the most common ways is for the ***MarkovChainClass*** class to contain at least 3 functions:

- **Initialization Function**: The *Initialization* function which creates a Markov Chain object. Firstly, it must create a matrix of all unique states along with a matrix of probabilities. With those two matrices we create a Transition matrix which matches each state transition with its probability.

    - **States matrix**: A matrix that contains all unique states of the business process.
    - **Probability matrix**: A probability matrix that contains all transition probabilities from-to for each state.
    - **Transition matrix**: A combination of two previous matrices which helps with the simulation of the Markov Chain.

- **Next state Function**: The *Next State* function's main reason is to select the next state from the available states based on the current state, usually with the help of a random choice function that takes into account probabilities as well.

- **Generate States Function**: The *Generate States* function is used to create a pool of transitions starting from a state, named as current state, for as many instances needed with the use of the *Next State* function.

3.6 Markov Decision Process

3.6.1 Markov Decision Process Description

As previously mentioned, **Markov Decision Processes (MDP)** are decision making models in problems where outcome is partially random. MDPs are created based on the Markov chain, with the addition of *choice*, thus explaining the feature of the decision maker. The solution of MDPs is a Policy, finding the best policy, also known as best path, which is the optimal path of the MDP which accumulates the highest reward.

3.6.2 Markov Decision Process Solution

In order to calculate the value regarding each state, we used Bellman's equation

$V(s) := \sum_{s'} \left(P_{\pi(s)}(s, s')(R_{\pi(s)} + \gamma V(s')\right)$ which calculates the value for each state, based on the rewards of the next available states.

A function to calculate the ***Values*** must contain the following inputs:

- The ***Values*** function needs to be recursive.
- ***S***: number of unique states.
- States of the MDP (states): States contains a 1x***S*** matrix with all unique state names.
- Current state (cs): Current state to calculate its value.
- First state (ss): The starting state for the MDP.
- Probability list (prob): Prob contains a ***S***x***S*** matrix with all probabilities from-to accordingly for each state.
- Reward list (reward): Reward contains a ***S***x***S*** matrix with all rewards from-to accordingly for each state.
- Discount factor γ (gamma): $0 \leq \gamma \leq 1$.

With the previously mentioned elements along with Bellman's equation it is simple to create the ***Values*** function that gives as output the value for the chosen state.

After the values for each state are calculated, the next step is to create a function to calculate the **Best Policy *(π)***, which is the optimal path that provides the highest sum of rewards based on

$\pi(s) = argmax_a\{\sum_{s'} (\mathrm{P}(s'|\, s,\ a)(\mathrm{R}(s'|\, s,\ a) + \gamma V(s'))\}$

The **Best Policy** formula seeks to accumulate the highest sum of state values. Beginning from the *first state*, transitioning every time on the next available state with the highest value until it reaches the *final state*. The path of states chosen along with the values' sum is the **Best Policy** for this business process.

3.7 Visualization of Markov Models

Since the Markov Chains can be complicated and hard to visualize with automated tools, it is suggested firstly to use the mined process as a guideline in order to design a manmade Markov Chain graph. A Markov Chain should resemble the example shown below:

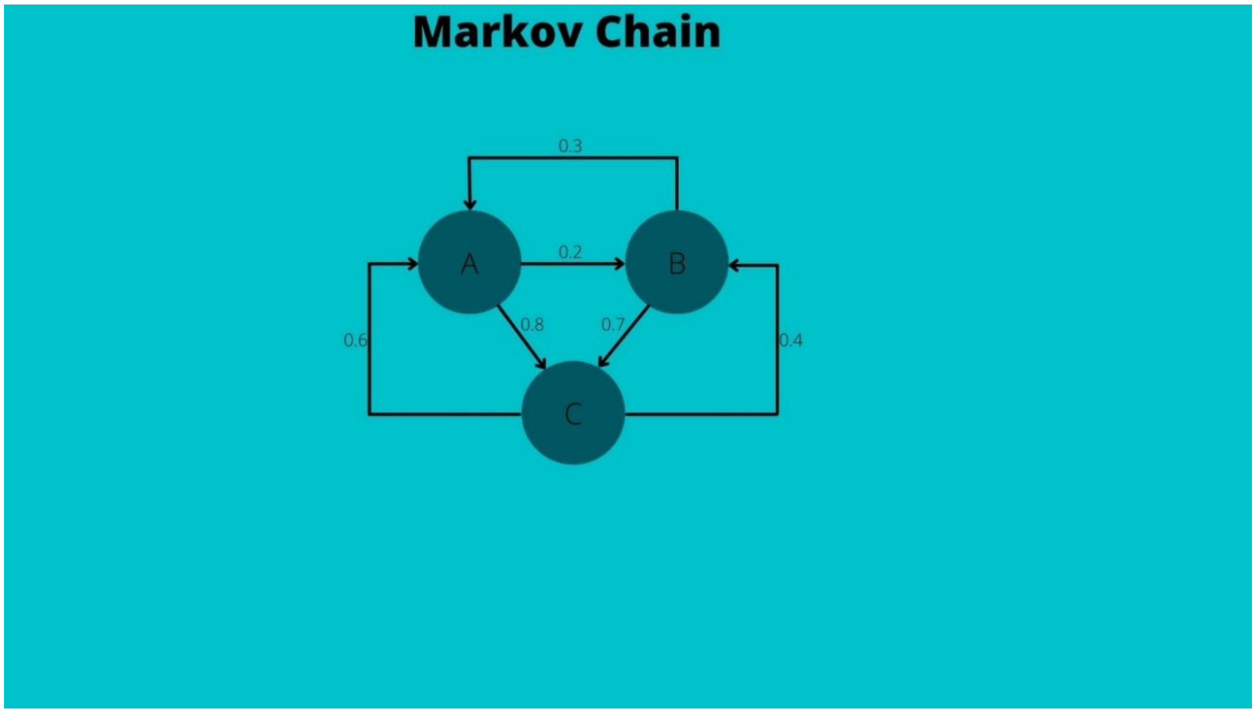


Figure 13 Markov Chain Example

The states are A, B and C with a probability on each transition.

After the completion of the Markov Chain graph, the next step is to design the Markov Decision Process based on the Markov Chain but with the addition of *rewards* for each transition. A Markov Decision Process should look like the example shown below:

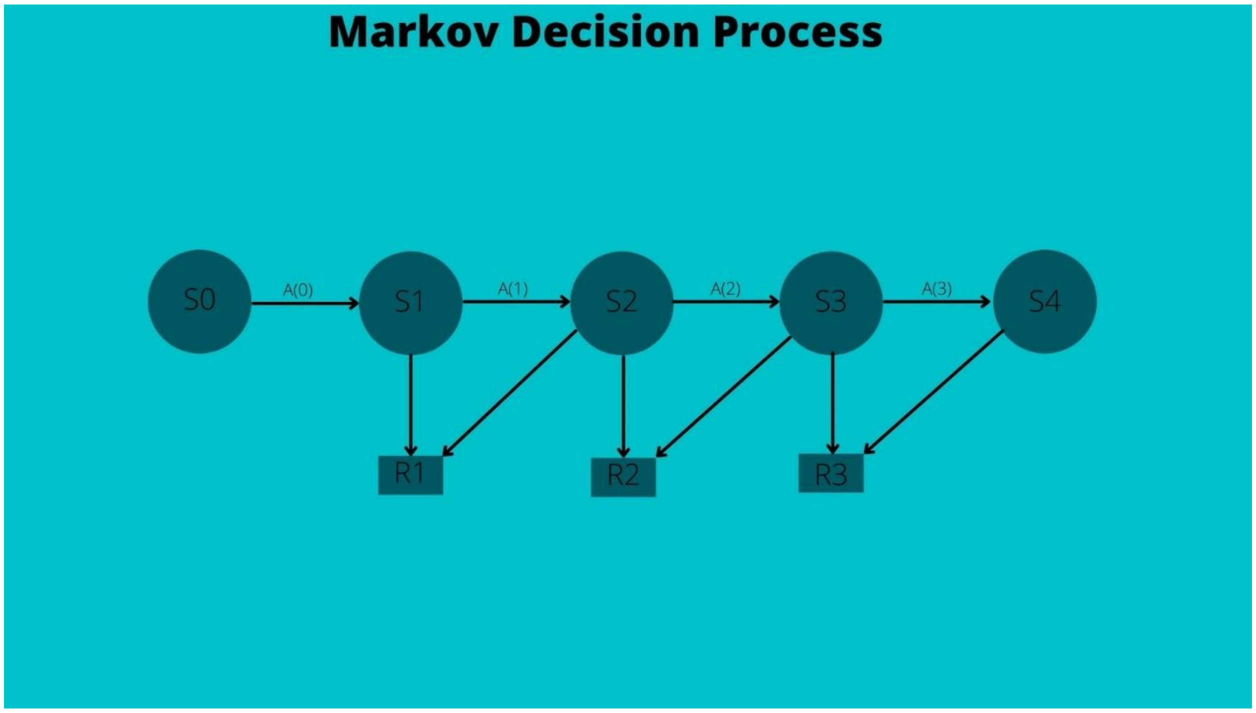


Figure 14 Markov Decision Process Example

The states are S0, S1, S2, S3 and S4. Each action from the *Action pool* displayed as A(0)…A(3) and the rewards for transitioning on each State , R1…R3. As it is demonstrated each state accumulates its own rewards along with the rewards of the previous state, as will do the next state until the *final state*.

4 Reinforced Learning in Predictive Business Process Monitoring

4.1 Dataset Analysis

Most Datasets are in XES format. XES is an XML-based format, with the acronym eXtensible Event Stream. In the XES standard, there are four guiding principles:

- **Simplicity**: Information is represented in the simplest possible way, XES logs should be easy to parse and generate.
- **Flexibility**: XES standard is capable of capturing even logs from any background, no matter the application of the observed process.
- **Extensibility**: Extension of the standard should be as transparent as possible while maintaining backward and forward compatibility.
- **Expressivity**: XES should encounter as little loss information as possible, so it is striving for a generic format which elements are strongly typed with human-interpretable semantics to them.

The dataset must also be converted into a csv/xls file in order to have a better understanding of the dataset and its elements. The next step is to analyze, separate and distinguish the elements of the dataset, along with the states that are used in the business process.

After the analysis of each element's contribution in the process execution, our next step lies in the process mining tools in order to have a better understanding of the business process's structure.

4.2 Process Mining Tools

As mentioned in the process mining section one of the first steps is process discovery. To help with the understanding of the model three algorithms and two types of visualization were used. The algorithms used are α-miner, inductive miner and heuristic miner along with metrics, as for visualization Petri Nets and Directly follows graphs were used.

The metrics used to analyze and choose which mining algorithm suits better the case are

- **Log Fitness**: Representation of capture model regarding the event data.
- **Precision**: Measures the unseen behavior of the model in the event long.
- **Generalization**: Estimation of replicating capability for future traces.
- **Simplicity**: Quantification of the process model's complexity.

In order to decide based on metrics which algorithm gives us the best results, it is important to understand which the importance order of those metrics is. ***Log fitness*** is a key factor in deciding which algorithm modeled process fits best the event log, as the name indicates. Aside fitness, ***Generalization*** comes second as replication and use of the modeled process for future traces and execution tests are helpful for a better understanding of the process.

Based on the metrics' results along with visualized processes of each corresponding algorithm, we can decide which algorithm has the best approximation of the business process.

4.3 Statistical Analysis

Usage of statistics can provide us with useful insight to analyze and evaluate performance, predict future behavior and create more efficient practices. Through the discovery and process of the data both in XES and csv formats we can produce metrics for this particular case. It is very important to calculate frequency of activities because it will be used later to calculate the transition probabilities used in the Markov Chain as well as in the Markov Decision Process.

4.4 Process model - modification

After exploring data, it is important to check if all cases reach a final state, if not it would prove problematic in the calculation of the Markov model probabilities. It is suggested to create a new state for all cases that do not reach terminal states and terminate them in that state. Probabilities used on the Markov Models are calculated using all individual instances on each state divided by the complete number of instances: $P(s) = \frac{instances(s)}{instances}$.

4.5 Reinforcement Learning

4.5.1 Introduction

In order to implement a reinforcement learning algorithm, we need an environment and an agent. The environment is represented by a set of actions and states. In our case the unique activities of the event is a set of states and the selection of the next state is the possible action. Rewards are chosen by the developer based on what needs to be maximized. The agent is trained through the algorithm that interacts with the environment that is set.

4.5.2 Q-learning

The simplicity provided by the Q-learning obligates you to construct a solution tailored to your particular problem. Firstly, we initialize the Q-table, then the agent chooses and enacts an action from the action pool, accumulates the rewards and finally updates the values of the Q-table. We need a reward matrix, an initial state and three functions.

- **Available Actions:** Create a pool of available actions using the reward matrix each element of the matrix corresponds to a state. Non-negative rewards correspond to available actions negative rewards correspond to unavailable actions.
- **Next Action**: Randomly choose an action from the pool of available ones.
- **Update:** Evaluate each action and update the Q table based on Q-function.

We set initial conditions and loop through a number of episodes to train the agent.

4.5.3 DQN

First, in DQN we need to create a network and a target network. Then, choose an action using with respect to the exploration vs exploitation dilemma. Finally, we need to update the network’s weights. In more detail, the tradition DQN uses a three convolution layers with Rectified Linear activation function (ReLU). The reason for this particular architecture is the need for handling images as an input. In our case the input data is numerical so we used four dense layers. Three Dense layers with ReLU activation and a dense layer with linear activation. For the creation of the environment and managing the reinforcement learning tasks we used the gym library. To create an environment using gym we need a class that at least contains the following attributes:

- **action space**: Is the set of possible actions, after observing the data
- **observation space**: Is the current state of the environment
- **step function**: Performs a single action, returns the reward and the next state.
- **render function**: Used to visualize the environment, display graphics etc.
- **reset function**: Sets environment to the initial state

5 Implementation

5.1 Technical implementations

Python is one of the most popular languages used today with simplified syntax and a variety of libraries. It is very common to use in data analytics and ML projects.

The Anaconda platform was used due to its easy navigation and management of the different environments. It comes packaged with spyder IDE and jupyter notebook. **Spyder** is an open-source IDE for data analyst engineers and scientists.It also offers inspection of variables, visualization and debuging capabilities. The **jupyter projects** supports for many programming languages including **JupyterLab** which is a web-based IDE. The usage of cells makes it ideal for data cleaning transformation and machine learning.

Pm4py is a library created by Fraunhofer Institute for Applied Information Technology (FIT). The library was used for the process mining part. In more detail, reading the XES files and extracting useful information with the use of process mining algorithms (α-miner, inductive miner and heuristic mine). Finding performance related information and creating process models in the form of Petri Nets and DFGs.

Numpy was used in conjunction with other libraries like pm4py and pandas. The information extracted from the dataset was used to calculate the probabilities for the Markov models. Numpy is essential for many scientific domains. It is powerful because it combines the ease of writing of python and the computational power of the C language. It is also key component for the visualization ecosystem of matplotlib. Matplotlib was used for creating the plots and figures.

Pandas is an open-source library designed for data analysis. This library was used for data exploration, statistics and permutations of dataset.

NetworkX is a package used for creating and studying the properties of complex networks and data structures like graphs. It was used for visualizing the Markov models; Markov chain and MDP.

TensorFlow is a collection of tools and libraries designed for ML development. Build on top of **TensorFlow** is **keras** which is an api for deep learning. **Keras-rl** implements reinforcement learning algorithms and integrates with the keras library. It also works with openAI Gym. These libraries were used for the creation on ANN and DQN agent.

Gym is an open-source library that was developed for creating and evaluating reinforcement learning algorithms. It is compatible with libraries like TensorFlow. It has a built in a collection of problems for experimentation and exploration. Also, it allows for the creation and management of custom environments. In our case it was used for the creation of the custom environment.

5.2 Data Preparation

5.2.1 Dataset Analysis

The dataset used in this case, comes from the ***BPI Challenge 2017*** and specifically *the Offer Event log*. This event log provides all application filed in 2016, which contains 1,202,267 events pertaining to 31,509 loan applications. For these applications, a total of 42,995 offers were created. We will be focusing on the business process that manages the 42,995 offers. The dataset's original form was in XES but it was also converted into a csv/xls file in order to have a better understanding of the dataset and its elements.

The elements contained in the csv file were:

Element name	**Description**
Action	Action taken in the business process.
Org:resource	User/Actor from the organization.
Concept:name	Business process State name.
EventOrigin	Origin of business process (Offer).
EventID	The unique identifier of the event.
Lifecycle:transition	Transition of state (complete).
Time:timestamp	Given time at each state.
Case:concept:name	The unique identifier of the event.
Case:MonthlyCost	The monthly costs to be paid by the customer to reimburse the loan.
Case:Selected	Boolean that indicates whether an offer is signed by the customer or not.
Case:ApplicationID	The identifier of the application.
Case:FirstWithdrawalAmount	The initial withdrawal amount.
Case:CreditScore	The credit score of the customer. The higher the credit score, the higher the client trustworthiness.
Case:OfferedAmount	The loan amount offered by the bank.
Case:NumberOfTerms	The number of payback terms.
Case:Accepted	The offer is acceptable based on Bank's terms.
OfferID	The unique identifier of the offer.

Table 5 Elements of the dataset

The Offer states are:

State name	**Description**
O_Create offer	Creating a credit offer.
O_Created	Offer created.
O_Sent (online only)	Offer sent online.
O_Sent (mail and online)	Offer sent online and by mail.
O_Returned	Client submitted documents for the offer.
O_Accepted	Application passed all checks and verification.
O_Cancelled	Offer canceled by the client.
O_Refused	Offer canceled by the bank.

Table 6 States of the dataset

5.2.2 Process Mining Tools

As mentioned in the process mining section one of the first steps is process discovery. To help with the understanding of the model three algorithms and two types of visualization were used. The algorithms used are α-miner, inductive miner and heuristic miner. As for visualization Petri Nets and Directly follows graphs were used.

	Alpha miner	Inductive	Heuristic
Percentage Fit traces	0.0	100.0	38.31143156180951
Average trace Fitness	0.838870808159509	1.0	0.9085864081648142
Log Fitness	0.8353714081519499	1.0	0.9136617651369092
Precision	0.812482961156979	0.7800641202426917	1.0
Generalization	0.9909298059055299	0.9833047865820989	0.7988654429631288
Simplicity	0.45454545454545453	0.6296296296296295	0.5769230769230769

Table 7 Process Mining algorithm results

As dictated from the results above the best metrics come from the inductive miner and this is also shown in the process models produced by each algorithm.

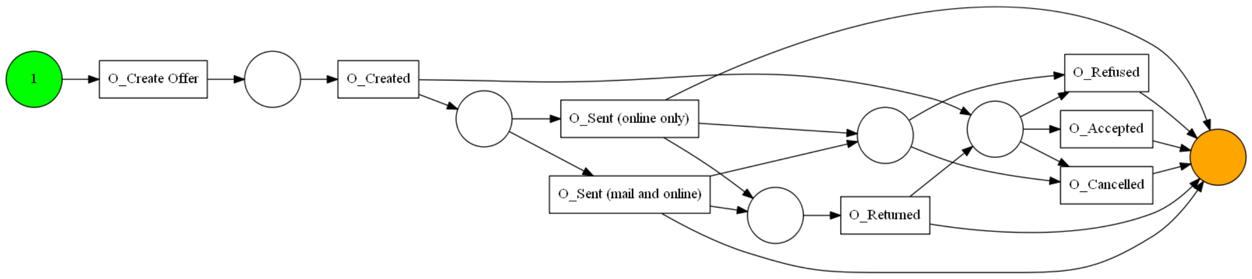


Figure 15 A-miner Petri net

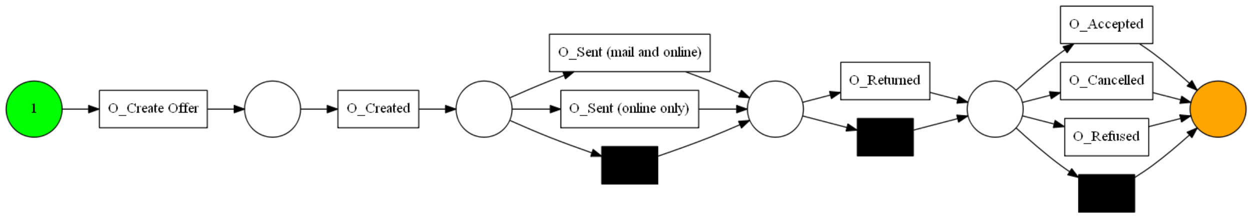


Figure 16 Inductive miner Petri net

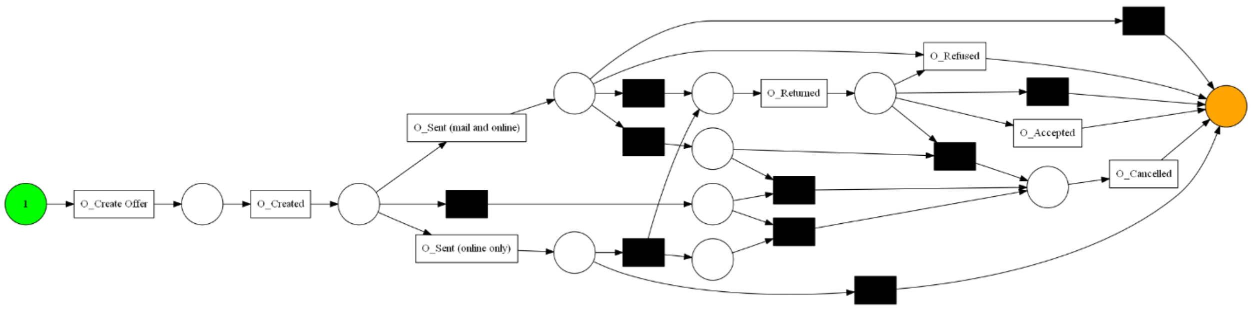
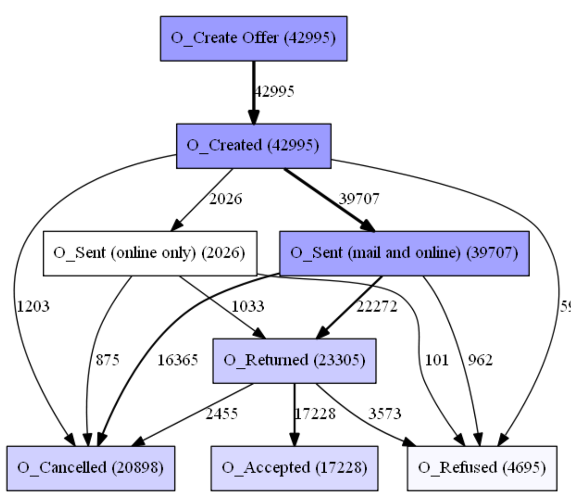


Figure 17 Heuristic miner Petri net

We can observe that even though the number of events is not that big the models can become complex and somewhat hard to read with the arcs crossing sometimes. The other type of visualization is DFG.

We have two DFGs one based on frequency and one based on performance.



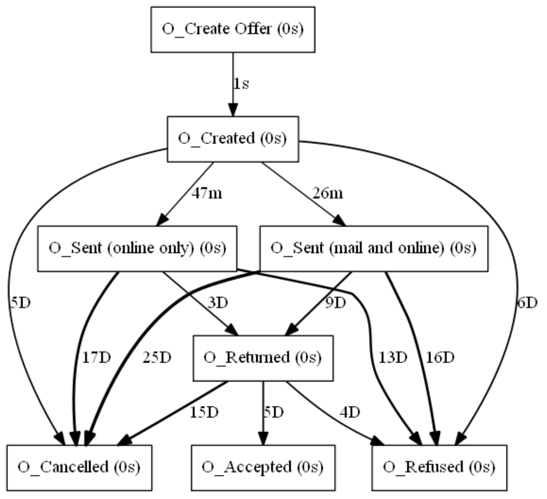


Figure 18 Performance DFG

Figure 19 Frequency DFG

Also, a Petri net produced form DFG.

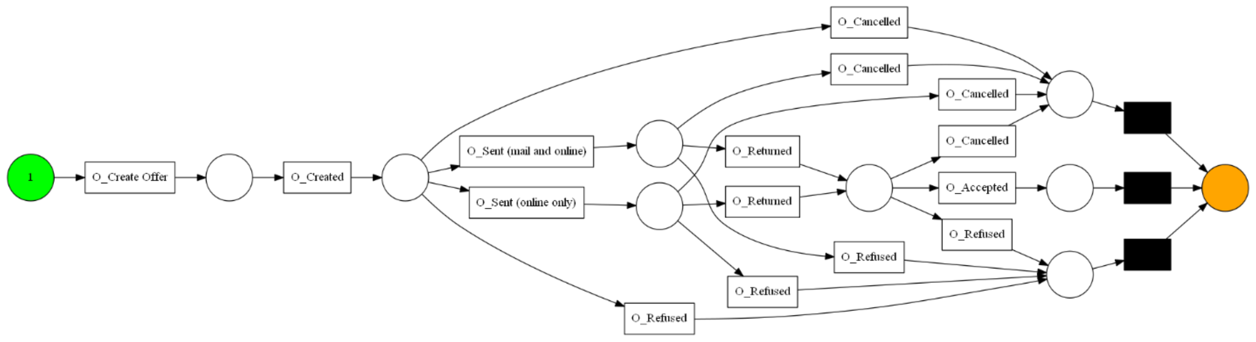


Figure 20 DFG-mine Petri Net

5.2.3 Statistics

Usage of statistics can provide us with useful insight to analyze and evaluate performance, predict future behavior and create more efficient practices. Through the discovery and process of the data both in XES and csv formats we can produce metrics for this particular case.

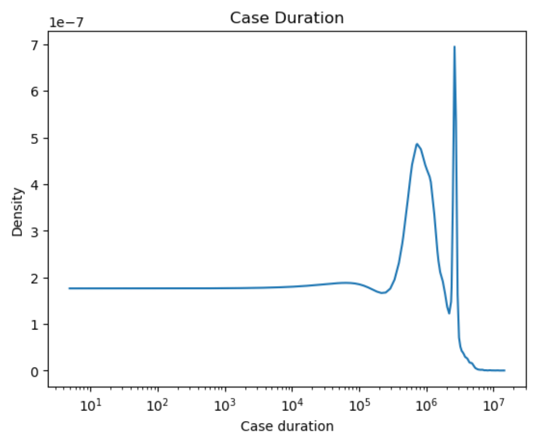


Figure 21 Case duration in seconds

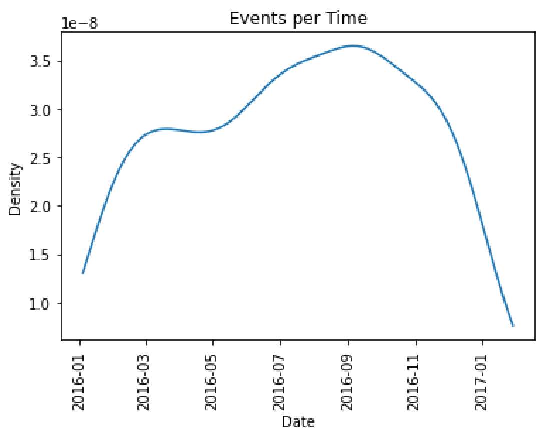


Figure 22 Event per time

Furthermore, we can calculate useful metrics

- Mean Case Duration in days: 19.1
- Median Case Duration in days: 15.9
- Case arrival Ratio in minutes: 13.3
- Case dispersion Ratio in minutes: 13.3

Frequency of activities is very useful information as it is used later for calculating the probabilities for the Markov chain and the Markov model.

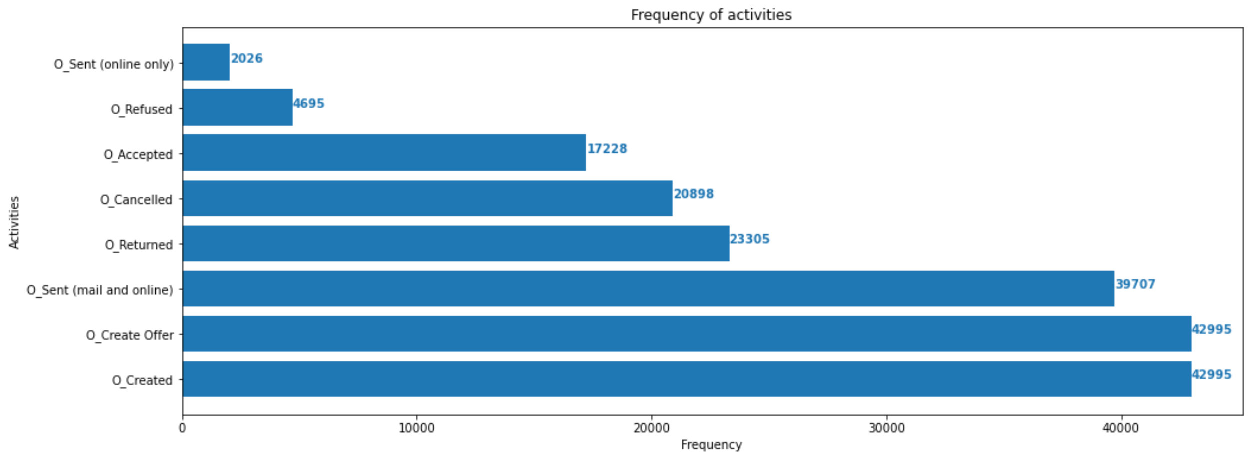


Figure 23 Activity Frequencies Bar Chart

To show association between the data elements of the dataset we created a correlation matrix. Its usefulness becomes clear later as we create a classification model.

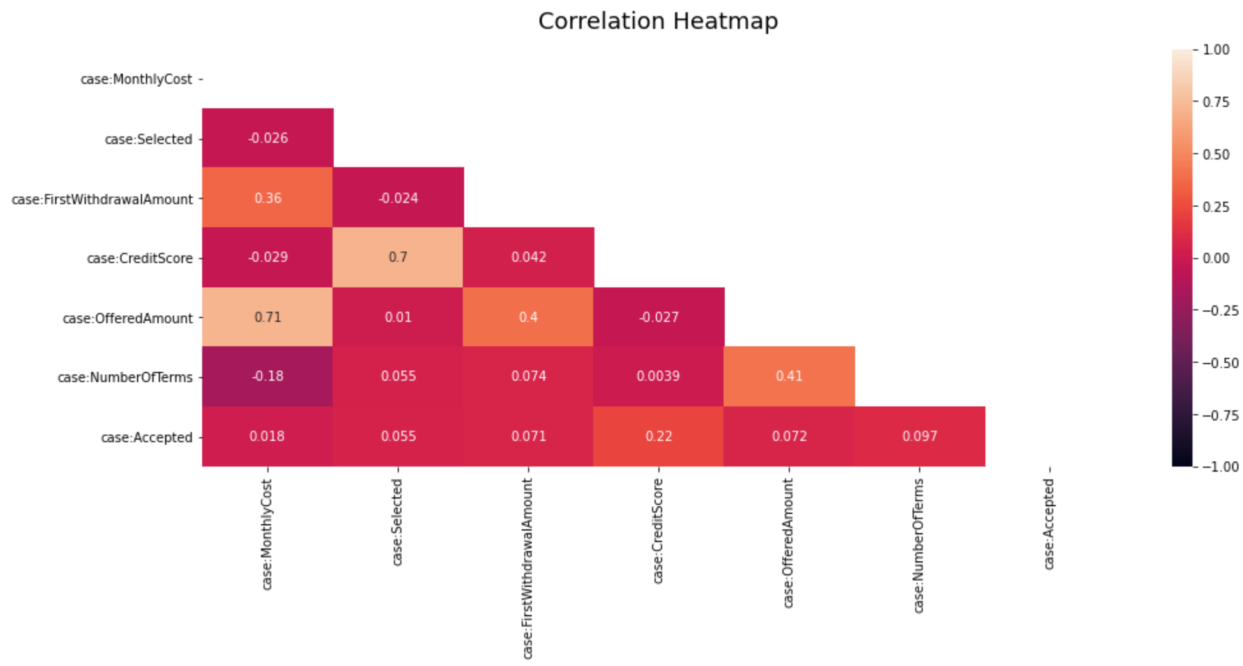


Figure 24 Correlation Matrix

5.2.4 Process model - modification

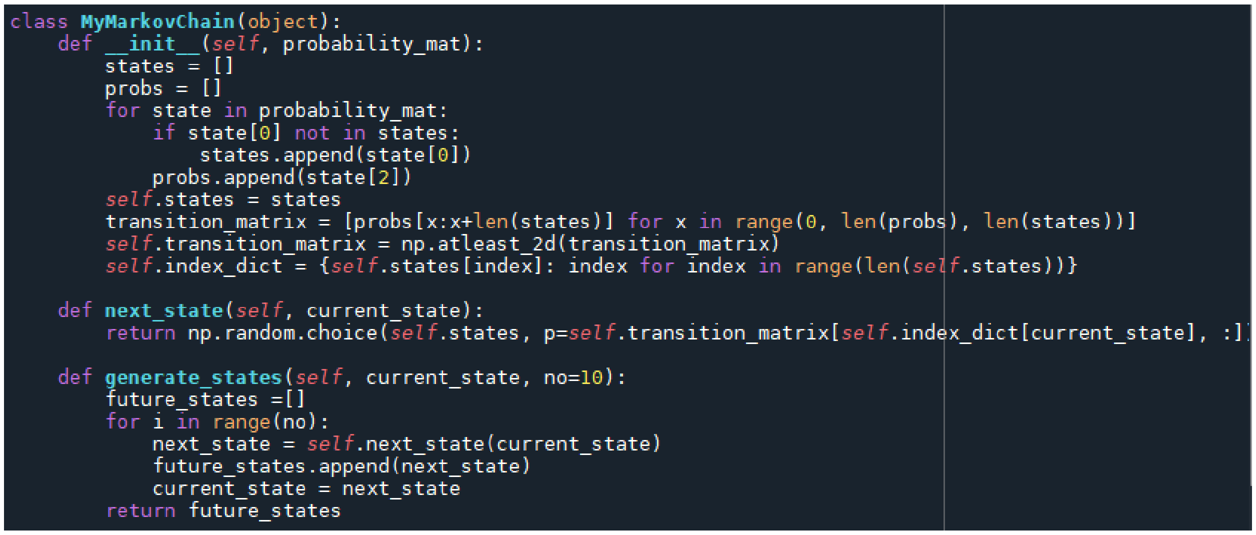
After the data exploration, we came to the conclusion that some cases never reached a terminal state. This would prove problematic in the calculation of the probabilities for the Markov models. Thus, we created a new stated called "Frozen" for all unfinished cases. We also noticed that all cases from the “O_Created_Offer” state transitioned to “O_Created” state, so we removed “O_Created_Offer” for simplicity matters. Probabilities used on the Markov Models are calculated using all individual instances on each state divided by the complete number of instances: $P(s) = \frac{instances(s)}{instances}$.

5.3. Markov Model Implementation

5.3.1 Markov Chain

A model that can manage uncertainty is the Markov model, more specifically a Markov Chain. Markov Chains models uncertainty by adding a new element, the probability of change.

The Markov Chain Class we built is:



```python
class MyMarkovChain(object):
    def __init__(self, probability_mat):
        states = []
        probs = []
        for state in probability_mat:
            if state[0] not in states:
                states.append(state[0])
            probs.append(state[2])
        self.states = states
        transition_matrix = [probs[x:x+len(states)] for x in range(0, len(probs), len(states))]
        self.transition_matrix = np.atleast_2d(transition_matrix)
        self.index_dict = {self.states[index]: index for index in range(len(self.states))}

    def next_state(self, current_state):
        return np.random.choice(self.states, p=self.transition_matrix[self.index_dict[current_state], :]

    def generate_states(self, current_state, no=10):
        future_states =[]
        for i in range(no):
            next_state = self.next_state(current_state)
            future_states.append(next_state)
            current_state = next_state
        return future_states
```

Figure 25 Markov Chain Class

There are 3 functions in the ***MyMarkovChain*** class:

- **Init**: The initialization function which takes as input a MyMarkovChain object and a probability matrix. It creates a list with all unique states and a list with probabilities, named *states* and *probs* accordingly. Also, based on the two previous matrixes it creates a *Transition matrix* which is 8x8 with probabilities matching all unique states transitions while *index_dict* is a dictionary with all unique states and available states for transition.
    - **Probability matrix**: The probability list's each row consists of ['From State', 'To State', probability].
- **Next_state**: This function selects the next state based on the current state with the help of *np.random.choice* to choose the next state randomly based from given choices on the probability matrix.
- **Generate_States**: Creates a pool of transitions from the current state to the next state, for a number of times as *no*, the default number of transitions is 10.

Example Run given its first current state is O_Created:

Next state: *O_Sent (mail and online)*

Cycle: *['O_Sent (mail and online)', 'O_Returned', 'O_Accepted', 'O_Created', 'O_Sent (mail and online)', 'O_Cancelled', 'O_Created', 'O_Sent (mail and online)', 'O_Returned', 'O_Cancelled']*

5.3.2 Markov Decision Process

Markov Decision Process Description

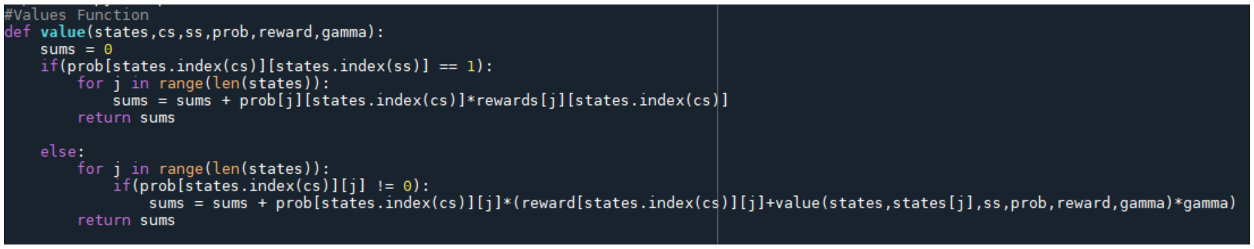
As previously mentioned, **Markov Decision Processes (MDP)** are decision making models in problems where outcome is partially random. MDPs are created based on the Markov chain, with the addition of *choice*, thus explaining the feature of the decision maker. The solution of MDPs is a Policy, finding the best policy, also known as best path, which is the optimal path of the MDP which accumulates the highest reward.

We've created 2 solutions regarding MDPs.

Markov Decision Process Solution

This solution features a function based of the Bellman's equation

$V(s) := \sum_{s'} \left(P_{\pi(s)}(s,s')(R_{\pi(s)} + \gamma V(s')\right)$ Which calculates the value for each state, based on the rewards of the next available states.



```
#Values Function
def value(states,cs,ss,prob,reward,gamma):
    sums = 0
    if(prob[states.index(cs)][states.index(ss)] == 1):
        for j in range(len(states)):
            sums = sums + prob[j][states.index(cs)]*rewards[j][states.index(cs)]
        return sums

    else:
        for j in range(len(states)):
            if(prob[states.index(cs)][j] != 0):
                sums = sums + prob[states.index(cs)][j]*(reward[states.index(cs)][j]+value(states,states[j],ss,prob,reward,gamma)*gamma)
        return sums
```

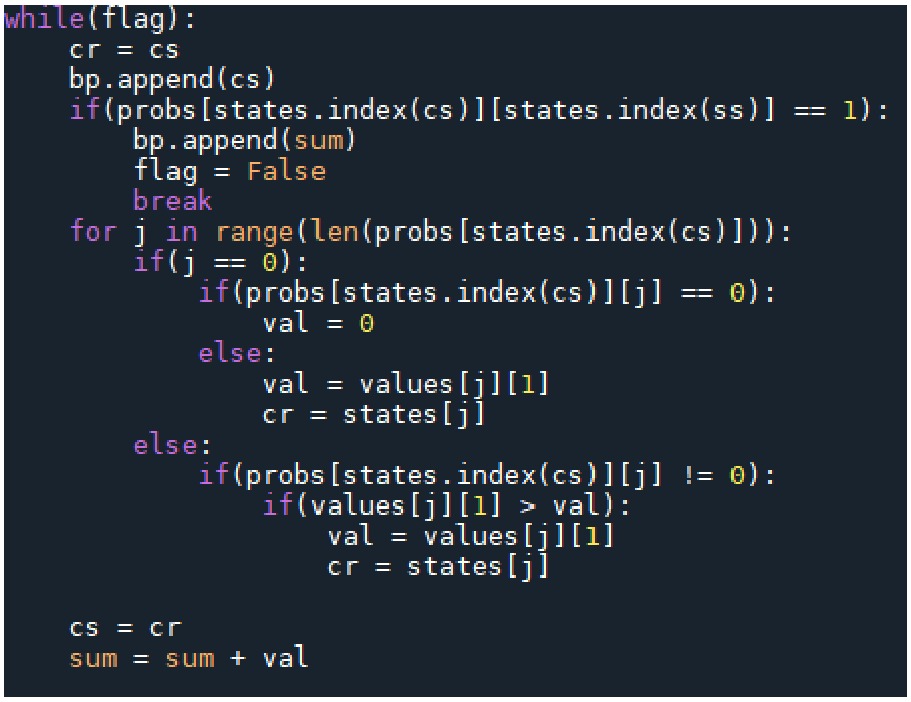
Figure 26 MDP Solution Value Function

In the ***value*** function inputs are:

- States of the MDP (states): States contains a 1x8 list with all unique state names.
- Current state (cs): Current state to calculate its value.
- First state (ss): The starting state for the MDP.
- Probability list (prob): Prob contains a 8x8 matrix with all probabilities from-to accordingly for each state.
- Reward list (reward): Reward contains a 8x8 matrix with all rewards from-to accordingly for each state.
- Discount factor γ (gamma): $0 \leq \gamma \leq 1$.

After the values for each state are set, the next step is to calculate the **Best Policy *(π)***, which is the optimal path that provides the highest sum of rewards based on

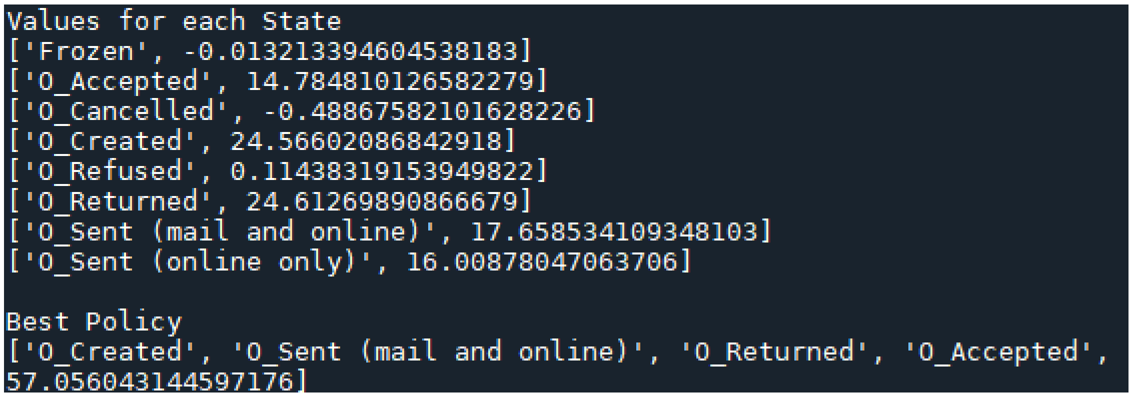
$$\pi(s) = argmax_a\{\sum_{s'} (\mathrm{P}(s'|\, s,\ a)(\mathrm{R}(s'|\, s,\ a) + \gamma V(s'))\}$$



```
while(flag):
    cr = cs
    bp.append(cs)
    if(probs[states.index(cs)][states.index(ss)] == 1):
        bp.append(sum)
        flag = False
        break
    for j in range(len(probs[states.index(cs)])):
        if(j == 0):
            if(probs[states.index(cs)][j] == 0):
                val = 0
            else:
                val = values[j][1]
                cr = states[j]
        else:
            if(probs[states.index(cs)][j] != 0):
                if(values[j][1] > val):
                    val = values[j][1]
                    cr = states[j]

    cs = cr
    sum = sum + val
```

Figure 27 MDP Solution Best Policy

In this coding block, a list for the best policy (bp) will contain the best available path and the sum of the reward. If the current state is the ending state (numbered as “1”) then it appends the *sum* in the list and breaks the loop. On any other occasion, for any current state that is not the ending state, it calculates which the next available state is with the highest reward based on the *current state*. Then adds the value to the *sum* and makes that state our new *current state*. The results are displayed below:



```
Values for each State
['Frozen', -0.013213394604538183]
['O_Accepted', 14.784810126582279]
['O_Cancelled', -0.48867582101628226]
['O_Created', 24.56602086842918]
['O_Refused', 0.11438319153949822]
['O_Returned', 24.61269890866679]
['O_Sent (mail and online)', 17.658534109348103]
['O_Sent (online only)', 16.00878047063706]

Best Policy
['O_Created', 'O_Sent (mail and online)', 'O_Returned', 'O_Accepted',
57.056043144597176]
```

Figure 28 MDP Solution Results

5.3.3 Visualization of Markov Models

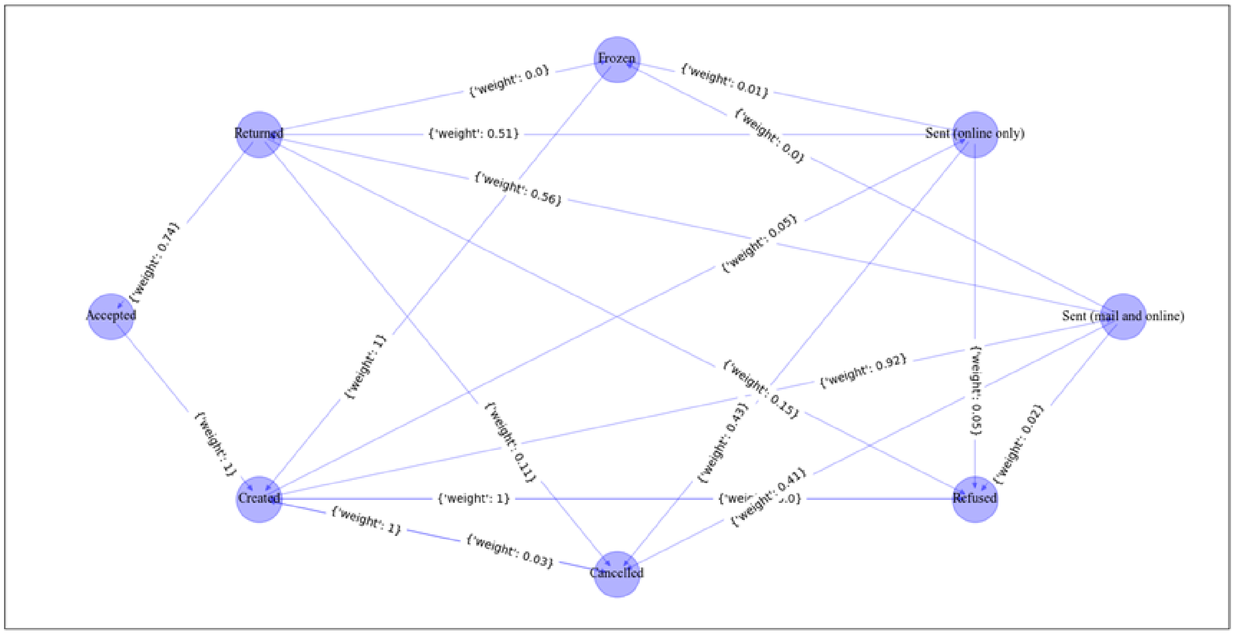


Figure 29 Markov Chain

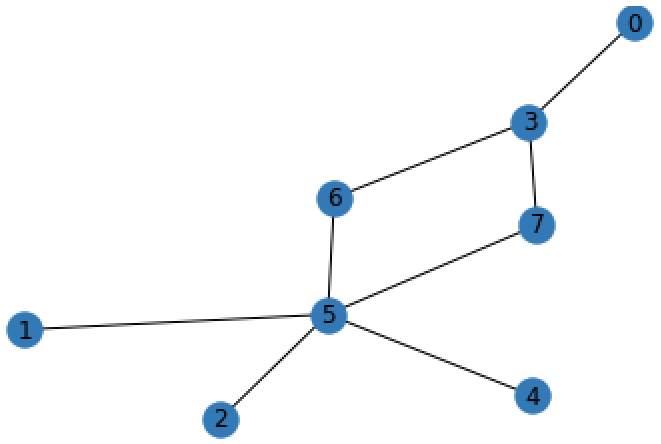


Figure 30 MDP

Each node is assigned to a state:

- **0**: Frozen
- **1**: O_Accepted
- **2**: O_Canceled
- **3**: O_Created
- **4**: O_Refused
- **5**: O_Returned
- **6**: O_Sent (mail and online)
- **7**: O_Sent (online only)

5.4 Reinforcement Learning Solution

5.4.1 Q-Learning

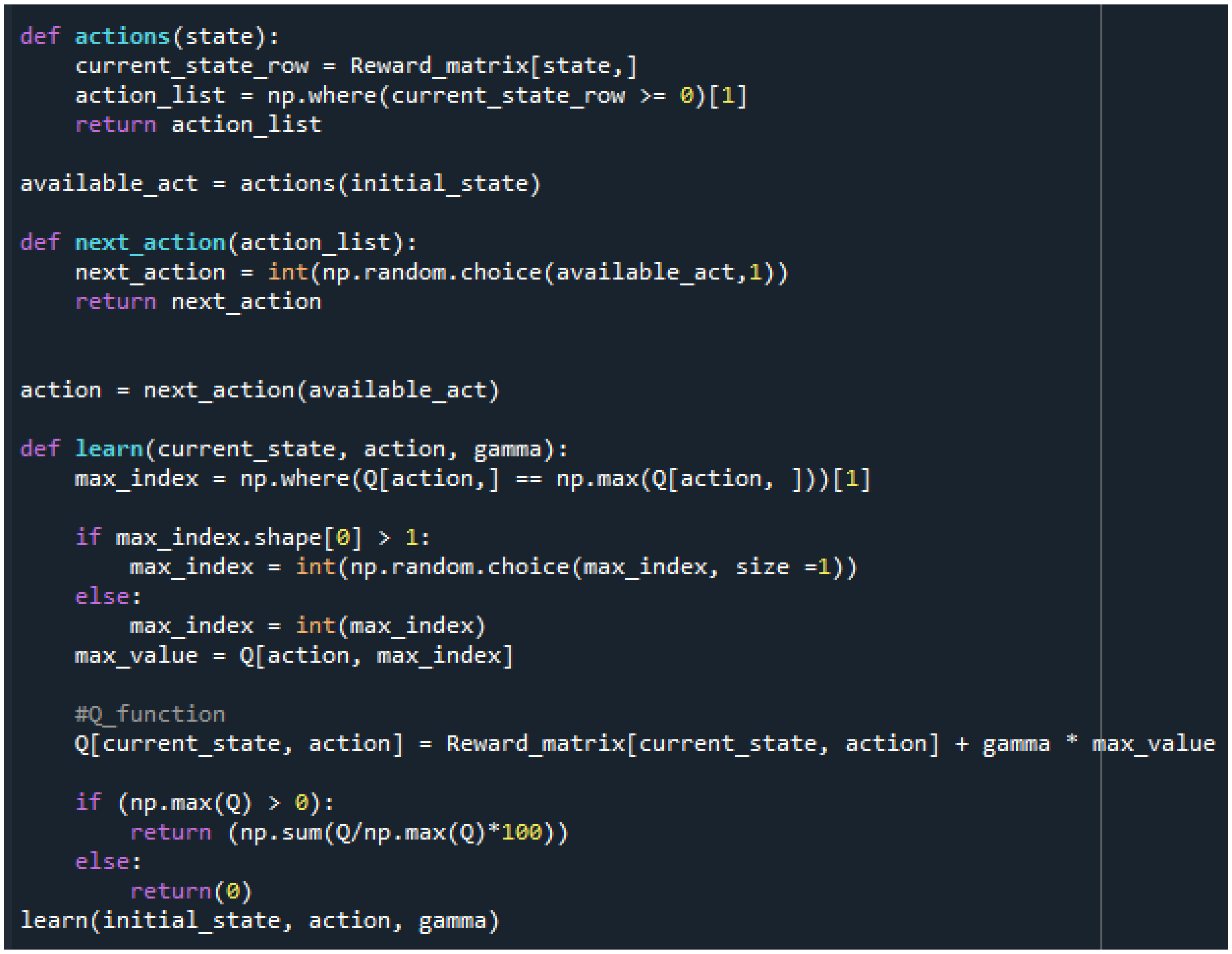
In the implementation of the Q-learning algorithm the numpy library was used. The first step is the construction of the reward matrix. The rows and columns of the matrix represent the activities of the process model. Each element corresponds to a state or a transition to a state.

	Frozen	O_Accepted	O_Canceled	O_Created	O_Refused	O_Returned	O_Sent (mail and online)	O_Sent (online only)
Frozen	-1	-1	-1	0	-1	-1	-1	-1
O_Accepted	-1	100	-1	0	-1	-1	-1	-1
O_Canceled	-1	-1	-1	0	-1	-1	-1	-1
O_Created	-1	-1	0	-1	0	-1	0	0
O_Refused	-1	-1	-1	0	-1	-1	-1	-1
O_Returned	0	100	0	-1	0	-1	-1	-1
O_Sent (mail and online)	0	-1	0	-1	0	0	-1	-1
O_Sent (online only)	0	-1	0	-1	0	0	-1	-1

Table 8 Q-learning Reward Matrix

The elements with value minus one (“-1”) express that there is no direct connection between the states. Elements with zero value (“0”) express direct connection between states thus, a valid action to move to that state. The value one hundred (“100”) corresponds to goal state and the transition to the goal state. The next step is the initialization of the Q-matrix, the size of the Q-matrix is the same as the reward-matrix, 8x8. There are three functions for the implementation of the Q-learning.

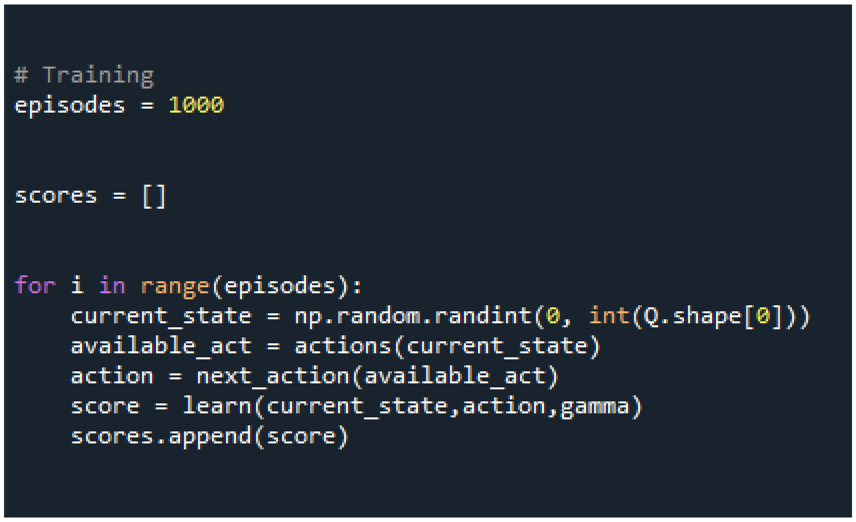
- **Available_actions:** Takes as an input a number that matches a state. The input is a value that corresponds to an activity. From that row we select all indexes whose elements are non-negative. This list is the available actions for the input state.
- **next_action:** Takes as an input the list of available actions and selects one of them at random.
- **learn:** It has three inputs. The current state, an action and the discount factor gamma. We implement the Q function and we use a greedy method to choose an action.



```python
def actions(state):
    current_state_row = Reward_matrix[state,]
    action_list = np.where(current_state_row >= 0)[1]
    return action_list

available_act = actions(initial_state)

def next_action(action_list):
    next_action = int(np.random.choice(available_act,1))
    return next_action


action = next_action(available_act)

def learn(current_state, action, gamma):
    max_index = np.where(Q[action,] == np.max(Q[action, ]))[1]

    if max_index.shape[0] > 1:
        max_index = int(np.random.choice(max_index, size =1))
    else:
        max_index = int(max_index)
    max_value = Q[action, max_index]

    #Q_function
    Q[current_state, action] = Reward_matrix[current_state, action] + gamma * max_value

    if (np.max(Q) > 0):
        return (np.sum(Q/np.max(Q)*100))
    else:
        return(0)
learn(initial_state, action, gamma)
```

Figure 31 Q-learning man functions

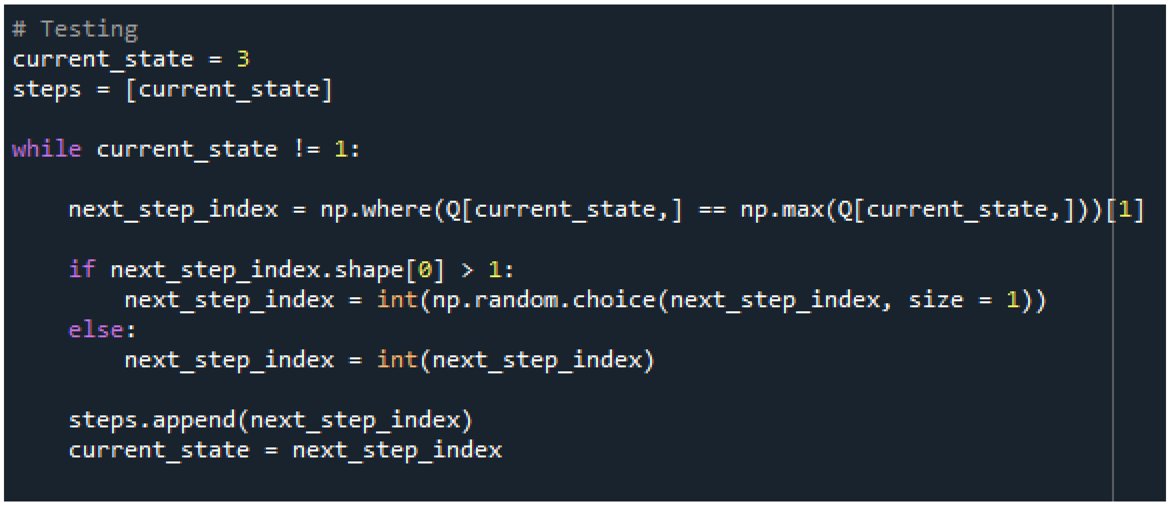
After the implementation of the three functions, we created a training and a testing loop. In the training we apply the three functions in sequence and we keep the scores to a list.



```python
# Training
episodes = 1000


scores = []


for i in range(episodes):
    current_state = np.random.randint(0, int(Q.shape[0]))
    available_act = actions(current_state)
    action = next_action(available_act)
    score = learn(current_state,action,gamma)
    scores.append(score)
```

Figure 32 Q-learning training loop

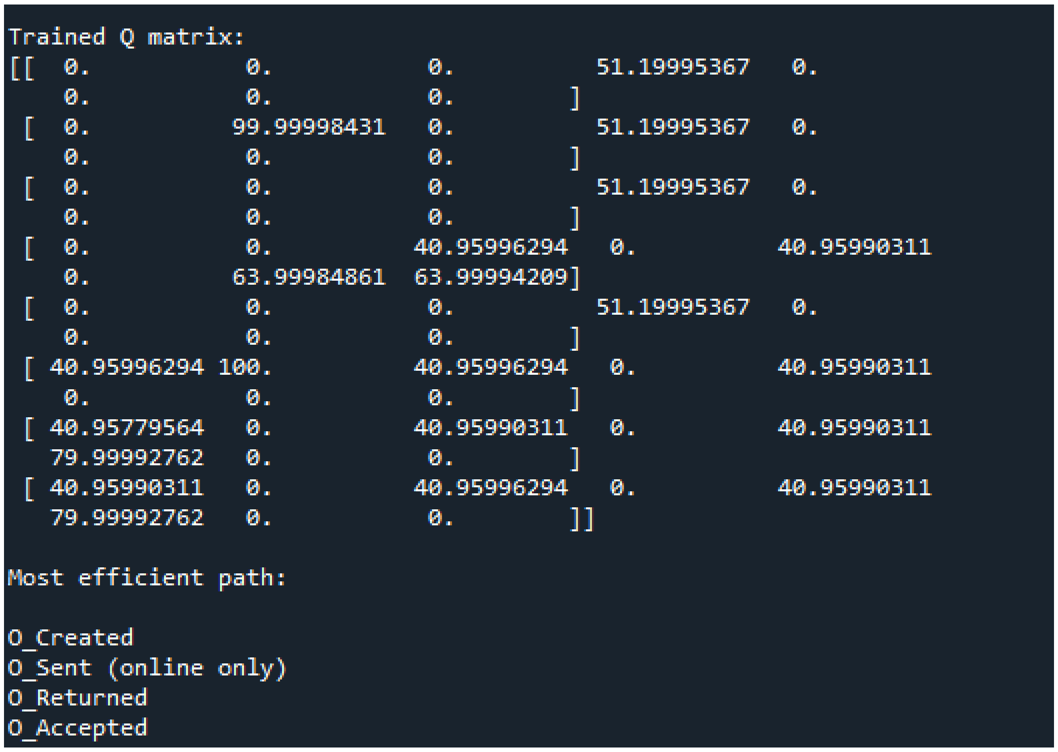
In the training loop we start from the initial state. Then, we choose the max value from the Q table of the given state. If there is no score, we choose a next state at random.



```
# Testing
current_state = 3
steps = [current_state]

while current_state != 1:

    next_step_index = np.where(Q[current_state,] == np.max(Q[current_state,]))[1]

    if next_step_index.shape[0] > 1:
        next_step_index = int(np.random.choice(next_step_index, size = 1))
    else:
        next_step_index = int(next_step_index)

    steps.append(next_step_index)
    current_state = next_step_index
```

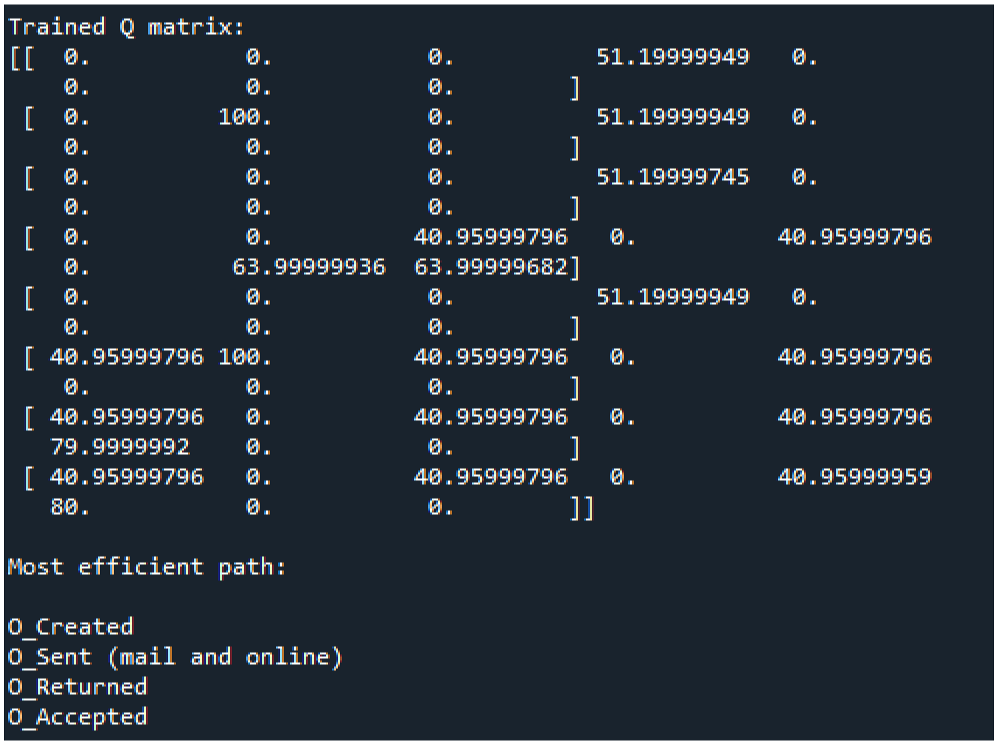
Figure 33 Q-learning testing loop

The most efficient path can vary between the “O_Sent (mail and online)” and “O_Sent (online only)” states. Because the two states have the same reward picking one is a matter of chance and it may differ depending on training.



```
Trained Q matrix:
[[  0.           0.           0.          51.19995367   0.
    0.           0.           0.        ]
 [  0.          99.99998431   0.          51.19995367   0.
    0.           0.           0.        ]
 [  0.           0.           0.          51.19995367   0.
    0.           0.           0.        ]
 [  0.           0.          40.95996294   0.          40.95990311
    0.          63.99984861  63.99994209]
 [  0.           0.           0.          51.19995367   0.
    0.           0.           0.        ]
 [ 40.95996294 100.          40.95996294   0.          40.95990311
    0.           0.           0.        ]
 [ 40.95779564   0.          40.95990311   0.          40.95990311
   79.99992762   0.           0.        ]
 [ 40.95990311   0.          40.95996294   0.          40.95990311
   79.99992762   0.           0.        ]]

Most efficient path:

O_Created
O_Sent (online only)
O_Returned
O_Accepted
```

Figure 34 Q-learning results 1



```
Trained Q matrix:
[[  0.           0.           0.          51.19999949   0.
    0.           0.           0.        ]
 [  0.         100.           0.          51.19999949   0.
    0.           0.           0.        ]
 [  0.           0.           0.          51.19999745   0.
    0.           0.           0.        ]
 [  0.           0.          40.95999796   0.          40.95999796
    0.          63.99999936  63.99999682]
 [  0.           0.           0.          51.19999949   0.
    0.           0.           0.        ]
 [ 40.95999796 100.          40.95999796   0.          40.95999796
    0.           0.           0.        ]
 [ 40.95999796   0.          40.95999796   0.          40.95999796
   79.9999992    0.           0.        ]
 [ 40.95999796   0.          40.95999796   0.          40.95999959
   80.           0.           0.        ]]

Most efficient path:

O_Created
O_Sent (mail and online)
O_Returned
O_Accepted
```

Figure 35 Q-learning results 2

In the training graph we can see that we need about 500 episodes of training to achieve maximum score and stable results.

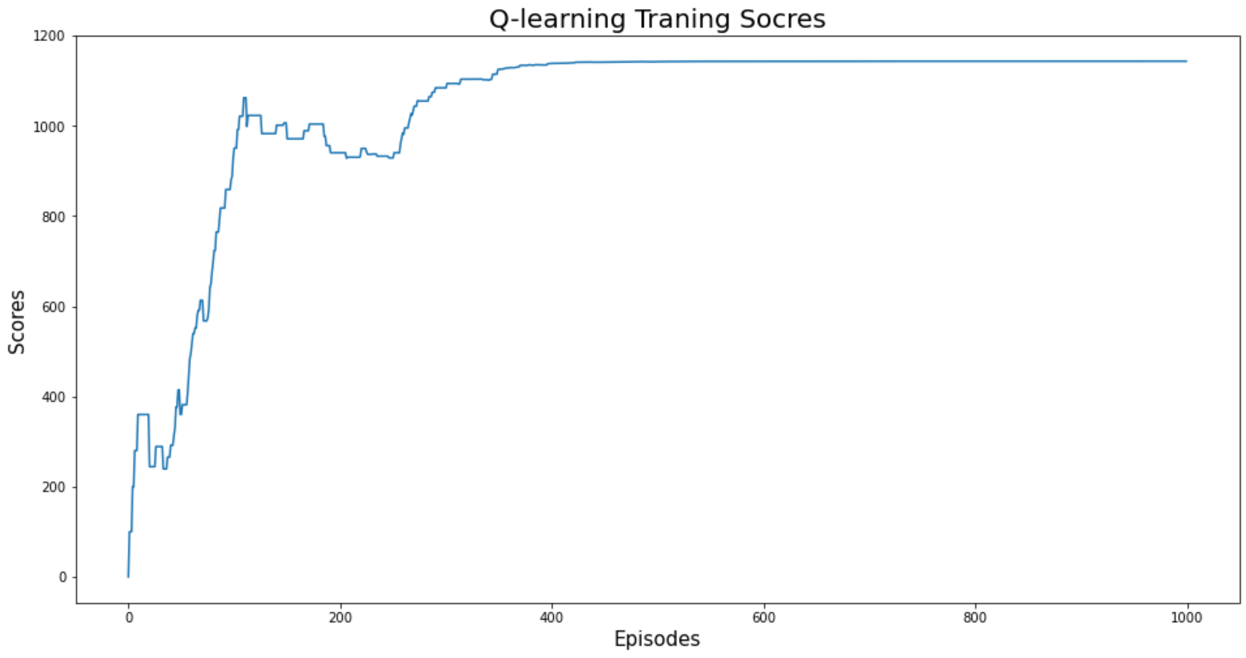


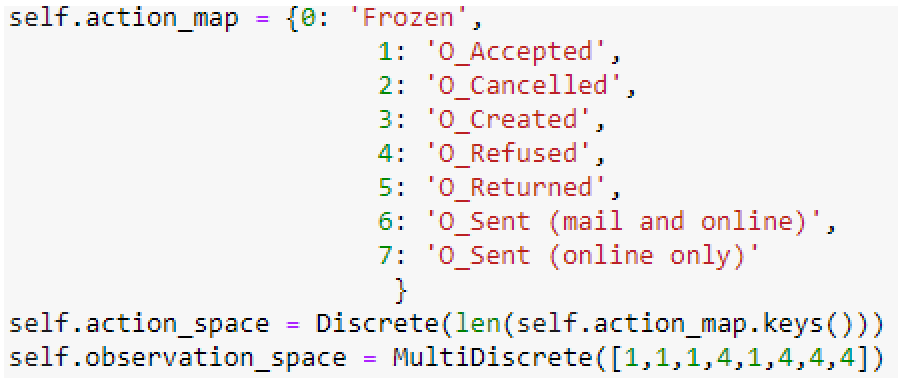
Figure 36 Q-learning training scores

5.4.2 DQN implementation

In the DQN implementation the Gym library was used. In order to use the Gym framework first we need to create an environment that describes our problem. The environment is represented in the form of a class. The class is obligated to contain some attributes and functions

- Attributes
    - **action space:** A discrete space of length eight, same as the number of states
    - **observation space:** A multi-discrete space, a series of discrete observation spaces. Each series in the observation space represent the number of connections each state has.
- Functions
    - **step:** Performs an action and returns next state reward done flag.
    - **reset:** Sets the environment to the initial state.

Following the previously mentioned rules, our environment set up looks like:



```
self.action_map = {0: 'Frozen',
                   1: 'O_Accepted',
                   2: 'O_Cancelled',
                   3: 'O_Created',
                   4: 'O_Refused',
                   5: 'O_Returned',
                   6: 'O_Sent (mail and online)',
                   7: 'O_Sent (online only)'
                    }
self.action_space = Discrete(len(self.action_map.keys()))
self.observation_space = MultiDiscrete([1,1,1,4,1,4,4,4])
```

Figure 37 Reinforcement learning environment

- **Action_map**: Action Map based on all available states.
- **Action_space**: It is built as ***Discrete(8)*** since the action will be a number in range of [0-7] indicating which will be the next state after the transition.
- **Observation_space**: It is built as MultiDiscrete([1,1,1,4,1,4,4,4]) indicating how many states are available or "observable" to transition from each state.

After the environment space is set up setting up a reward matrix or rather having set on the step function the rewards based on the designed goal is the next step. We decided to have a reward matrix and have the set up function get each corresponding reward based on the transition.

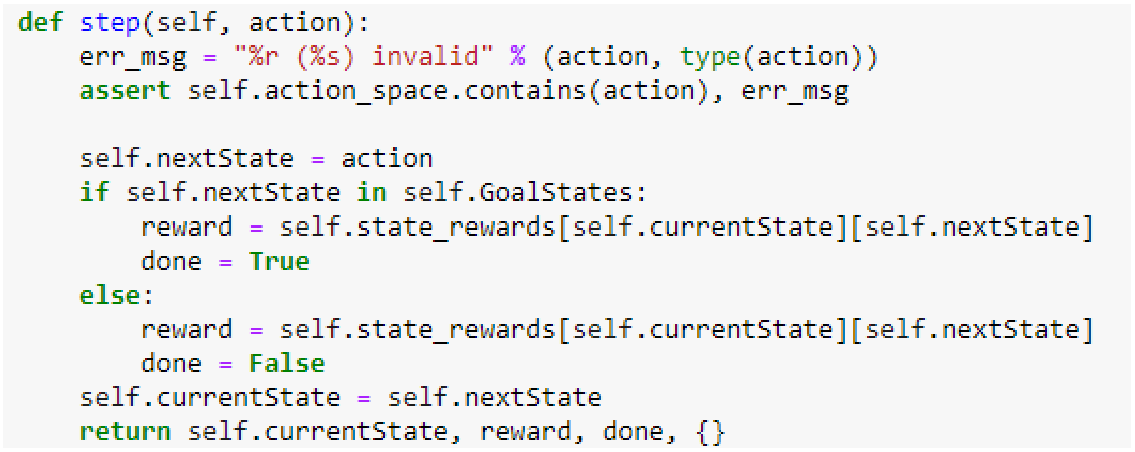
	Frozen	O_Accepted	O_Canceled	O_Created	O_Refused	O_Returned	O_Sent (mail and online)	O_Sent (online only)
Frozen	-1	-1	-1	0.1	-1	-100	-1	-1
O_Accepted	-1	-1	-1	0.1	-1	-100	-1	-1
O_Canceled	-1	-1	-1	0.1	-1	-100	-1	-1
O_Created	-1	-1	-1	0.1	-1	-100	1	1
O_Refused	-1	-1	-1	0.1	-1	-100	-1	-1
O_Returned	-1	100	-1	-1	-1	-100	-1	-1
O_Sent (mail and online)	-1	-1	-1	-1	-1	1	-1	-1
O_Sent (online only)	-1	-1	-1	-1	-1	1	-1	-1

Table 9 DQN Reward Matrix

In conclusion, we guide the agent through our reward matrix set up to learn how to transition in the environment and decide which the optimal path is across the available paths.

After all the important attributes are set, the next step is adjusting the needed functions to make use of the attributes.

Our step function:

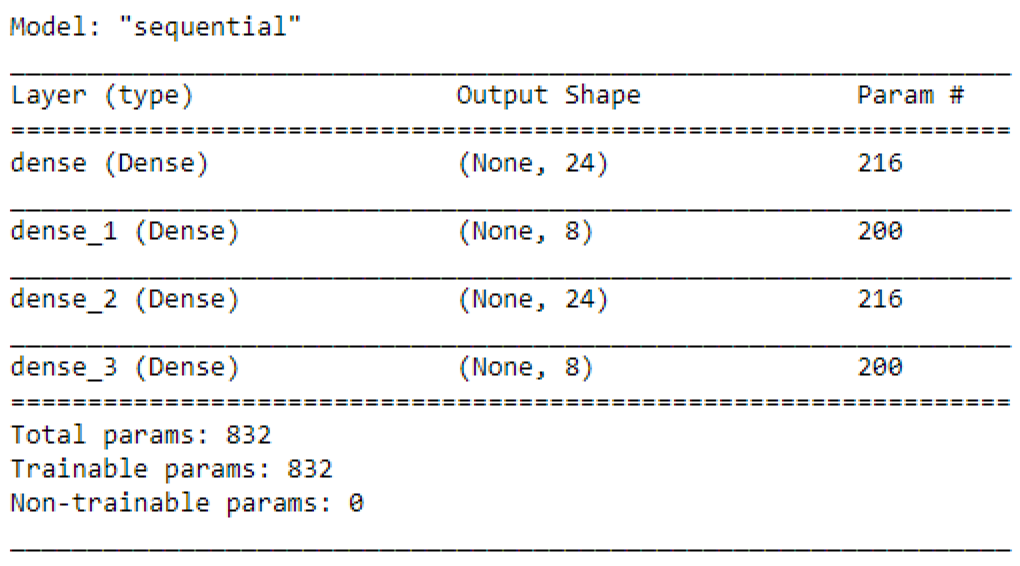


```python
def step(self, action):
    err_msg = "%r (%s) invalid" % (action, type(action))
    assert self.action_space.contains(action), err_msg

    self.nextState = action
    if self.nextState in self.GoalStates:
        reward = self.state_rewards[self.currentState][self.nextState]
        done = True
    else:
        reward = self.state_rewards[self.currentState][self.nextState]
        done = False
    self.currentState = self.nextState
    return self.currentState, reward, done, {}
```

Figure 38 Reinforcement learning step function

First, a validation that the action given from the agent is within the set of available actions. After the action validation, we check if the state that the agent will transition is either an **ending state**, also known as a ***Goal State***. If it is an ending state, we set the reward for this transition and notify the agent that another cycle in the environment has been complete, if it is not an ending state, we set the corresponding reward for this transition and notify the agent that he has not reached yet an ending state.

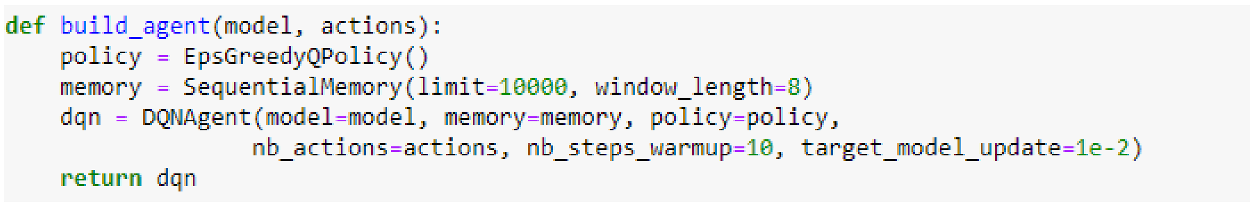
Using the same bases as in Q-learning we create a reward matrix to calculate and choose the rewards for each action. The ANN (Artificial Neural Network) used is described in the image below:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 24)                216
_________________________________________________________________
dense_1 (Dense)              (None, 8)                 200
_________________________________________________________________
dense_2 (Dense)              (None, 24)                216
_________________________________________________________________
dense_3 (Dense)              (None, 8)                 200
=================================================================
Total params: 832
Trainable params: 832
Non-trainable params: 0
_________________________________________________________________
```

Figure 39 ANN model description

Our agent is described in the image below:



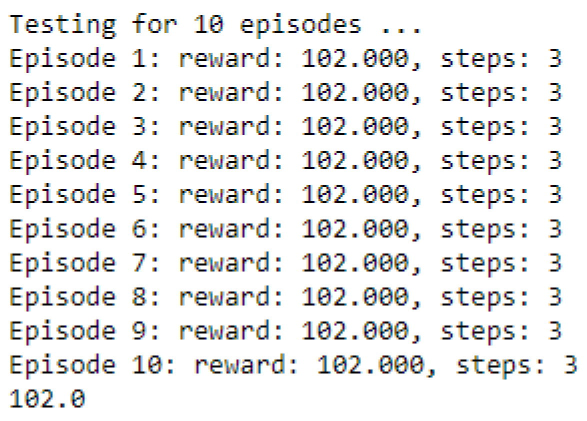
```
def build_agent(model, actions):
    policy = EpsGreedyQPolicy()
    memory = SequentialMemory(limit=10000, window_length=8)
    dqn = DQNAgent(model=model, memory=memory, policy=policy,
                   nb_actions=actions, nb_steps_warmup=10, target_model_update=1e-2)
    return dqn
```

Figure 40 DQN agent function

The agent uses the ANN described above, the ε-greedy policy and sequential memory.

After 50000 number of training steps we have the following results.

First our testing episodes:



```
Testing for 10 episodes ...
Episode 1: reward: 102.000, steps: 3
Episode 2: reward: 102.000, steps: 3
Episode 3: reward: 102.000, steps: 3
Episode 4: reward: 102.000, steps: 3
Episode 5: reward: 102.000, steps: 3
Episode 6: reward: 102.000, steps: 3
Episode 7: reward: 102.000, steps: 3
Episode 8: reward: 102.000, steps: 3
Episode 9: reward: 102.000, steps: 3
Episode 10: reward: 102.000, steps: 3
102.0
```

Figure 41 Episode testing results

As we can see from the results, the agent has found the most efficient path, which is 3 steps and accumulates the highest possible reward: 102.

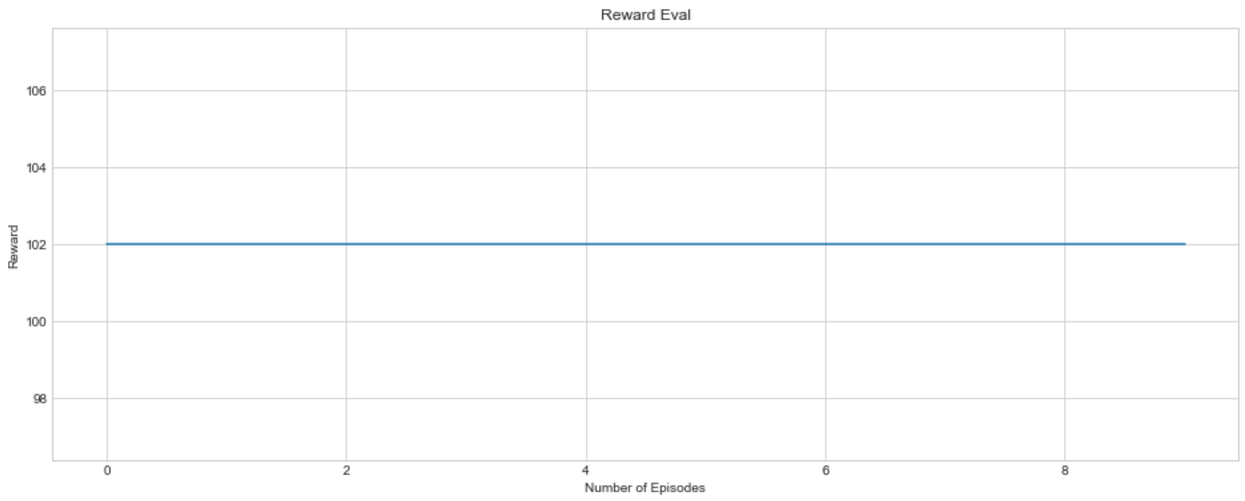


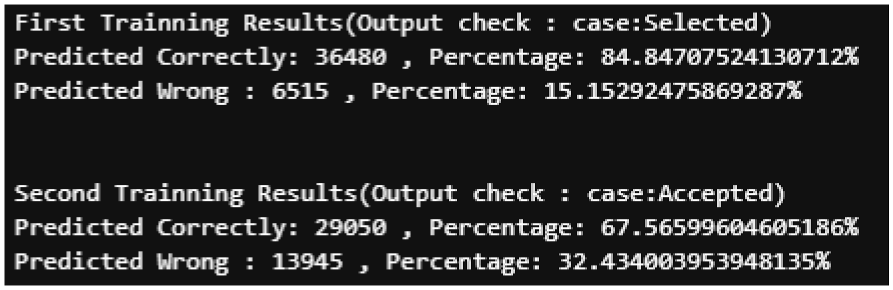
Figure 42 Reinforcement Learning training results

5.5 Experimental Implementations

We also implemented a neuron which does binary sorting and uses a cleaned version of the data set. As an input it has 7 elements, 5 as the main data for each case and 2 as the classification results.

- **Main elements:**
    - case: MonthlyCost
    - case: FirstWithdrawalAmount
    - case: CreditScore
    - case: OfferedAmount
    - case: NumberOfTerms
- **Classification Results:**
    - Selected
    - Accepted

First, we train the ANN, having both main elements and classification results. After the training is complete, we use the ANN to classify each case based on the 5 elements and then compare its own predictions with the actual classification. Below we see the accuracy of the classification prediction:



```
First Trainning Results(Output check : case:Selected)
Predicted Correctly: 36480 , Percentage: 84.84707524130712%
Predicted Wrong : 6515 , Percentage: 15.15292475869287%


Second Trainning Results(Output check : case:Accepted)
Predicted Correctly: 29050 , Percentage: 67.56599604605186%
Predicted Wrong : 13945 , Percentage: 32.434003953948135%
```

Figure 43 Deep learning Classification

In conclusion, if a problem can be modified or remodeled as a classification problem, an implementation like this is suggested.

5.6 Composition and Evaluation of Results

One thing that it is important during conclusion, is how the results are handled and how you decide to implement, modify and create the solution for each problem. In our case, the first main result that we had to choose was the business processes mined from the mining algorithms. We decided to go with the **Inductive miner** since it was the simplest and cleanest one among the rest of the results. Also while we were on the latter stages of development, we noticed that the results of the **Inductive miner** were fitting our solution approach perfectly.

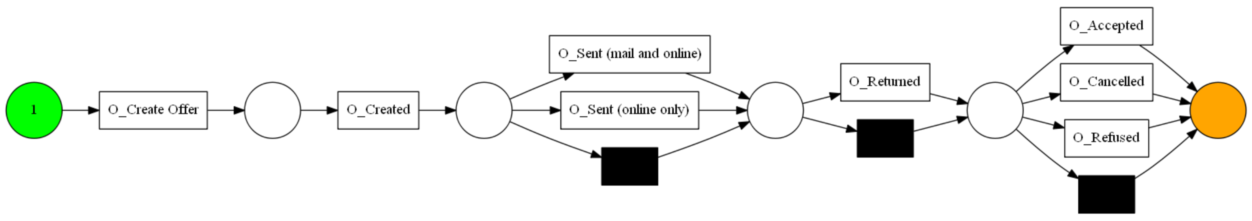
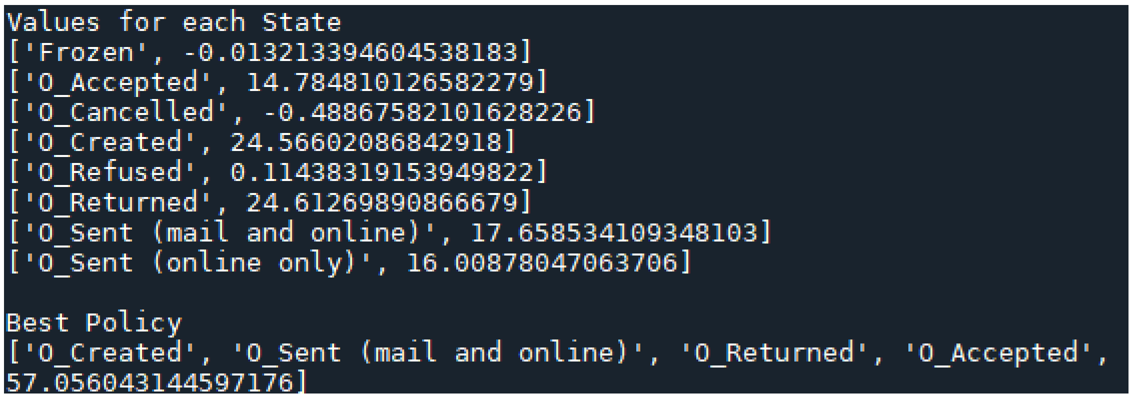


Figure 44 Inductive miner petri net

This *black box* that was unidentified from the Inductive miner, was our process modification that added another state, the **Frozen** state in order to have completed instances and a better distribution on the probabilities for our Markov Model.

The second main result we had to choose was, how the development of the **Markov Model** would occur. It was important to create a fitting **Markov Chain** since that is the base form of our **Markov Decision Process**. The Markov Chain is based on the Inductive miner results, with the addition of probabilities. Our **Markov Decision Process** is based on the Markov chain, but also takes into account the stochastic behavior of the real world. The main goal for our MDP to find the values for each state along with the most efficient path from the *starting state* to the *ending state*, having also the property of uncertainty.

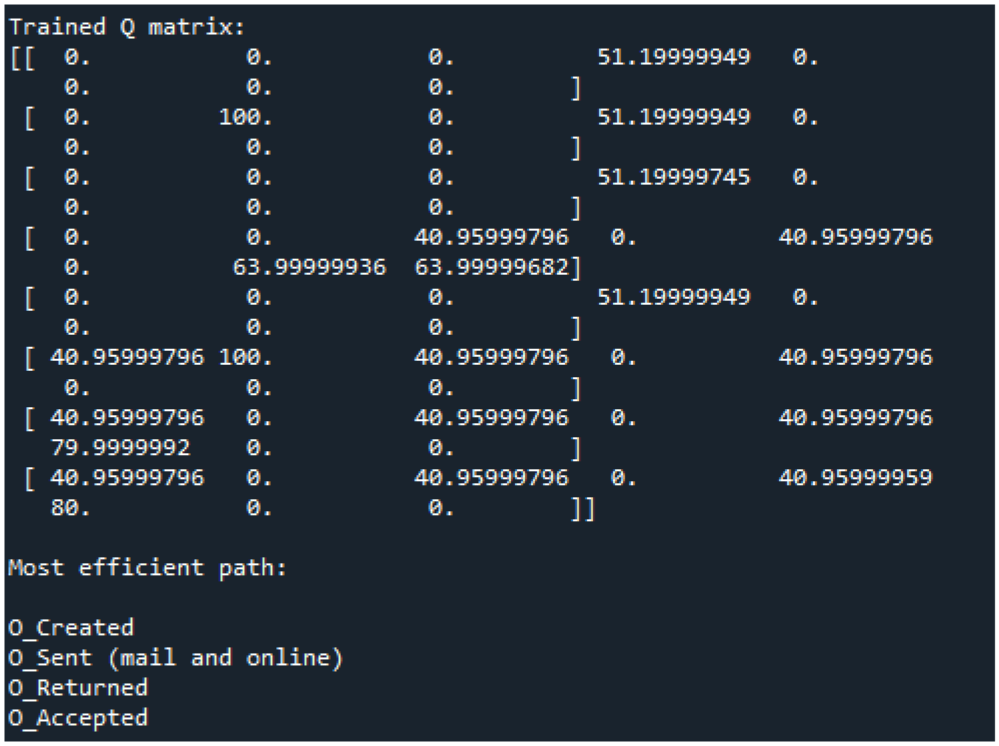


```
Values for each State
['Frozen', -0.013213394604538183]
['O_Accepted', 14.784810126582279]
['O_Cancelled', -0.48867582101628226]
['O_Created', 24.56602086842918]
['O_Refused', 0.11438319153949822]
['O_Returned', 24.61269890866679]
['O_Sent (mail and online)', 17.658534109348103]
['O_Sent (online only)', 16.00878047063706]

Best Policy
['O_Created', 'O_Sent (mail and online)', 'O_Returned', 'O_Accepted',
57.056043144597176]
```

Figure 45 MDP results

Finally, the results from the previous sections of our implementation helped us have a better starting point of view of how reinforcement learning works and how to set it up. Comparing the results and the implementation difficulties, Q-learn is better and simpler in this case. DQN is used in more complex problems because its' agent is trained as a neural network and it is challenging to create an environment for a simple problem, unlike the Q-learning algorithm, which it can be used in simpler problems and it is easier to set up the attributes needed with problems like ours. If the DQN model is created properly through all its complexity to build up, it can be more rewarding as a solution. Another difference between Q-learn and DQN is that Q-learn finds the most efficient path between two states while in the DQN the agent trained to "walk" the most efficient path in the environment. To distinguish the difference in our problem between Q-learn and DQN solutions, the DQN solution needs more work for the same results as the Q-learn.

Q-learn Results:



```
Trained Q matrix:
[[  0.           0.           0.          51.19999949   0.
    0.           0.           0.        ]
 [  0.         100.           0.          51.19999949   0.
    0.           0.           0.        ]
 [  0.           0.           0.          51.19999745   0.
    0.           0.           0.        ]
 [  0.           0.          40.95999796   0.          40.95999796
    0.          63.99999936  63.99999682]
 [  0.           0.           0.          51.19999949   0.
    0.           0.           0.        ]
 [ 40.95999796 100.          40.95999796   0.          40.95999796
    0.           0.           0.        ]
 [ 40.95999796   0.          40.95999796   0.          40.95999796
   79.9999992    0.           0.        ]
 [ 40.95999796   0.          40.95999796   0.          40.95999959
   80.           0.           0.        ]]

Most efficient path:

O_Created
O_Sent (mail and online)
O_Returned
O_Accepted
```

Figure 46 Q-learning Results

6 Conclusions and Future Work

6.1 Conclusions

The two rapidly advancing fields of Process Mining and Machine Learning capitalize on the increasing amount of data generated. These two branches of computer science are common tools in Business Intelligence. Process mining is focused on what is really happening and when. This can give us a clear picture of the problem we are trying to solve. Machine learning is mostly used for predicting future behavior of a system with respect to the known data. In other words, Process Mining can provide us good data for the model. Process Mining and Machine learning is yet to be bridged. The solution of multi factor problems and complex systems are always a deemed challenging task. In our opinion the two disciplines do not need to overlap but complement each other.

6.2 Future work

Many approaches have been left out of the implementation due to lack of time. Experimenting with more complex ideas can be very time consuming considering the experimentation and the problems that arise along the way. Implementation of an environment for reinforcement learning is a demanding task.

1. It could be interesting to take into account more features besides state transitions.
2. The existing model and environment can always be improved.
3. Considering a way to increase the complexity a dynamic environment can be created.

References

[1] Paul Soare, 2012. "Opportunities For Driving Continuous Improvement Through Tqm, Lean And Six Sigma Within Business Process Management," Proceedings of the INTERNATIONAL MANAGEMENT CONFERENCE, Faculty of Management, Academy of Economic Studies, Bucharest, Romania, vol. 6(1), pages 193-202, November.

[2] Article : Why Business Process Management?

https://www.igrafx.com/articles/why-business-process-management/

[3]Lee, R. G., & Dale, B. G. (1998). Business process management: a review and evaluation. Business Process Management Journal

[4] Article : Business Process Management: What Is BPM and Why You Need It

https://www.outsystems.com/blog/posts/business-process-management/#tools

[5] Hammer, M. (2014). What is Business Process Management? Handbook on Business Process Management 1, 3–16.

[6] Process simulation support in BPM tools: The case of BPMN by Freitas, António Paulo Pereira, José Luís Mota, Proceedings of 2100 Projects Association Joint Conferences – Vol.X (20XX)

[7] Aalst, W. van der (2016). Process Mining: Data Science in Action

[8] W.M.P. van der Aalst. Process Mining: Discovery, Conformance and Enhancement of Business Processes.

[9] Leemans, S. J. J., Fahland, D., & van der Aalst, W. M. P. (2013). Discovering Block-Structured ProcessModels from Event Logs - A Constructive Approach. Lecture Notes in Computer Science, 311–329.

[10] W.M.P. van der Aalst. Process Mining: Discovery, Conformance and Enhancementof Business Processes. (p 128)

[11] Van der Aalst, W. M. P. (2019). A practitioner’s guide to process mining: Limitations of the directly-follows graph. Procedia Computer Science, 164, 321–328.

[12] Carl Adam Petri and Wolfgang Reisig (2008) Petri net. Scholarpedia, 3(4):6477.

[13] Van der Aalst, W. M. P. (2013). Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases, 31(4), 471–507.

[14] OMG (2011), 'Business Process Model and Notation (BPMN), Version 2.0' , Object Management Group , Technical report, Object Management Group

[15] Kalenkova, A. A., van der Aalst, W. M. P., Lomazova, I. A., & Rubin, V. A. (2015). Process mining using BPMN: relating event logs and process models. Software & Systems Modeling, 16(4), 1019–1048.

[16] Rozinat, A., & van der Aalst, W. M. P. (2008). Conformance checking of processes based on monitoring real behavior. Information Systems, 33(1), 64–95. doi:10.1016/j.is.2007.07.001

[17] Ailenei, I., Rozinat, A., Eckert, A., & van der Aalst, W. M. P. (2012). Definition and Validation of Process Mining Use Cases. Lecture Notes in Business Information Processing, 75–86. doi:10.1007/978-3-642-28108-2_7

[18] Dunzer, S., Stierle, M., Matzner, M., & Baier, S. (2019). Conformance checking. Proceedings of the 11th International Conference on Subject-Oriented Business Process Management - S-BPM ONE '19. doi:10.1145/3329007.3329014

[19] IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams. (n.d.). doi:10.1109/ieeestd.2016.7740858

[20] Van der Aalst, W. M. P., & van Dongen, B. F. (2013). Discovering Petri Nets from Event Logs. Lecture Notes in Computer Science, 372–422. doi:10.1007/978-3-642-38143-0_10

[21] Weijters, A. J. M. M., Aalst, van der, W. M. P., & Alves De Medeiros, A. K. (2006). Process mining with the HeuristicsMiner algorithm. (BETA publicatie : working papers; Vol. 166). Technische Universiteit Eindhoven

[22] Gagniuc, Paul A. (2017). Markov Chains: From Theory to Implementation and Experimentation.

[23] Eddy, S. What is a hidden Markov model? Nat Biotechnol 22 (2004).

[24] Semi-Markov Models and Applications by Jacques Janssen, Nikolaos Limnios

[25] Kaelbling, L. P.; Littman, M. L.; Cassandra, A. R. (1998). "Planning and acting in partially observable stochastic domains".

[26] Kindermann, Ross; Snell, J. Laurie (1980). Markov Random Fields and Their Applications

[27] Bui, H. H.; Venkatesh, S.; West, G. (2002). "Policy recognition in the abstract hidden markov model".

[28] Pratas, D.; Hosseini, M.; Pinho, A. J. (2017). "Substitutional tolerant Markov models for relative compression of DNA sequences". PACBB 2017 – 11th International Conference on Practical Applications of Computational Biology & Bioinformatics, Porto, Portugal.

[29] Carpinone, A; Giorgio, M; Langella, R.; Testa, A. (2015). "Markov chain modeling for very-short-term wind power forecasting".

[30] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep Reinforcement Learning That Matters. Proceedings of the AAAI Conference on Artificial Intelligence, 32(1). Retrieved from https://ojs.aaai.org/index.php/AAAI/article/view/11694

[31] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., & Pineau, J. (2018). An Introduction to Deep Reinforcement Learning. Foundations and Trends® in Machine Learning, 11(3-4), 219–354. doi:10.1561/2200000071 arXiv:1811.12560v2

[32] Kaelbling, Leslie P.; Littman, Michael L.; Moore, Andrew W. (1996). "Reinforcement Learning: A Survey". Journal of Artificial Intelligence Research. 4: 237–285. arXiv:cs/9605103.

[33] Russell, Stuart J. (Stuart Jonathan). Artificial Intelligence : a Modern Approach. Upper Saddle River, N.J. :Prentice Hall, 2010.

[34] El Naqa I., Murphy M.J. (2015) What Is Machine Learning?. In: El Naqa I., Li R., Murphy M. (eds) Machine Learning in Radiation Oncology. Springer, Cham. https://doi.org/10.1007/978-3-319-18305-3_1

[35] Burnetas, A. N., & Katehakis, M. N. (1997). Optimal Adaptive Policies for Markov Decision Processes. Mathematics of Operations Research, 22(1), 222–255. doi:10.1287/moor.22.1.222

[36] Reinforcement Learning: An Introduction: R.S. Sutton, A.G. Barto, MIT Press, Cambridge, MA 1998, 322 pp. ISBN 0-262-19398-1.

[37] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016). https://doi.org/10.1038/nature16961

[38] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602

[39] Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S. & Hassabis, D. (2015), 'Human-level control through deep reinforcement learning', Nature 518 (7540), 529--533.

[40] Lillicrap, T., Jonathan J. Hunt, A. Pritzel, N. Heess, T. Erez, Yuval Tassa, D. Silver and Daan Wierstra. “Continuous control with deep reinforcement learning.” CoRR abs/1509.02971 (2016): n. pag.