Aalto University
School of Electrical Engineering
Degree Programme in Automation and Electrical Engineering

Jaakko Kilpeläinen

Automating knowledge work of service desk:

Machine learning model for software robot

Master's Thesis
Espoo, May 27, 2019

Supervisor: Professor Valeriy Vyatkin
Advisor: Karri Lehmusvaara M.Sc. (Tech.)

Aalto University
School of Electrical Engineering
School of Electrical Engineering — ABSTRACT OF
Degree Programme in Automation and Electrical Engineering — MASTER'S THESIS

Author:	Jaakko Kilpeläinen		
Title: Automating knowledge work of service desk: Machine learning model for software robot			
Date:	May 27, 2019	**Pages:**	vii + 62
Major:	Control, Robotics and Autonomous Systems	**Code:**	ELEC3025
Supervisor:	Professor Valeriy Vyatkin		
Advisor:	Karri Lehmusvaara M.Sc. (Tech.)		

Aging population, legacy systems and pressure on cost savings are all growing problems for modern-day companies. One relief for these problems is to automate business processes and IT-service desk tasks with software robots. The aim of the automation is to reduce employees' growing workload so that they have time to work with the more valuable tasks. One of the most limitating factors of using software robots are that the automated processes must be strictly rule-based and the input data must be highly structured. In business there has been a lot of talk about using machine learning and other AI-techniques for achieving more generic solutions, but the actual results have not yet been publicly recognised.

In this thesis it is intended to examine the suitability of machine learning for software robotics by automating company's internal IT-services. The goal is to build a working solution and find a machine learning platform for the company that provides software robotic solutions as a service.

The end result is a viable automation solution which uses machine learning model for probability-based decision making. Based on this research it is possible say that there exist synergy benefits between the two technologies, as long as there is a suitable application for them.

Keywords: RPA, IPA, Software Robotics, Machine Learning
Language: English

Aalto-yliopisto
Sähkötekniikan korkeakoulu
korkeakoulu

Aalto-yliopisto
Sähkötekniikan korkeakoulu
Automaation ja sähkötekniikan koulutusohjelma

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä:	Jaakko Kilpeläinen		
Työn nimi:	IT-palvelun tietotöiden automatisointi: Koneoppimismalli ohjelmistorobotille		
Päiväys:	27. toukokuu 2019	**Sivumäärä:**	vii + 62
Pääaine:	Säätötekniikka, Robotiikka ja Autonomiset Järjestelmät	**Koodi:**	ELEC3025
Valvoja:	Professor Valeriy Vyatkin		
Ohjaaja:	Karri Lehmusvaara M.Sc. (Tech.)		

Ikääntyvä väestö, vanhat järjestelmät ja paine kustannussäästöille ovat nykypäivän yritysten kasvavia ongelmia. Yksi helpotus näihin ongelmiin on liiketoimintaprosessien ja IT-palveluiden automatisointi ohjelmistorobottien avulla. Automatisoinnin pyrkimyksenä on vähentää työntekijöiden kasvavaa työtaakkaa, jotta heillä jää enemmän aikaa liiketoiminnalle tärkeämpien tehtävien hoitoon. Yksi suurimmista ohjelmistorobotiikan käyttöä rajoittavista tekijöistä on se, että automatisoitavien prosessien tulee olla tarkasti sääntöihin perustuvia, sekä hyödynnettävän tiedon tarkkaan jäsenneltyä. Alalla on ollut paljon puhetta koneoppimisen ja tekoälyn hyödyntämisestä geneerisempien ratkaisujen saavuttamiseen, mutta tuloksia näiden projektien onnistumisesta ei ole kantautunut suuren yleisön tietoisuuteen.

Tässä työssä on tarkoitus tutkia koneoppimisen soveltuvuutta ohjelmistorobotiikkaan automatisoimalla yrityksen sisäisiä IT-palveluja. Päämääränä on rakentaa toimiva prototyyppi ja löytää koneoppimismalleja hyödyntävä alusta, jota prosessien automatisointia tarjoava yritys voisi alkaa käyttämään osana ratkaisukokonaisuuttaan.

Lopputuloksena saatiin toimiva ratkaisu, joka höydyntää koneoppimista todennäköisyyksiin perustuvassa päätöksenteossa. Tutkimuksen avulla voidaan sanoa, että kahden tekniikan välillä on synergiahyötyjä, kunhan niille löydetään sopiva käyttökohde.

Asiasanat:	RPA, IPA, Ohjelmistorobotiikka, Koneoppiminen
Kieli:	Englanti

Acknowledgements

This thesis was done concurrently with my daily work, which occasionally caused problems with time management. I wish to thank my wife, professor Vyatkin and all my co-workers involved in the thesis for encouraging me to complete the work. Without your support, I would probably still be writing.

Espoo, May 27, 2019

Jaakko Kilpeläinen

Abbreviations and Acronyms

2FA	Two-Factor Authentication
AI	Artificial Intelligence
AML	Azure Machine Learning
API	Application Programming Interface
BPO	Business Process Outsourcing
CSV	Comma Separated Value
ERP	Enterprise Resource Planning
FTE	Full-Time Equivalent
IDF	Inverse Document Frequency
IEC	International Electrotechnical Commission
IPA	Intelligent Process Automation
ISO	International Organization for Standardization
ITIL	Information Technology Infrastructure Library
ITSM	IT Service Management
JSON	JavaScript Object Notation
REST	Representational State Transfer
RPA	Robotic Process Automation
SOA	Service-Oriented Architecture
SQL	Structured Query Language
TF	Term Frequency
WS	Web Service
URI	Uniform Resource Identifier
UML	Unified Modeling Language
XML	Extensible Markup Language

Contents

Chapter 1

Introduction

Since the industrial revolution work has been more efficient when looking at production volumes. In practice this means the same working process that was earlier done by one human is now divided in to sub-processes and done by multiple humans [1]. When it was noticed that work based on predefined processes increases volumes and cuts costs, a lot of work has been tried to structure into predefined processes.

Major part of today's work done in developed countries is done with computers. This connected with the fact above means that there is a huge amount of processes made with computers. Process or more precisely a business process is a broad concept, which can mean almost any organizational activity, that contains sequential tasks related to each other [2].

A typical back-office business process usually contains the following steps. First, logging into the system and in to the necessary programs, taking a case for handling from the software's work queue and checking the case details. After assessing the necessary actions for the case, booking resolutions steps, making necessary information/configuration changes and sending a confirmation email. Process like this requires the use of more than one software. This slows down the work and makes it more difficult when employees need to log in into multiple systems and transfer data between them. The number of intermediate steps also increases the amount of human errors, which then reduces the quality of service. Above described simple business critical processes are typically only reformed during major organizational changes. They can also be automated, which leaves more time for employees to focus on more challenging things. Companies have awaken to automate these repetitive routine tasks and McKinsey Global Institute's research predicts that by 2030, 75 to 375 million workers will need to switch occupational categories due to increasingly capable machines [3].

If a company wants to automate these above-mentioned processes, they have three options from which to choose; buy or re-factor a new software that suits better for the company's purposes, integrate legacy systems under one master system, or automate the usage of current systems with scripts or software robots. Every technique has its own problems. Buying or refactoring a new software costs and the data in use needs to be connected to it. In system integration the biggest problem is that older software is not usually designed to communicate with other vendors' products and they suffer from overall lack of openness [4]. Also the hidden costs of architectural design, data structuring and nontechnical difficulties can be significant and affect the total cost of the project negatively [5]. When the work performance is increased with software robots that mimic how a human would do the same sequential tasks, the old systems will remain the same as earlier. This means that sooner or later old systems need to be changed or updated. Software robots are also not suitable for most time-critical environments due to latencies when using systems via user interfaces. The technology is also sensitive to changes in the user interface [6]. When something changes in the target system, the software robot typically needs be reconfigured to reflect the changed system. Despite the weaknesses, software robot industry is growing and it is predicted to grow from $250 million in 2016 to $2.9 billion in 2021 [7].

By using software robots for automating high volume back office business processes, large companies have reported convincing results. Telecommunication provider Telefonica O2 has reached hundreds of full-time equivalent (FTE) savings only by automating 15 of their core processes with around 160 software robots (one robot equals to one software license in this case.) [8]. The project's estimate for three-year return on investment (ROI) is also strong: between 650 and 800 percentages. For achieving these results software robots need to handle 400000 - 500000 transactions per month. Similar results have also been obtained from other studies. Global company Xchanging have received 420 percentage three year ROI by automating 14 of their core processes, while the monthly transaction amount was 120000 [9]. Neither of the two researches are based on actual measurements but the companies have given out results from their own projects.

The company that this master's thesis is done for sells software robots, or in other words robotic process automation (RPA) as a cloud solution. The basic idea is that the software robot server is connected to customer's network via VPN so that the software can be used from outside network in the same way as in on-premise solutions. When the software robot starts its work, it first logs in into customers network through the normal Window's login screen, launches software and starts doing the process same way as a human would do.

The unit that this project is made for is called run management (RM). It takes care that the software robots are in a working condition 24/7. Most of the RM's work is routine maintenance tasks that need human decision making based on error messages, i.e. tickets. Oxford dictionaries defines a ticket in the field of information technology as follows: *a request logged on a work tracking system detailing an issue that needs to be addressed or task that must be performed* [10]. The same definition is used in this master's thesis to mean specifically a message received by run management's IT service management (ITSM) system, which tells that something is wrong with an automated process. The most common error is caused by network errors and latencies in the customer's systems and the software robot doesn't know how to recover from these. Incident investigation is time-consuming manual work, and in many cases a simple software robot's reboot solves the disturbance situation temporarily. The amount of this manual work is growing all the time due to new customers and the company is looking for new ways to automate the work. Automating manual work is still so challenging that it cannot be automated by software robots with normal methods. One solution for this could be to use machine learning, which is a subpart of more commonly known concept called artificial intelligence. Software vendors and service providers working in the RPA sector are already marketing solutions that combine artificial intelligence and RPA. When talking about the combination they speak about Intelligent Process Automation (IPA). Despite the marketing speeches and articles, the actual scientific results have not yet been publicly recognized.

1.1 Problem statement

In this master's thesis it is intended to investigate the synergy benefits between machine learning and RPA by building a prototype that uses both techniques and automates run management's maintenance tasks. Effective end results would have direct impact to the RM unit's efficiency and with information obtained from the thesis, the company could continue the development and start to offer machine learning solutions as a part of their service portfolio.

Work is divided into three individual parts. In the first part it is intended to choose and build the best machine learning model for predicting if software robot's restart could solve the received ticket. The data used for teaching the model is tickets from company's IT service management system and emails. In the second phase the aim is to build a web service, where the software robot could send new tickets and obtain values based on model's prediction. In the last step it is intended to build a software robot that reads the ticket, sends it to web service and reacts based on the received value. A principle of the idea is seen in the figure 1.1 below. In the picture the arrows reflect the direction of communication and colours reflect how the work is divided.

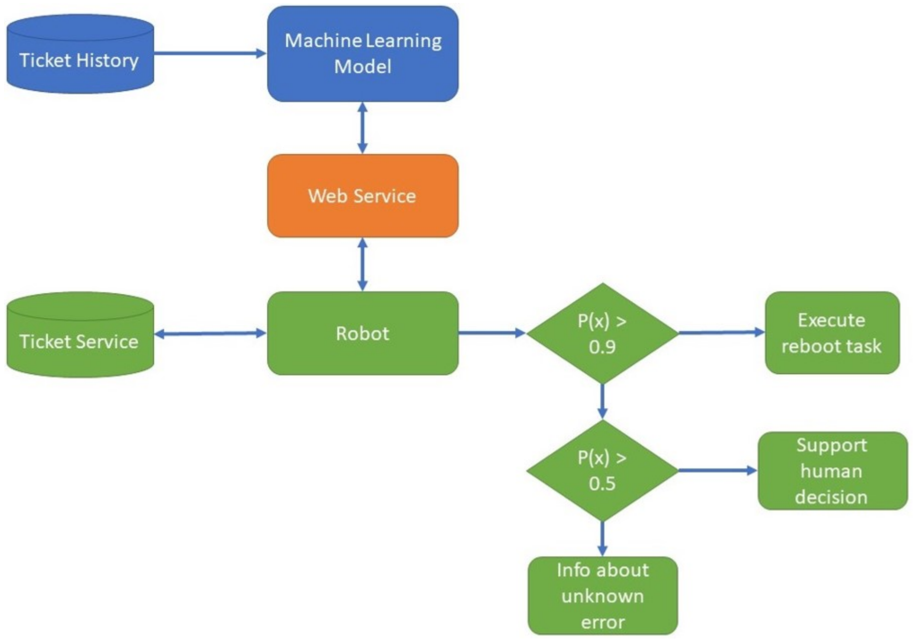


Figure 1.1: Principle idea of the prototype.

1.2 Structure of the thesis

The study is divided under three parts that logically follow the progress of the project. First there is a review of fields literature and researches to gather up enough information to find and design the best possible architecture and techniques for later implemented solution. Simultaneously the aim is to open the dealt technical areas for the reader, so that s/he can later critically review the created solution and the obtained results.

In the next section the developed automation solution is opened to the reader. The created entity is divided into three logical parts based on the solution architecture; machine learning, web service and robotic process automation process. During these chapters the reader will be provided with enough information and results to open up the conclusions made during the last paragraph.

The last chapter brings together the findings of the research and discusses the visions of the future. This section highlights the problems encountered in the master's thesis and provides internal and external further research topic suggestions from a business perspective.

Chapter 2

Background investigation

The purpose of this chapter is to study subjects needed for successful automation project. At the beginning of each section the handled subject is clarified for the reader, so that she or he can critically review the obtained results.

2.1 Business process automation

Researchers Michael Hammer and James Champ define business process as follows; "a collection of activities that takes one or more kinds of input and creates an output that is of value to the customer." [11]. Depending on the size of the company business processes can be managed by one person or they can pass through every single department inside a massive organization. An example of this is the employee on-boarding process. When a new employee starts, information about this needs to flow at least to payroll, IT and to the department that handless employee access permits. The arrival of information invokes sub-processes inside each department, for example the IT department creates user IDs for systems requested by the employee's supervisor and sends a response when these have been created. Company's current situation might be that a human in the IT service-desk needs to handle this kind of routine tasks manually. In addition, a large part of the working time is spent on processing users' forgotten passwords.

Typically, the business processes' efficiency is investigated in a more detailed level only when a problem is noticed. With the on-boarding process this means that a new employee needs to wait multiple days before accounts are created for the needed systems. This is because service desk is struggling with higher priority tasks and none of them have the time to perform routine tasks due to resource outage. This reduces the efficiency of other departments when they need to wait for the IT to solve their problems. Currently the company's top management realizes that the service desk is a bottleneck that slows down the company's growth. They decide to reduce the workload with automation, rather than hiring new employees.

As described in the introduction, companies have three options to choose from when automating business processes; buy new software or extensions for the existing systems, integrate systems or automate processes with software robots. Every technique has its own pros and cons and the company needs to choose the most suitable one. Most important for a company's successful automation project is that suitable processes are chosen and that they are improved before the actual implementation of the new technology starts [12]. Humans are still more flexible when performing a work, than a machine. Instead of seeking fully automatable business processes and trying to force the technology into where it is not yet suitable, an automated support for the resource executing the business process needs to be considered[13].

2.1.1 Model of the automated process

Business process modeling and business process mining are their own science fields that are not deeply introduced in this master's thesis. The most important thing to understand is that there is no universal model that fits for all purposes. On the upper level, business processes can be modeled in example with flowcharts, RACI matrices or business process modeling notation (BPMN) [14]. If more detailed model needs to be build, Petri nets and State Space models can be used [15]. However, the main goal of modeling is that the model describes what the modeler wants it to bring up. Universal Modeling Language (UML) is used in this master's thesis, which is suitable for upper and lower level diagrams, depending on how detailed level the modeler wants to describe the workflow.

The company has already earlier done a research in which it found a suitable process to be automated for the pilot project. The choice was based on the fact that they already have historical data that could be analyzed and used for modeling purposes.

In figure 2.1 we see the principle of how an employee solves the software robot's crash situation. Workflow starts from black dot when the employee logs into ITSM system and reads a ticket. After that, he or she decides whether there is need for further investigation and if there is, gets credential information and connects to customer network. After employee has read the run logs and decided that simple restart does not help in this case, he or she launches the change process. These stages are only described on the top level and the actual process includes more steps and decisions.

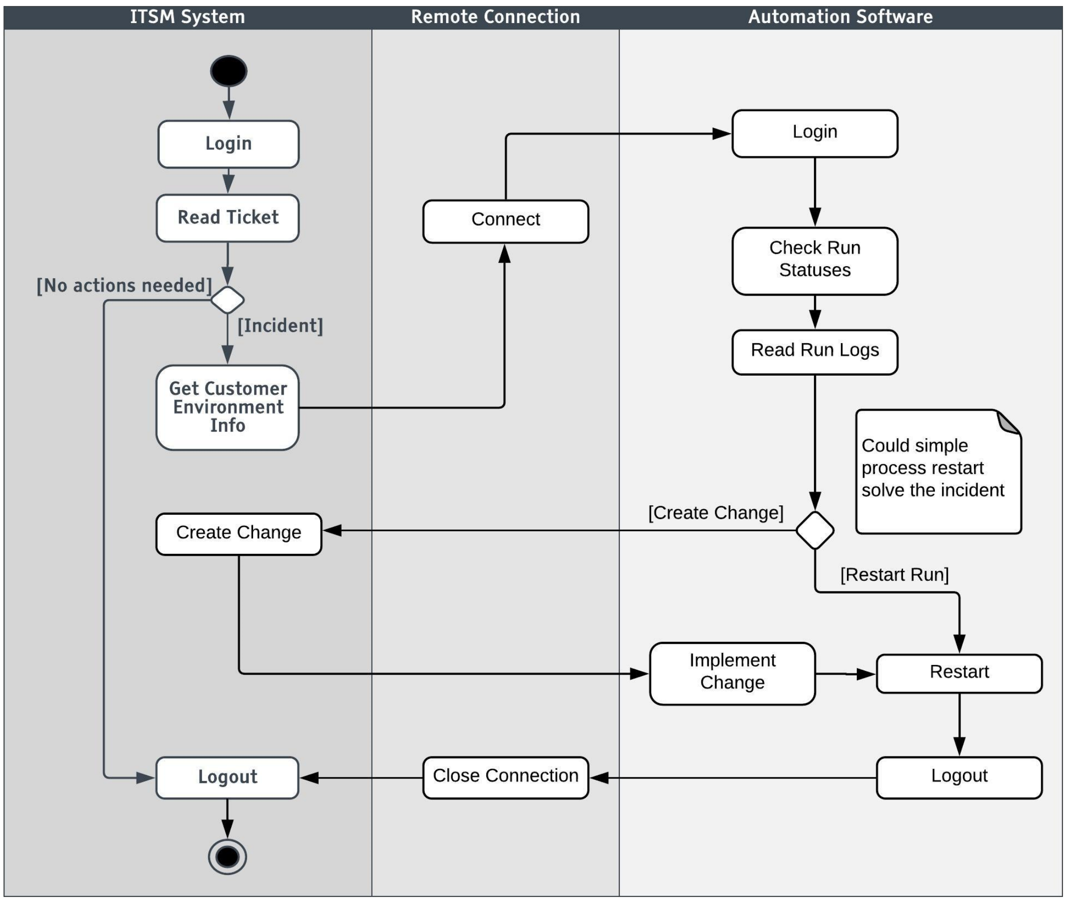


Figure 2.1: How a human checks and solves the software robot's crash situation.

2.1.2 Robotic process automation

IEEE's guide to intelligent process automation terms defines RPA as follows: "*A preconfigured software instance that uses business rules and predefined activity choreography to complete the autonomous execution of a combination of processes, activities, transactions, and tasks in one or more unrelated software systems to deliver a result or service with human exception management*" [16]. This definition describes the concept well but it needs to be opened a bit more.

Robotic process automation (RPA) is a technique to automate mature, high-volume and routine work tasks which typically are parts of back-office business processes. The idea is to mimic how humans work and then create a program to do these. Typically, the software robot is not programmed to handle every possible outcome of input cases, but those ”exceptions” are transferred to humans. Created solution should be easily configurable because typically software robots are not seen as a permanent solution and underlying software will change in the long run.

RPA software use the cursor for selecting items and the same operating system’s functions that people use with mouse and keyboard. From the RPA’s perspective a solution where a mouse is moved and clicked only based on screen coordinates and where the keyboard presses are sent without any further verifications is not a desired solution. There must be full assurance that the right items are selected, and the exact text strings are passed to the correct places. Otherwise the automation might lead to an undesired behavior that might have disastrous consequences for example in a company’s ERP system.

The reliability of the automated process can be improved so that before any actions are made the software robot checks that the wanted element (e.g. textbox) is available. Two different methods can be used to identify different elements from a computer’s screen. Software robot can search pre-defined pictures from the screen, or it can use the target system’s internal structures. Examples of internal structures are html-page’s elements and techniques that were originally developed for assisting blind people. These are for example Java’s Access Bridge (JAB) and Microsoft’s Active Accessibility (AA). The picture recognition is a much more laborious method and not as reliable as the structure-based ones. Finding the right spot on the screen requires taking predefined images and verification must be based on the image recognition algorithm. Also, after a target system’s visual update, the software robot’s pre-defined pictures need to be changed. After being sure that the right element is visible and the action is done, it is still necessary to ensure that the software robot has performed the correct operation and not for example entered the information in wrong field.

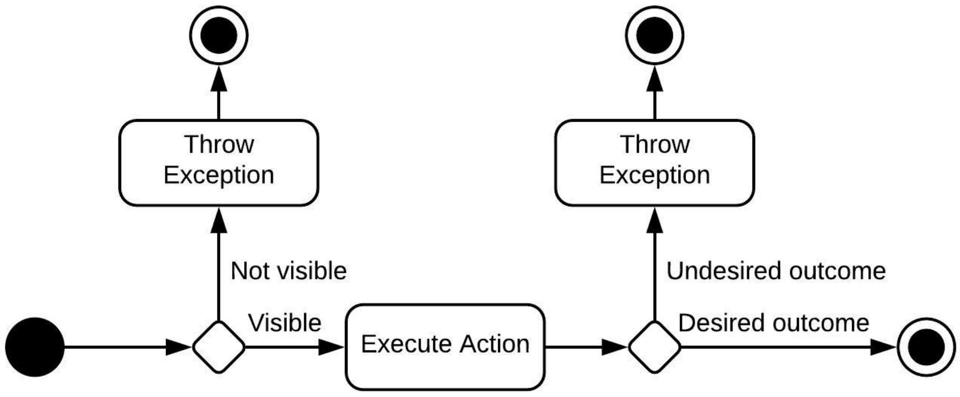


Figure 2.2: Principle of safe action execution.

Figure 2.2 shows the event chain of this idea. In the picture one software robot's navigation action in customer's system is demonstrated. The same logic can be utilized for reading and writing actions. Workflow goes from left to right starting from the black dot. Before the action is executed (reading, writing or navigation in customer's system), it must be guaranteed that the planned action can be safely executed. Otherwise the workflow needs to be stopped in a pre-planned way. In coding level this happens in a similar way than in many well-known textual programming languages. This happens by throwing an exception that is caught in another part of the code and here the error condition is handled in a controlled way. After the action has been executed successfully, there is another similar check for making sure that the action happened and was not interrupted for example by system latency or malfunction.

According to two the researchers who have written most about software robotics, Willcocks and Lacity [17], two things separate RPA from other tools in Business Process Management(BPM). In this thesis, when talking about the BPM related automation techniques, the term "system integration" is used. Firstly, because RPA is easier to configure and the developers don't need actual programming skills. In a typical RPA software, the actual developing is handled visually by dragging, dropping and linking blocks that present work phases. This reminds graphically IEC 61131-3's functional block diagram (FBD) and sequential function charts (SFC) languages [18]. Secondly RPA is non-invasive compared to system integration solutions. This means that they sit on top of existing systems and no modifications are needed to the underlying systems or IT security policies. Typically, the term software robot references to a combination of an automatic process running and the resource machine it runs on. On the IT side, one ”robot” equals one software license and thus one robot can run sequentially multiple processes [19].

In figure 2.3 is shown one automation software vendor’s basic automation workflow. The process starts from the top left corner and between the conditions ”Got Item?” and ”Stop?” exists a loop phase and process end to the top right corner when one of the loop condition is met.

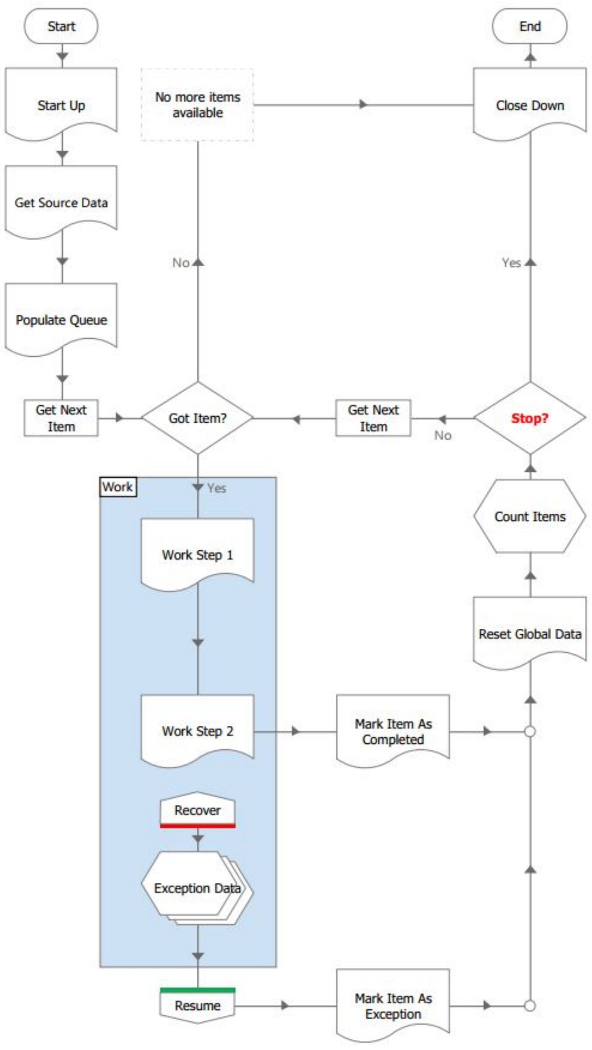


Figure 2.3: Software robots main program flow.

Creating a software robot starts with documenting how a person performs the sequential tasks one after another. The business process description should go to such a deep level that it is told what item is selected from software's current page's dropdown-menu, what is passed to textbox and what buttons are pressed. As stated in the IEEE's RPA definition, a part of the automation processes exception handling is left for humans. This is usually done for cost reasons. The set of all possible process input cases are normally distributed. Automation of major part of cases takes the same amount of labor than automation the remaining. Idea of this can be seen in figure 2.4 in a form of a s-curve.

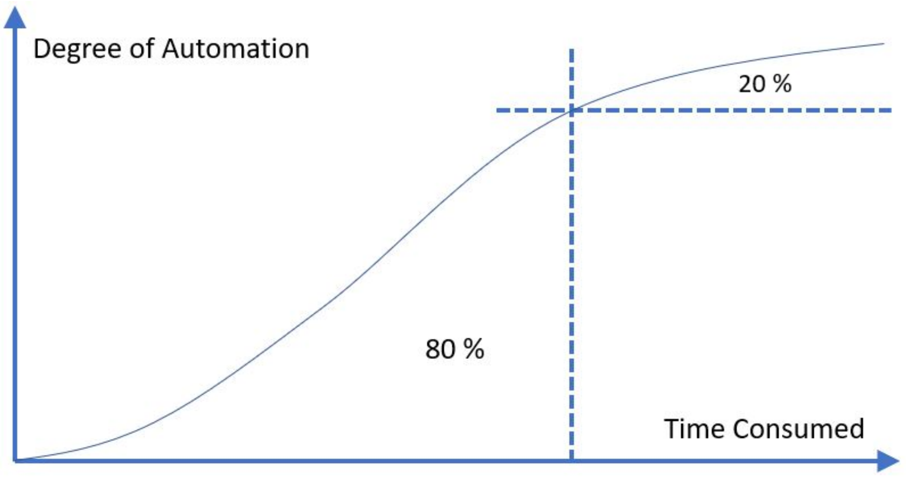


Figure 2.4: Process input case distribution divided to easily automated and exceptional cases.

When the actual programming starts, it is typical that the developer first creates the "happy path". Happy path means the workflow is successfully completed through software, which should happen for the majority of the process' input cases. The remaining part of cases are marked as exceptions and returned for employees. After that starts the implementation for less common input cases and the improvement of the software robot's resilience. From the programming perspective, this is similar to adding and optimizing "throw-catch"-structures that are used in the most widely used text-based programming languages like Java and C-Sharp. As in programmable logic controller (PLC) programming, the increase of resilience, testing and optimization should take the most part of the programming time. One thing that separates robotic process automation from automation macros is that RPA software offer control room for maintenance and reporting purposes. From there it is easy to follow runs, survey logs and schedule new runs, which improves maintainability and traceability of software robots.

2.1.3 RPA IT Infrastructure

In this section is introduced the basic IT infrastructure for larger RPA solutions. The company that this master's thesis is made for provides similar infrastructure as an on-premise or as a cloud solution. A typical network architecture consists of more than two servers. All RPA platform software can also be installed as a standalone installation for proof of concept (PoC) purposes, but this is not a scalable solution for large environments. In a large environment, in order to facilitate the maintenance, it is necessary that all the automated processes can be managed from one centralized place.

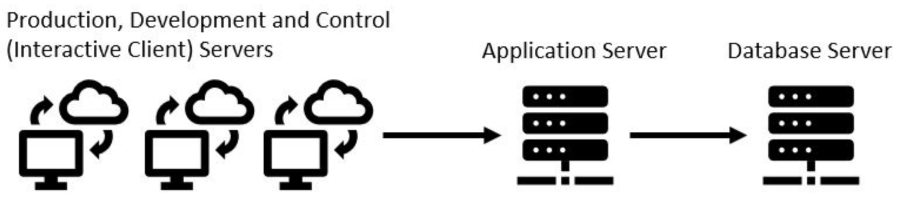

Production, Development and Control
(Interactive Client) Servers
Application Server
Database Server

Figure 2.5: Default RPA IT infrastructure.

Figure 2.5 introduces the basic design for larger scale RPA infrastructure. In the picture, the arrows indicate direction of access restrictions. All computers in the figure can be imagined as virtual servers where are connected from physical machines for development or configuration purposes. When viewing the image from right to left, the database server become first. The point of this part is to store the RPA processes' codes, user credentials, audit information and automation processes' run logs. The application server is an optional part which idea is to remove the need for various connections between the other components and database. Typically, there is also installed Microsoft Windows service or services which are responsible for managing data encryption, allocation and execution of automated processes and schedule orchestration. Interactive clients are used for automation environment controlling and development machines for processes' coding. The actual automated robotic processes run on production resources. The company's run management unit therefore mainly uses IC computers for monitoring and configuring the processes running in production servers. IT department handles the database and application servers and developers work on development machines.

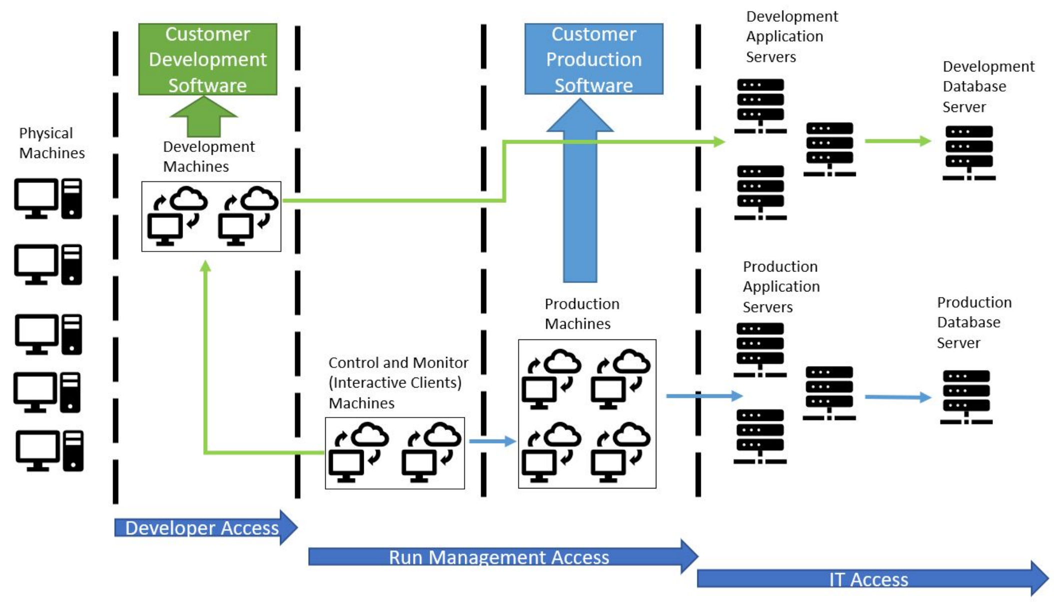

Physical Machines
Customer Development Software
Development Machines
Customer Production Software
Development Application Servers
Development Database Server
Production Application Servers
Production Database Server
Production Machines
Control and Monitor (Interactive Clients) Machines
Developer Access
Run Management Access
IT Access

Figure 2.6: Example of infrastructure that the company provides.

Figure 2.6 shows typical IT infrastructure that the company provides. The environment is split to development and production parts. From the development side there is only connections to customers development target systems against which the automation processes are being developed. From here the fully developed automations are deployed to production through interactive client (IC) machines. The idea of this is to minimize unwanted interactions with production. IC machines are also the place where the centralized production control (for example the terminated process restarts or environment configurations) and monitoring happens. Back-end side (application and database servers) is also separated to development and production. This enables that the new platform updates can be first installed to development so that an unexpected errors that those might bring up are noticed at the beginning, before they are deployed to production. Developers has access only to development parts, IT to server parts of the architecture and run management to customers development and production parts excluding the back-end servers. This division is indicated by the arrows at the bottom of the image.

For accessing the virtual IC or development machines from the physical machines, employees use Windows's remote desktop protocol (RDP) software. Over the RDP connection moves only pictures from the remote display [20]. In terms of robotic process automation this means that for identifying elements from the automated target systems there is no possibility to use software's internal structures. Element identification needs to be based on picture recognition, which makes the creation of this master's thesis automation more time consuming and the end result is more sensitive to changes in server configuration and automation tool updates.

2.2 Principles of text analysis

Techniques tested in this master's thesis are based purely on statistics. No advanced techniques with which more deeper meaning of sentences could be obtained are studied. This kind of technical research would take too much time and that is why they are let out of the scope on purpose. As described in the introduction, the purpose of the implemented machine-learning model is to predict the outcome of a received ticket (if simple software robot's restart could be a solution for solving an incident). Because we can label manually each previously obtained ticket or email based on a human professional review, the creation of accurate model becomes easier. In machine learning terms, this kind of approach where model is taught based on correct answers, is called supervised learning. In text analyzing field this is called document classification [21]. This is opposed to a labelled data problem that is unsupervised learning or document clustering (text analyzing), where the unlabelled input data should be divided mathematically into logical groups. Our problem has only two possible outputs ”restart helps” or ”restarts doesn't help”, so it is more precisely called binary classification problem. Prediction based on continuous data is called regression. Categorization of introduced terms is seen in figure 2.7.

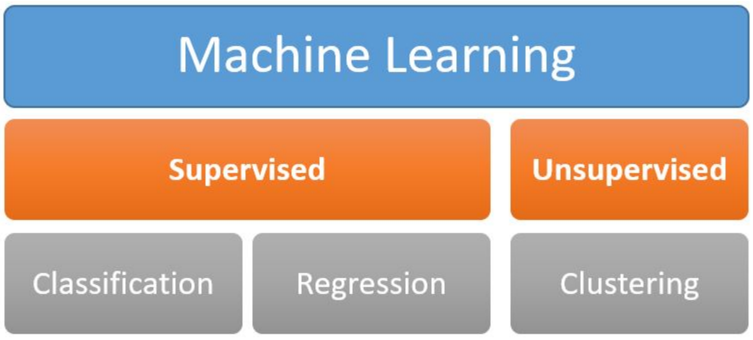


Figure 2.7: Model hierarchy.

The available data for building the machine learning model is in text format. At first look the textual data seems to differ a lot from numerical, but it can be transformed to spreadsheet format, so that the same methods can be used as with numerical data [21]. The basic idea is to gather words from documents (tickets or emails) into columns and put the documents into rows. If a word occurs in that document it is marked as one, otherwise as zero. This basic idea is presented in Table 2.1.

	Word 1	**Word 2**	**Word 3**	**Word 4**	**Word N**
Document 1	0	1	0	1	0
Document 2	1	1	1	0	0
Document 3	0	1	0	1	0
Document M	1	0	0	0	1

Table 2.1: The basic idea of the structure of text analyzing.

For example, the word number 4 appears in documents 1 and 3, but not in 2 and M. There are other variants of this presentations style, but this is a fundamental concept that works accurately with machine-learning algorithms [21].

The set of unique words is called a dictionary. Depending on the type of documents in the dataset one can imagine that a matrix formed from these might become very sparse, i.e. the ones and zeros in one row are very far away from each other. For example, if the examined dataset consists of emails and the dictionary has 10000 unique words, the investigated rows might have only 5-10 ones. If one document equals one book, the number of occurrences in one row is much higher. Text analyzing techniques assume sparse data because the dimensions of the handled matrices are large. Only positive values are processed, in which case the requirement for computational power doesn't grow too large. So if the analyzed data was books, it might be clever to split the books according to chapters or sections.

Too sparse data is also a problem that has effect to the taught model's prediction accuracy [22]. In literature this problem is referred to as "curse of dimensionality". In practice this is due to too many words meaning the same thing, which grows the dimensionality of the matrix used to teaching the model. Solution for this is to try to combine words meaning the same thing under the same variable, which reduces the matrix column dimensions and simultaneously increasing the frequency of words.

2.2.1 Initial data preparations

When dealing with text analyzing we should speak about tokens when meaning words in text documents. Each token is an instance of type, which means that there can exist multiple instances of the same type in a one sentence [21]. This definition helps us to remove useless information from our data when calculating word frequencies from documents. Removing useless information from documents in order to get accurate result, is a state of art. There isn't a method that suits for all. Almost always at first every word is put into lowercase. Typically, line endings and tabs (word delimiters) and special characters (punctuations) can also be removed, but that's not always the case. Example when analyzing tweets from Twitter, the hashtags might give some valuable information and commas are used for many numerical values. Tokens can also be modified to achieve higher frequencies and to reduce the matrix dimensions. For example, we can modify the documents so that every email address (firstname.lastname@company.com) is transferred only to a token "email" if we are not interested in the frequencies of the precise email addresses. The analyst has to do the initial preparations based on the actual purpose. The more data is removed, merged or modified, the higher the frequencies.

Reducing the number of similar features or types is called word stemming. In practice this means the removal of word plurals and verb bending (teach, taught, taught). The main goal is to return the word to its basic form without losing the actual purpose of the word. Words that usually don't give any valuable information for prediction like articles "a" and "an", are called stop words. Usually they occur in almost every document and because of that, they can be removed. Fortunately, there are ready-made algorithms for these purposes and there is no need to use time for creating custom functions. In other words, the algorithms do not survive the different languages and problematic phrasing. Those can be gathered as a list, that is then given as an input for a function. All the above-mentioned techniques need to be tested with the obtained datasets and the frequency of occurring words needs to be studied for example through a graphical review.

2.2.2 Feature engineering and selection

The goal of feature engineering and selection is to identify and modify features from the given dataset that are most relevant for the prediction. This is because some words are much more likely to be correlated with the case that is trying to be predicted [23]. As already brought out in the previous chapter, the information about token frequencies is useful for forecasting and it needs to be used somehow. Most frequent words might occur in every document and that is why they might not bring any valuable information. It is the same case with the less frequent ones.

Raw zeros and ones are modified so that we can obtain more information about the importance of words in order to get more accurate predictions from the created model. The easiest way is to count the occurrence of same words in one document and then replace the type column's number with the obtained sum. For the same idea there are also better algorithms. Next step is to count term frequency (TF), in which the sum of type instances in one document is divided by the sum of document's all instances. TF's formula is seen in Equation 2.1.

$$tf(t,d) = \frac{f_{t,d}}{\sum_{n=1}^{m} t', d} \tag{2.1}$$

Perhaps the most commonly known frequency-based modification algorithm is term frequency-inverse document frequency (TF-IDF) method, in which the idea is to provide more information on how valuable the word is across the entire set of documents (corpus). First, we need to calculate TF. After that is calculated the IDF, that is a scale factor for term frequency. It is obtained by dividing the total number of documents by the number of documents where the feature appears and scaling it to logarithmic scale. TF-IDF can be see in Equation 2.2.

$$tfidf(t,d,D) = \frac{f_{t,d}}{\sum_{n=1}^{m} t', d} * \log \frac{N}{n_t} \tag{2.2}$$

Because the scale is inverse, when a word appears in many documents it is considered unimportant and the scale is lowered [21].

2.3 Model performance metrics

It doesn't matter if the input data is numerical or textual, the same basic principles and methods apply when building and evaluating a prediction model. First the data set needs to be split into training and testing sets. Basic strategies are to split 70 % for training and 30 % testing or 70 % testing, 15 % cross-validation and 15 % testing. In the latter case, cross-validation is used for tuning the model parameters before the actual results are obtained from the test set. If the data set is small, better results might be obtained by dividing it into 10 equal random splits using nine sets for training a model, testing each of them with the remaining 10 % of data and picking up the model which produces the best results [21].

One should also focus on the time period that the data has been collected during. For example, in the ticket case, the whole data set might include cases from companies that are not customers anymore. It might give indication on the model's prediction power in the future, if we use older data for training and newer for testing. This approach simulates the situation where a model is not re-trained continuously and it needs to behave correctly a few months later [21]. By randomizing the data before splitting we might end up creating a more generic model because the both data sets should include less similar cases from the same time period. But doing this we might lose the above-mentioned future prediction power.

Model's performance can be measured in many ways. For a model that categorizes text, typical metrics are accuracy, precision and recall. All of these are easy to calculate but result evaluation for precision and recall needs a little bit of understanding. At first, we need to think what the possible outcomes are. Our model can predict true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Correct cases are TP and TN and falsely predicted outcomes are FP and FN. Idea of this is seen in Table 2.2.

	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Table 2.2: Possible prediction outcomes.

Accuracy is the most easiest of the three to understand and it is calculated as a sum of actual correct values divided by all predicted outcomes (equation 2.3). It can be seen as a guiding value for a model's performance, but it leaves important features of the prediction that are also needed (precision and recall) unseen.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

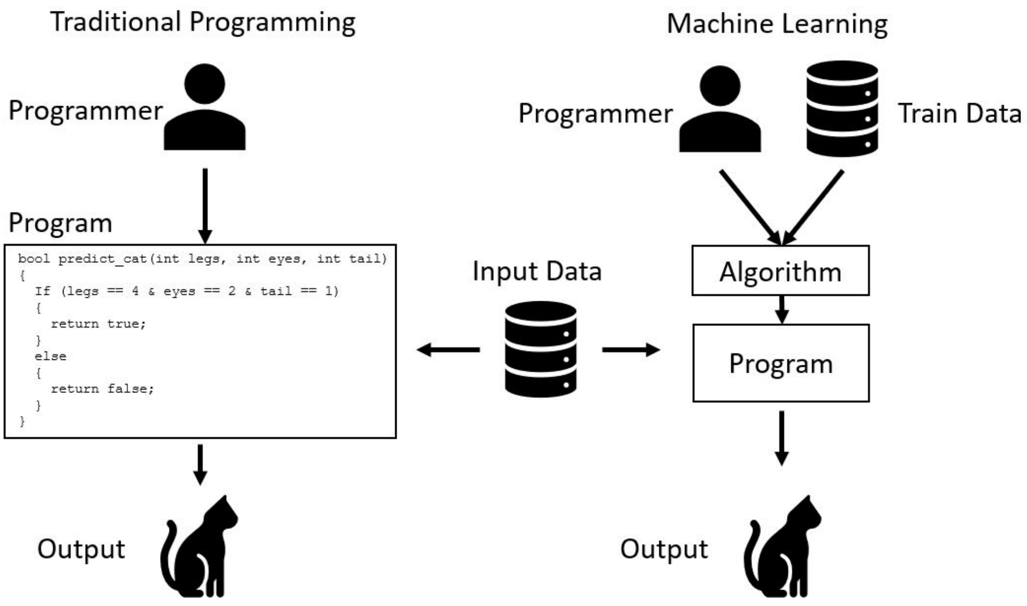
$$Recall = \frac{TP}{TP + FN} \tag{2.5}$$

For example, let's think different precision and recall values in a case of an e-mail filter that tries to predict if an e-mail is spam or not. High precision means that when the model decides to move e-mail to the trash folder, this is typically wanted behavior. Low precision means that it might also move the un-spam emails to trash because the number of false positives are high. When the recall is low, our predictor leaves many spam e-mails to inbox, which is also unwanted behavior. It is also possible to adjust the model's recall to be always 100 %, which means that every received e-mail is spam. This also increases accuracy in a case when an early built model has predicted that results where TP<FP or TN<FN. This is one of the reasons why all of the three key metrics are needed, and the obtained results need to be investigated carefully.

2.4 Machine learning

In this section is introduced machine learning algorithms that are typically found to be the most suitable for text classification. From the perspective of the pilot project's success it is enough that the prediction model gives accurate values and its accuracy can later be refined. Machine learning algorithms introduced in this master's thesis are based on supervised learning and a selection of later tested algorithms is based on observations collected during literature review.

Machine learning is a subpart of artificial intelligence, that studies how computers can be programmed to do tasks without explicitly programming them to do it. The idea is to create an algorithm that can learn from data or create a program that can learn a task by itself. Learning means that with every iteration the performance is improved, and the program succeeds better in the given task. Typically, this technically means finding a global minimum for a cost function. It can be viewed that in traditional programming a programmer creates a software that gives an output and in machine learning she or he creates a system that creates the actual prediction program. The idea is seen in Figure 2.8



```
bool predict_cat(int legs, int eyes, int tail)
{
  If (legs == 4 & eyes == 2 & tail == 1)
  {
    return true;
  }
  else
  {
    return false;
  }
}
```

Figure 2.8: Traditional programming versus machine learning.

Compared to traditional computer programs where performance can be increased by coding more rows, machine learning programs performance can be increased by giving it more data and iterations. Performance cannot be endlessly improved by just pouring new data and there exists a threshold value for the number of iterations, that needs to be found. Still there are many examples where machine learning programs have beat human experts in specific tasks and thus exceeded the human performance capabilities.

2.4.1 Neural networks

There are many variations of neural networks' structures and their learning methods. Basic feed forward structure with backpropagation feedback should be enough for obtaining desired results from simple text classification problem [23]. That is why more advanced techniques are not introduced in this thesis.

Artificial neural networks are a biologically inspired field of research where the goal is to create an algorithm that mimics function of the brains. The network is composed of multiple nodes (neurons) which are connected to each other with weights (synapses). In basic structure there typically exists an input layer, one or more hidden layers and an output layer. Every node from one level is connected to every node in the second level. This kind of architecture is called a fully connected neural network. This basic structure is seen in Figure 2.9. Input data is passed to input layer and then flows through the hidden layers and prediction comes out from the output layer. In the input layer there are as many nodes as there are features in the training set. For example, if the input data is 100 x 100 pixel pictures we need 10000 nodes in the input layer. The number of hidden layers depends on how hard the prediction problem is. The more layers there is the more computational power is needed. In image classification problems there might exist multiple hundreds hidden layers and because of that it is often talked about deep neural networks. There exists no rule of thumb for the number of nodes in the hidden layers and the most suitable number needs to be tested and selected by the user. The number of nodes in the output layer depends on the problem. For example, in binary classification there can exist only one output node.

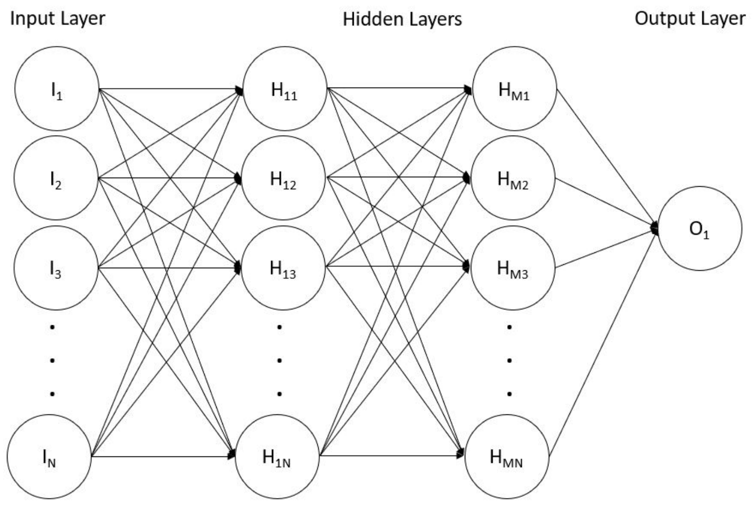


Figure 2.9: Basic structure of fully connected neural network.

Training the neural networks means basically adjusting its weights. One can imagine that the training goes from left to right and after each iteration network gives a response. During the iteration a weighted sum for every node (S) is calculated based on the incoming weights (w) (equation 2.6).

$$S_j = \sum_{i=0}^{M-1} (a_i w_{ij}) \tag{2.6}$$

Sum is used to compute node's output, typically with a sigmoidal activation function (equation 2.7). If the output value is above mid-point of this

S-shaped curve, the node is activated. There are also other variations but for the simplicity only this one is tested.

$$f(S_j) = \frac{1}{1 + e^{-S_j}} \tag{2.7}$$

At the end of the iteration every nodes error is calculated based on the actual predefined correct value. Method for this is called gradient descent [24]. Based on the error, weights are tweaked to obtain better results in the next iteration. The amount of the changed value during one iteration can be adjusted with a learning rate that determines how fast the network learns. Wrongly selected value can lead to a situation where optimal minimum value for cost function can't be found. After multiple iterations, network should converge to predict the desired outcomes.

2.4.2 Support vector machine

Support vector machine (SVM) is one of the most widely used general classifiers. It was introduced by Bernhard E. Boser, Isabelle Guyon and Vladimir Vapnik in 1992 [24]. The SVM algorithm is also found to be very suitable for text data categorization problems because of the sparse high-dimensional nature of text. In this case a few features are irrelevant, but they tend to correlate with one another and generally organized into linearly separable categories [23]. The main principle is to determine separators (vectors) in the feature space that can optimally separate the different classes. Two dimensional example of this is illustrated in Figure 2.10, where vector 2. is the best choice because it has the largest normal distance to the nearest data points.

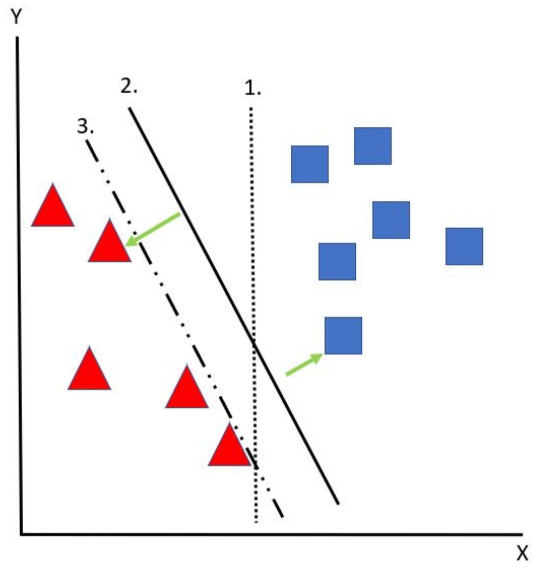


Figure 2.10: Three vectors separating two classes.

In high dimensional feature space, like with text classification problems, the SVM-algorithm constructs a set of hyperplanes for separating classes. If data is not linearly separable there can be used a method called kernel-trick that transforms non-linear problem to linear. Mathematics related to support vector machines are not introduced in this master's thesis. Almost every analyzing tool has a build-in SVM-function that can be used. Only the performance metrics and function's input hyperparameters (parameters which algorithm doesn't learn by itself) are studied more closely.

2.5 Web service

Web service is a programming technique where the idea is to enable machine-to-machine messaging over communication network [25]. Typical implementation is divided into client and server parts where the communication happens over the World Wide Web. Both can be done independently without the other one knowing about the other, which means that the software can be updated and modified separately. It is enough that the structure of information flowing between systems remains as the same. One server can be called by many clients and one client can call multiple servers [26], but in this work is only important that one server (where the machine learning model sits) can be called by multiple clients. In general this kind of software architecture, where modular components interact with each other through a communication protocol over a network, is called a service-oriented architecture (SOA) [27]. Web services are just one way to implement this approach and the later introduced Representational state transfer (REST) is one standardized way to implement web services.

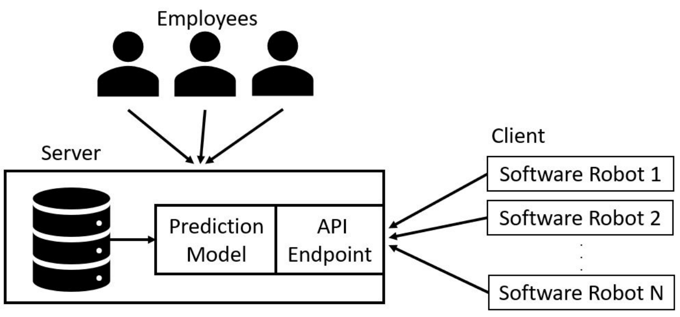


Figure 2.11: Prediction web service. Ideal situation where multiple developers have access to modify the system and multiple processes can utilize its endpoint.

The company that this master's thesis is made for wants that the created machine learning model could be used with multiple software robots from different locations. This means that it needs be built as a cloud service. It would be possible to build own web service but commercial software as a service (SaaS) solutions also exist. Google, Amazon and Microsoft offer cloud platforms for analytic purposes with a build-in possibility to create web service endpoints for building prediction models. Created model can be therefore used from outside through an application programming interface (API). Principle of this is seen in the Figure 2.11. In this master's thesis Microsoft's Azure Machine Learning (AML) service is investigated more deeply because the company is already using other Microsoft's Azure services. It can be used easily with any browser and the basic account, that is enough for testing purposes, is free of charge. For building a model AML offers default machine learning functions and also a possibility to implement own code blocks with Python or R programming languages. Therefore, we can use the best chosen solution for our text analyzing purposes and are not limited by pre-implemented algorithms. Using a commercial cloud solution also enables other benefits. After the model has been created, tested and put into operation, accounts can be created for other employees who can copy, use and modify created solutions for their own purposes. Cloud repository helps code's re-usability and improves cooperation between employees.

Cloud analytic tools provide demo projects that set a good rule of thumb on what the created models should look like. This improves traceability and maintainability of the solutions.

2.5.1 Application programming interface

Microsoft Azure Machine Learning uses REpresentational State Transfer (REST) architectural style for API implementation. When using a normal web browser (client) for navigating to a website (server) you type a uniform resource locator (URL) to the address bar. Then the web service on the other end replies and sends you the information from which the web page is created. URL is a specific type of uniform resource identifier (URI), that REST utilizes. URI is a string of characters, like URL. It is used to identify a resource, transfer input parameters and output results. When the command is send to web service through HTTP or HTTPS protocol the received response arrives typically in JSON (JavaScript Object Notation) or XML (Extensible Markup Language) format. HTTP/1.1 standard RFC2616 defines a set of methods with which it is possible to interact with REST web service [28]. Typically REST architecture offers for the user in the client end GET, POST, PUT and DELETE methods. For example, when using a GET-operation, user asks the web service to reply for a question. In our text prediction case this means that when a ticket's body text is sent to AML's model endpoint, it answers the prediction in JSON format. This predefined set of available operations in REST makes sure that the server developer creates an end results that other software developers can easily make use of in their own client implementations.

Chapter 3

Implementation and results

This chapter introduces the created automation solution. The entity is logically split into four sections: Input datasets, Model training, Web service and Software robot. Each section handles the most important things for the individual part of the automation solution. The goal is that after reading the chapter the reader understands the created solution and can critically review the automation and the obtained results.

At the beginning of the work, the analysed textual data had to be gathered as emails from Microsoft's Outlook and company's service management system. After the basic visualization and analysis, the dataset was modified into two a dimensional matrix. The matrix dimensions and the numerical content inside of it were adjusted with text modification algorithms so that when it would be later fed for a machine learning algorithm, the model would give the best possible predictions. After that, the selected text modification techniques were tested against chosen machine learning algorithms. All above listed tasks were initially tested on own local computer to obtain quick results. When it was assured that the chosen solutions could produce the desired results, the selected combinations were transferred to the cloud where the development of the machine learning model continued for the rest of the time.

Lastly, the actual software robot was created. Automation process picks new tickets from company's ITSM system and handles them one by one. Every ticket message's textual body is sent to prediction web service and handled according to the received output. During the work was also created a logic for re-training the prediction model continuously. Every ticket is appended to training dataset and labelled correctly with software robot. After that a small .exe-formatted program can be launched manually or in a scheduled way which re-trains automatically the prediction model with appended training dataset. With this logic, the model's predictive power should remain in the future, if the content of the tickets' changes.

3.1 Input datasets

The available datasets are e-mails and tickets both from a period of about two years. Sets have additional information like email sender, receiver and tickets resolution time, that don't offer any prediction power considering the implemented solution. Those are cut off so that we are only using features in which the problem is explained as text. E-mails have subject and body columns and tickets description column. For the model to be taught to mimic human reasoning, a pre-determined correct answer is required. In practice this means that each row in the dataset needs to have a binary value that determines if the software robot's restart could help solve the problem. Because these answers don't exist in advance, they need to be added manually. The e-mail set has 3527 rows and ticket set 7965 rows. In the beginning unnecessary items are removed from ticket dataset, which leaves 7412 tickets remaining.

Emails are exported from a mailbox that only has e-mails concerning problems with software robots, while tickets also include messages that concern company's other IT-problems and service requests. Coloured purple in the figure 3.1 we can see how the ticket amounts are divided. Other (IT, internal, etc.) has 3080, RM 3744 and tickets without group 588 items.

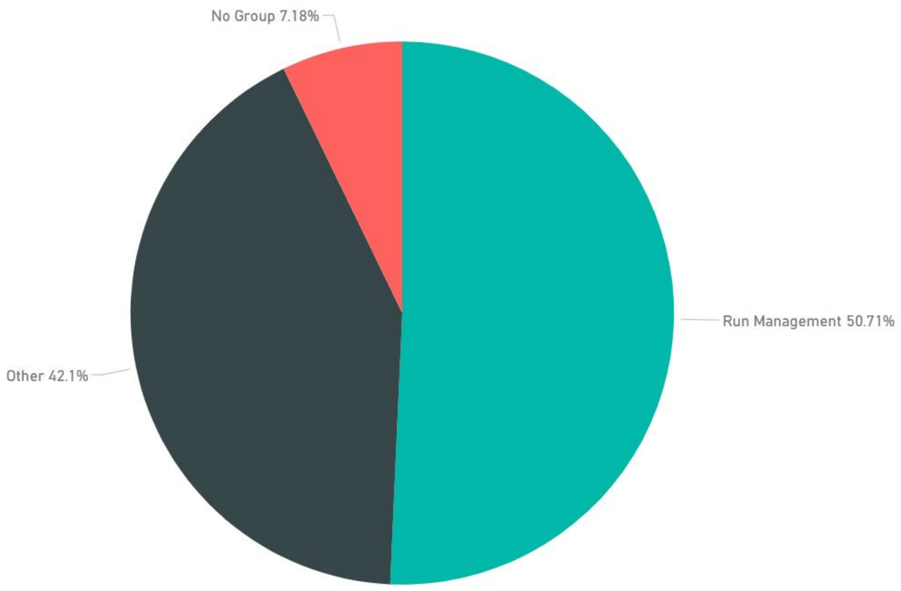


Figure 3.1: Ticket dataset by group.

Email dataset includes human-written and automatically generated messages. Messages from humans are typically received because they have noticed that software the robot's run has not successfully completed. These messages are unique because humans write different e-mails every time. Automatically generated messages include also information about successfully completed runs. Software robots are programmed to send these so that previously unrecognized errors, e.g. infinite software loops could be caught more quickly without time delays for customers. Automatically generated messages about successfully completed runs are very generic and the same messages occur frequently. This is not the case with automatically generated error messages because software robots are programmed to send details about the problem reasons, which makes the messages different from each other. This can be seen in figure 3.2 that shows the top 30 list of most frequent email body texts. One bar describes the amount of exactly similar messages. Green colour means that software robot's restart isn't needed when this message is received, and red colour means that the restart could help to solve the problem.

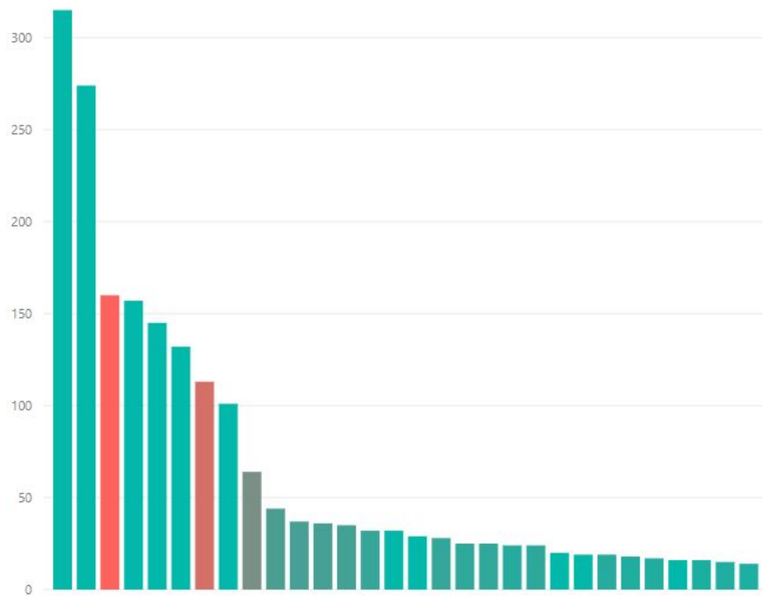


Figure 3.2: Top 30 list of most frequent messages. Colour reflects label value.

Tickets are divided in two groups: service requests (SR) and incidents. SR's include according to their name requests for services like resetting passwords, opening IP's in firewall or some robotic process related request that is not related to any problem. Incidents on the other hand are sent only when something is wrong. Most of company's tickets are coming from automated alerting sources which can be seen in figure 3.3.

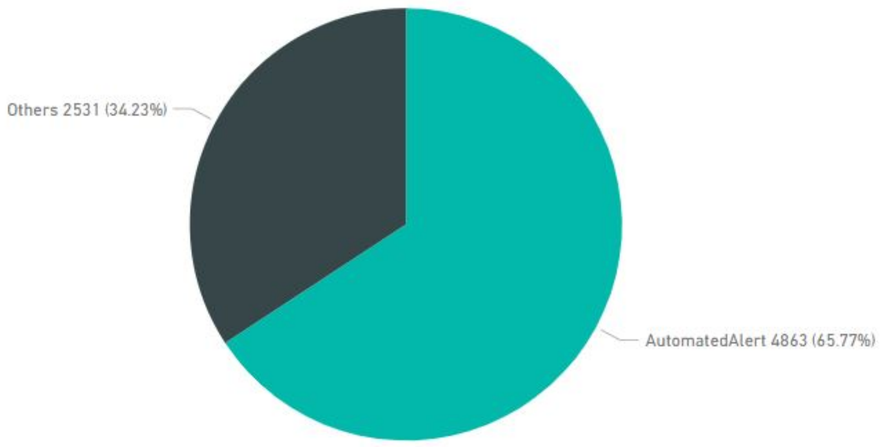


Figure 3.3: Ticket partition to automated and others.

On higher level automated alerts concerns RPA platform- and robotic process maintenance. Distribution of automated alerts can be seen in figure 3.4.

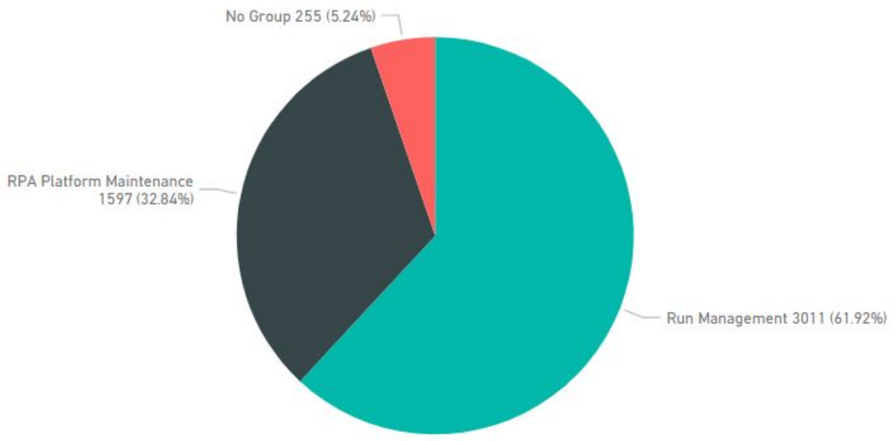


Figure 3.4: Automated ticket distribution.

The part of automated alerts concerning Run Management can be divided in three groups: login fails, task fails and process terminations. Login fails means that the software robot's scheduled run fails right at the beginning when automated sessions fails to log into operating system. Task fails are not very common and they happen when the production machine fails to construct a connection to the application server that takes care of the schedule launches and the actual scheduled automated process is skipped. Process termination comes basically when the automated process fails to perform the predefined workflow and crashes.

In conclusion we can extract the same kind of informative textual messages from email and ticket datasets, which can then be used to train the model. After the manual labelling we can see from the figure 3.5 that both sets have quite a similar distribution of positively and negatively labelled events.

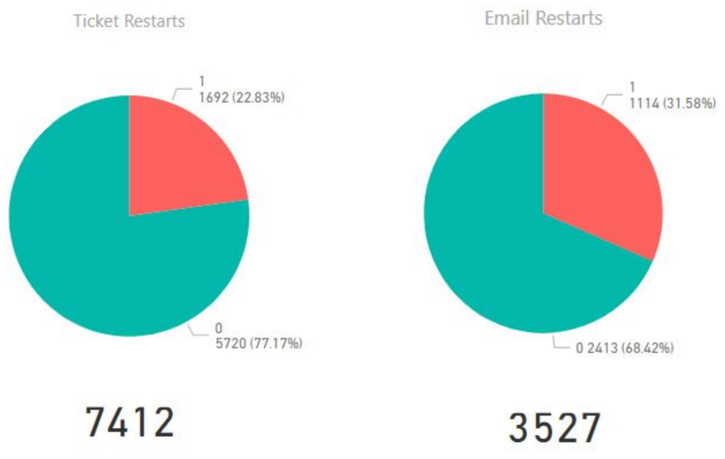


Figure 3.5: Restart distribution of both datasets.

Both have quite a lot of frequently repeated similar messages, which when processed wrongly can lead to model over-fitting. Most of the emails and tickets are written in English but there are also messages in Finnish and Swedish. This can become a problem when trying to set words in their basic forms. Next the contains of both sets are investigated and tested to see how good prediction power could be achieved with them. The both data sets are also united to obtain more rows in the training set.

3.1.1 Text preparations

If every word, number and special mark would be used to train the machine learning model as it is presented in the description text, we would have a huge variety of multiple features for describing the same thing. Feature space would grow unnecessarily large and we would lose a lot from the model's prediction power. That is why we try to get rid of useless features and combine similar words under the same feature column. Every word is lowercased and tried to return to its basic form. For this we are using R's tm library, which is a function collection developed for text mining purposes [29]. It provides among many other things pre-programmed functions to lowercase characters, remove numbers (digits), remove special characters (punctuations), remove stop words and word stemming.

Combination	Amount of Features	Top 10 freq avg	Top 100 freq avg	Avg freq all
Plain Text	5363	3656	1091	41
Lowercase	4855	4297	1155	45
Low + Digits	4574	4205	1126	46
Low + Dig + Special Characters	4572	4205	1126	46
Low + Dig + SpeChar + Stop words	4531	4205	1089	45
Low + Dig + SpeChar + Stemming	3275	3738	1079	59
Low + Dig + SpeChar + StopW + Stem	3254	3727	1049	57

Table 3.1: The amount of features after modifications in email dataset.

In table 3.1 it is illustrated how these feature modification techniques and their combinations have affected the total amount of features in the email dataset. Table also shows the average frequency after each modification when there is selected top 10, top 100 and all word frequencies. Because the used dataset includes e-mails written in Finnish it might be a problem that the used functions transform Scandinavian characters ä, ö and å into a undesired form due to encoding problems. In table 3.2 the same text modification techniques are tested but the Scandinavian characters are transformed into the same format as used in English. Ä is transformed to ae, ö to oe and å to ao.

Combination	Amount of Features	Top 10 freq avg	Top 100 freq avg	Avg freq all
Scandinavian to English	5688	3656	1061	37
Lowercase + Scandinavian	5142	4299	1124	41
Low + Scan + Digits	4885	4205	1096	42
Low + Scan + Dig + Special Characters	4883	4205	1096	42
Low + Scan + Dig + SpeChar + Stop words	4846	4205	1069	40
Low + Scan + Dig + SpeChar + Stemming	3775	3738	1069	52
Low + Scan + Dig + SpeChar + StopW + Stem	3757	3711	1037	40

Table 3.2: The amount of features after modifications with Scandinavian character transformation in email dataset.

In table 3.3 the same text modifications are done for the ticket dataset.

Combination	Amount of Features	Top 10 freq avg	Top 100 freq avg	Avg freq all
Plain Text	30499	27610	11132	63(1681)
Lowercase	27276	28250	11662	70(1707)
Low + Digits	17564	25212	10688	94(1530)
Low + Dig + Special Characters	20255	22997	8029	60(1123)
Low + Dig + SpeChar + Stop words	20145	20445	6658	51(933)
Low + Dig + SpeChar + Stemming	18952	23134	8280	54(1130)
Low + Dig + SpeChar + StopW + Stem	18854	20603	6908	54(941)

Table 3.3: The amount of features after modification in ticket dataset.

Last text preparation test is done for combined dataset which has 10939 (7412 + 3527) rows. With this we should be able to train one universal model that would have better prediction power than a model trained with either of individual datasets. This feature reduction can be seen in table 3.4.

Combination	Amount of Features	Top 10 freq avg	Top 100 freq avg	Avg freq all
Plain Text	33060	27665	11276	64(1799)
Lowercase	29533	28316	11836	72(1836)
Low + Digits	19602	25282	10867	95(1666)
Low + Dig + Special Characters	22567	23070	8079	62(1227)
Low + Dig + SpeChar + Stop words	22457	20453	6725	53(1037)
Low + Dig + SpeChar + Stemming	21247	23207	8340	66(1238)
Low + Dig + SpeChar + StopW + Stem	21149	20611	6990	57(1048)

Table 3.4: The amount of features after modification in combined dataset.

3.1.2 Feature selection

For easier result review in figure 3.6 is plotted how each modification has affected each dataset. With the email dataset we can see from both tables (normal and Scandinavian transformation) that in major cases after each modification step is executed the number of features is reduced while mean value calculated over all feature frequencies is increased (first and last columns). This is the desired behaviour when similar words are combined under one feature. It is not so unambiguous how the average frequencies behave when only 10 or 100 most frequent features are selected (two middle columns). After lowercasing words, the average values increase to the highest points. After each new edit has been executed frequency value decreases. Removing digits removes often occurring numerical features like current year from top average calculations, which explains part of the problem. Removing special characters and stop words also decreases the values, which is not desired behaviour. Manual investigation doesn't reveal any explaining causes and features just need to be tested when training the model.

When Scandinavian letters are transformed to a more English-like format, overall frequencies are decreasing, compared to results without the modification. Also, the number of features is higher. Manual investigation doesn't reveal any bigger or unwanted behaviour in the feature data matrix. Even after the modifications the Scandinavian characters remained in the original places. Due to the above-mentioned facts, Scandinavian transformation will not be used if there aren't problems with text encoding. There are more features and average frequencies have lower value in every examined category.

Tickets and combined datasets have similar behaviour after modification steps because the ticket set has much greater share than emails in the combined dataset.

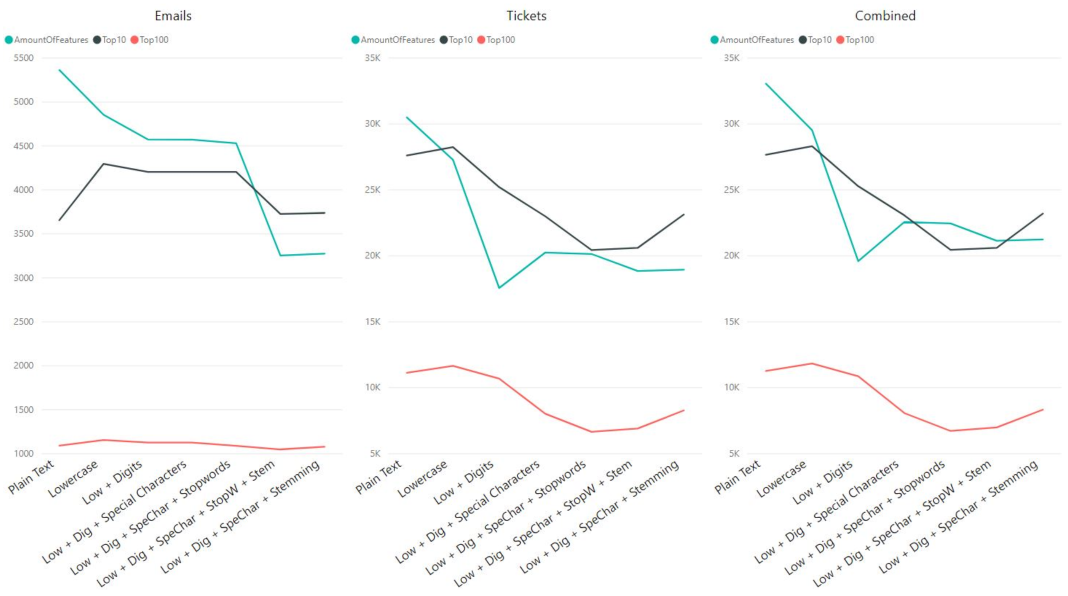


Figure 3.6: Feature reduction techniques for each dataset.

Removing digits has the biggest individual impact to the total number of features in ticket and combined datasets. It decreases the value with almost 10000 features. Adding modification techniques after that doesn't bring any value because the modifications decrease the average frequencies while simultaneously decreasing the number of features. Still it needs to be kept in mind that for example punctuational characters should not bring any value for the implemented model because the model should be as general as possible and special things in training dataset can lead to model over-fitting.

Next thing is to cut off features that are incorrectly formed or that can lead to model over-fitting. As explained above, next introduced methods need to be tested with actual model so that we can obtain the best combination. In the start the easiest way is to remove the words that are too long or too short. For example, with the combined datasets, in a case where lowercase and digits are removed when words that have less than 4 or more than 20 characters are removed, the feature space drops from 19602 to 17811. When all feature combining functions are applied drop is from 21149 to 16961. This modification has only meaningful impact to feature space and not to frequencies. When the actual model is taught it is also tested what kind of impact removing most- and less frequent words will have.

3.1.3 Frequency matrices

In order that textual data can be entered for machine learning algorithm it needs to be transformed into two-dimensional matrix (messages as rows and features as columns which means 10939 x SelectedNumberOfFeatures). Custom functions for this could be coded easily but R's tm-library already offers pre-created functions [29].

There are three options from which to choose. Binary weighting in which feature is set as zero or one depending on if the feature exists in message. This is the easiest weighting method. In term frequency-based weighting (TF) is calculated how many times feature appears in message and the sum as a row value is set. TF gives more value to words that occur often. Third option is called Term Frequency - Inverse Document Frequency (TF-IDF). The idea of this is to multiply the message terms by inverse, using the number of feature occurrences in the whole dataset. This method turns the weighting upside down, giving the highest value to words that occur infrequently.

3.2 Model training

In this section the created and tested text prediction models are introduced. The main goal is to find the best combination that could give accurate predictions also in the near future with new input data. During the pilot project no material was collected from long-term usage and because of this the studied measurements need to be easily calculable. The biggest thing that needs to be investigated later is how machine learning models' prediction efficiency will change over time. This was left out of the master's thesis' scope on purpose because the variation between received tickets during the pilot project was relative slow. It was also tested with small sample that the current model will not change remarkably in a small period.

For implementing own custom models, Azure Machine Learning service offers Python and R programming languages. With both it is possible to implement the same desired functionalities. They offer extensive ready-made machine learning libraries and the learning threshold of both languages is equally low. For this master's thesis we selected R from these two. This is because the writer has already pre-knowledge from Python and it is wanted to see how R suits for machine learning and text analysing purposes. Testing phase will be done with software called R-Studio on writer's own computer. At the beginning of the project it was found that the development phase can be largely realized without the benefits of cloud computing. Last steps starting from machine learning model training will be implemented directly into the cloud with Azure Machine Learning Studio that offers the needed computer power and possibility to publish model as web service that can be used through REST API.

As explained earlier, initial tests were done on a personal computer. During the early phase of development, it was identified what text modification and feature weighting techniques should be tested and how they react with normal neural network and support vector machine models with different parameter combinations. Initial model accuracy scores were well aligned with text feature reduction results. During initial testing it was kept in mind that the quality code developed during that time could be almost directly transferred to Azure's final solution, which turned out to be a useful choice later.

Azure machine learning studio provides opportunity to create own models using R's libraries or select one from pre-created initial models [30][31]. There is also possibility to tune created model hyperparameters with external block [32]. This functionality automatically tests different variations of those model parameters that would be otherwise chosen manually so that the highest value of selected performance metric (Precision, Recall, Accuracy) is obtained. After that the model can be cross validated [33]. This means that the final model can be tested with different small randomly selected partitions of the initial data. This allows the model to be simulated with unknown future input data and it should reveal over-fitting problems. Cross-Validation evaluation is computationally more intensive so it will be only used after couple of promising models are obtained.

First is tested the initial models that AML studio provides with model hyperparameter tuning and model cross-validation. If reasonably good results are not obtained with this combination it is tested to create own models. Different models will be tested with:

- Different combinations of feature reduction and modifications techniques.
- Different training and testing dataset divisions.
- Different machine learning algorithms.
- With hyperparameter tuning.
- With cross-validation.

AML provides two interfaces for model creation. There are notebooks that remind of normal integrated development environment (IDE) like R-Studio where the code is written as a line from line in text format. Second option is called experiments, which are more graphical and it reminds IEC 61131-3 function block diagram programming [18]. There logical steps are separated as own chunks of code and blocks are graphically connected to each other in right order. From these two options we choose the experiments because their way of coding reminds the RPA software that the company are using. By choosing this it is tried to make the approach easier for other employees. In figure 3.7 is seen an example of how this looks like in AML studio.

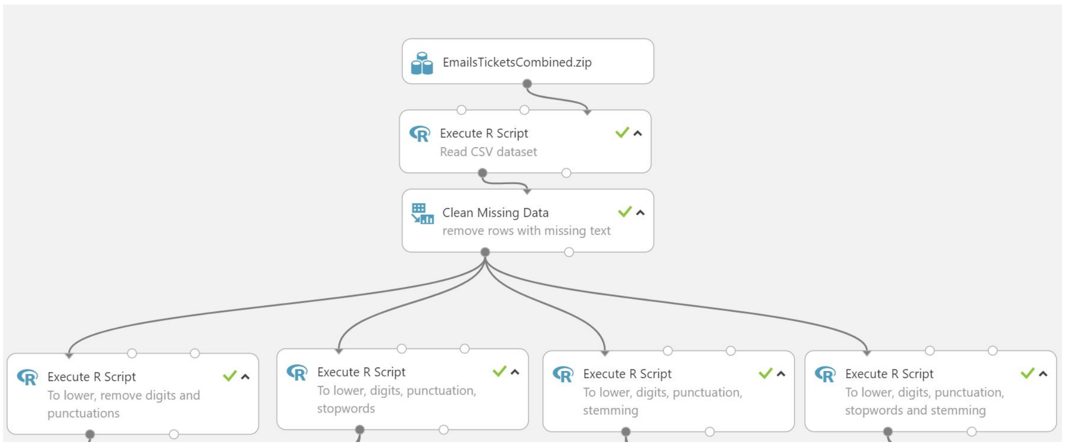

EmailsTicketsCombined.zip
Execute R Script
Read CSV dataset
Clean Missing Data
remove rows with missing text
Execute R Script
To lower, remove digits and punctuations
Execute R Script
To lower, digits, punctuation, stopwords
Execute R Script
To lower, digits, punctuation, stemming
Execute R Script
To lower, digits, punctuation, stopwords and stemming

Figure 3.7: AML user interface with initial data preparation blocks.

At first the data is imported and read to a table from a csv-file. After that the empty rows without pre-given prediction value are cleaned and initial text preparation functions executed. At this point it was noticed that AML doesn't yet have support for some non-English character encodings, which collapsed the system's execution. This was solved by removing all special characters from the dataset. Special characters exist rarely and almost only in people's first or last names, which will not have any impact on the finally selected model. Next the data is transformed into weighted feature matrix (binary, TF or TF-IDF) format and divided to training and testing sets and passed for model training algorithms.

3.2.1 Models

As described in the background investigation section of the thesis, two machine learning algorithms will be tested: support vector machine (SVM) and simple fully connected neural network (NN). AML provides own versions of these algorithms that are configured especially for binary classification problem. Documentation doesn't reveal any deeper mathematical details about the algorithm underneath the blocks but it is assumed that these use standard implementations.

First is tested which algorithm gives better results with initial parameters with different text parsing methods and different feature weighting methods. In figure 3.7 is shown the initial training setup for NN models where the input text data is lowered and digits, special characters and stop words removed. At this point the dataset is divided: 70 % for training and 30 % for testing the model.

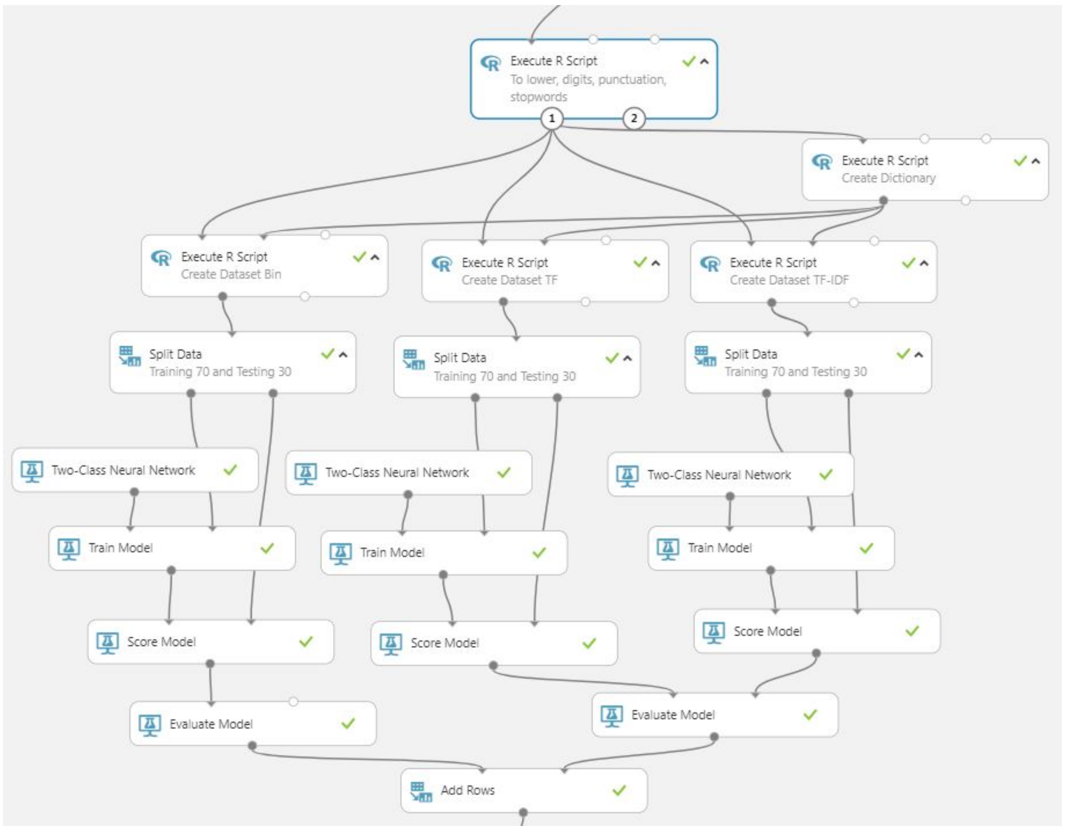


Figure 3.8: the initial training setup for one text parsing method testing.

After that both algorithms are tested with same inputs when hyperparameter tuning is enabled. For hyperparameter tweaking the data is divided as 70 % for training, 15 % for parameters tuning and 15 % for testing. Figure 3.9 shows what one test run setup looks like. Same kind of setup (4 different text parsing techniques and all with 3 different feature weighting methods) was used with initial model testing.

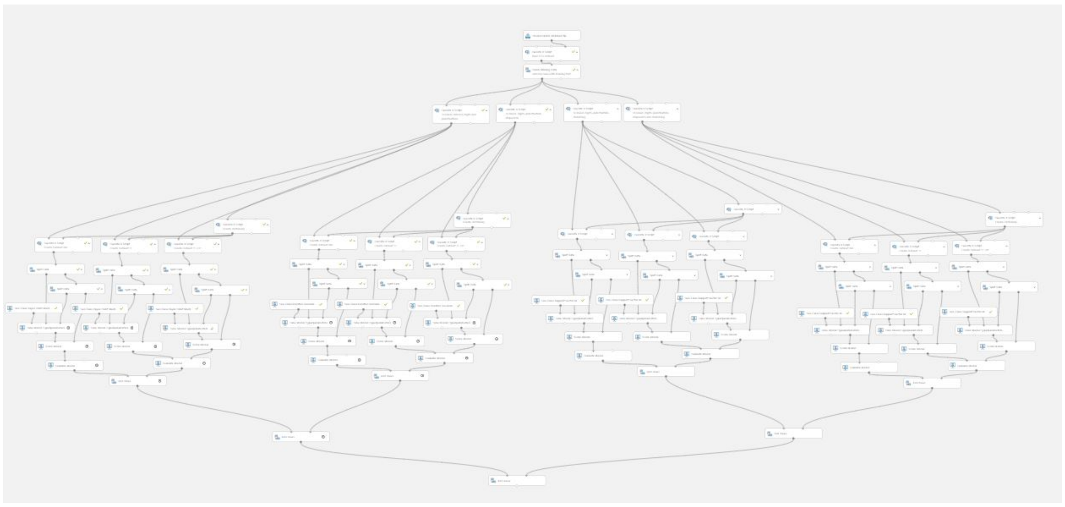


Figure 3.9: The initial training setup for testing one of two tested algorithms.

In AML the used SVM algorithm is called "Two Class Support Vector Machine" and it provides two hyperparameters that can be tweaked: number of iterations and lambda. Number of iterations is used to control trade-off between training speed and accuracy and lambda is a regularization coefficient (typically marked as L1) which is used to penalize more complex models in training phase [34]. Best combination for these values can be found programmatically with hyperparameter tuning.

Azure machine learning studio's binary neural network model is called as "Two Class Neural Network". It provides wider setup of hyperparameters: number of hidden nodes, learning rate, number of learning iterations, initial learning weight, momentum and the type of normalizer [35]. With hyperparameter tuning it is possible to find the best values for learning rate and for the number of iterations, but other parameters need to be tweaked manually.

Lastly the models with best accuracy were tested with a cross-validation check, where the selected number of input data is divided in 10 equally sized partitions, which are then used for testing.

3.2.2 Results

In initial cloud testing phase, it was found that the neural network model with initial hyperparameter values gives better results than the SVM when examining average values. Results of this can be seen in table 3.5. Although the differences are small, they are noticeable.

Algorithm	Highest Accuracy	Avg Accuracy	Avg Precision	Avg Recall
SVM	0.934	0.925	0.876	0.828
NN	0.947	0.942	0.885	0.920

Table 3.5: Results from initial models.

Table 3.6 shows results when hyperparameter tuning was enabled. When compared to the initial models, measured metric values were slightly improved. Decrease occurred only with neural network's recall. From the final model's perspective, a higher precision has more value than recall in desired prediction behaviour because it emphasizes cases where prediction is absolutely sure about what will be done. Highest accuracy in SVM case (0.948) was received when text was lowercased and digits, special characters and stop words removed with term frequency (TF) weighting. In NN case (0.951) initial part was same but without stop word removing and used weighting was binary.

Algorithm	Highest Accuracy	Avg Accuracy	Avg Precision	Avg Recall
SVM	0.948	0.943	0.922	0.856
NN	0.951	0.945	0.916	0.870

Table 3.6: Results from hyperparameter tuning test.

In general, it can be said that none of the tested feature parsing method did not rise above others. This can be seen from table 3.7 which shows the average model accuracy calculated over all tested models.

Lower+Digit+SpeChar(LDS)	LDS+Stop words	LDS+Stemming	LDS+Stop+Stem
0.938	0.940	0.939	0.937

Table 3.7: Models' average accuracies over different feature parsing methods.

Table 3.8 shows same kind of minimal differences between different weighting methods.

Binary	TF	IDF
0.938	0.940	0.938

Table 3.8: Models average accuracies over different weighting methods.

For cross-validation check, one setup for both algorithms that delivered the best accuracy value, is tested. The test is carried out in two stages. First both models are tested against the input dataset that is divided randomly into 10 equally sized partitions (folds). Each fold has 1093 or 1094 samples. After that 100 randomly selected samples are taken from the initial dataset that is divided into equally sized folds. Results from both tests can be seen in table 3.9that shows mean average accuracies, recalls, precisions and standard-deviation (σ) value for every investigated performance metric.

Samples	**Model**	**Accuracy**	**Acc.σ**	**Max. Acc.**	**Min. Acc.**	**Precision**	**Pre.σ**	**Recall**	**Rec.σ**
10939	SVM	0.939	0.006	0.952	0.931	0.911	0.019	0.843	0.020
10939	NN	0.952	0.006	0.964	0.946	0.920	0.022	0.888	0.020
100	SVM	0.860	0.117	1.000	0.700	0.935	0.106	0.722	0.344
100	NN	0.880	0.132	1.000	0.600	0.885	0.170	0.805	0.307

Table 3.9: Cross-validation mean results, where σ illustrates standard-deviation result over 10 data partition folds.

When cross validation test includes all rows from initial dataset, obtained results mirrors values received from earlier hyperparameter tuning test. Performance metrics are high and σ-values low. This concerns both models. When sample size drops to 100 and each fold has 10 rows, σ values rise and all performance metrics drop. Decreases are still fairly low which means that it should be possible to obtain desired accurate prediction results with both models when it is attached to a software robot. Still the final selected model is AML's Binary Neural Network model with tuned hyperparameter values.

How machine learning prediction accuracy will change over time after multiple re-trainings was left out from the master's thesis's scope. This decision was made due to the lack of ticket variation. It is suggested to add model monitoring as a part of the automation entity after the pilot project.

3.3 Web service

For the software robot to use the prediction model over the Internet as planned initially, it needs to be published as a web service. This could be built to run in own server, which was done during the earliest tests, but AML studio provides a feature designed for this purpose.

3.3.1 Prediction Web Services

In AML every created model can be set up as web service and the end-point can be called anywhere where the automation process will be running. This serverless platform as a service (PaaS) architecture leaves one thing less to worry for the company when there is no need to maintain the server that would only be running the web service. It also costs less than actual cloud computer, unless there is need for huge number of models. During the personal computer tests, a local web service that was able to call from the outside, was built for testing purposes. If in the future there is need for additional properties that AML can't provide, there exists already a tested solution that can be used as an initial step for next project.

Figure 3.10 shows what prediction model looks like in the AML studio. Blue rectangles reflect graphically the model input and output boxes. Between these there are the same steps as was done in the training phase, but blocks used for training have been changed to saved model, dictionary and scoring blocks. First the text is parsed (lowercase, remove digits, etc.), then the weighting matrix created by using the same dictionary as was used when the model was created, and finally weighted data vector fed to the model. For prediction is used the model ("Trained best model" in figure 3.10), that was trained and saved in training phase.

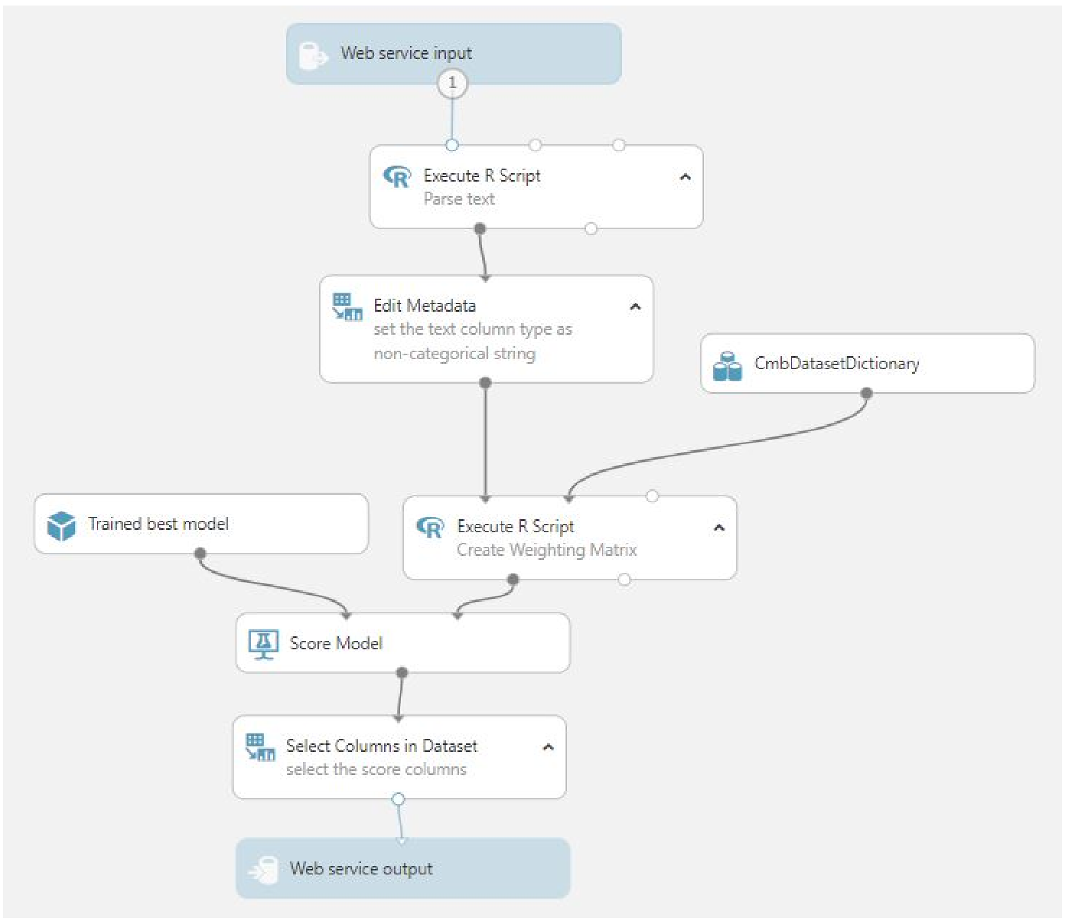


Figure 3.10: Example of what the created web service model could look like in Azure machine learning studio.

3.3.2 Continuous learning

The model was trained against the initial dataset, but for its prediction ability to last also with future messages, there might appear a need to re-train it from time to time with new data, when it loses its prediction accuracy against new tickets. This could be done manually by:

1. Appending the old dataset with new messages.
2. Labelling it with correct labels.
3. Importing dataset to AML studio.
4. Re-training the model with created prediction experiment.

5. Validating the prediction ability.

6. Overwriting the old model with new.

So that these wouldn't need to be done manually every time, an automation logic was created. There exist multiple ways on how the automated training could be done. In this master's thesis the first two steps will be implemented to software robot's workflow because the second step (correct labelling) could be marked after the software robot's actual work phase has been tried. By this implementation the robot would learn mostly from the correct answers and not from guesses that are based on human professional intuition. For other steps there is a possibility to create small individual program or write the needed blocks of code in software robot's automation workflow. For automating steps from 3. to 6. there will be first created small program for testing the solution. Later it can be included in the automation process' main workflow. Figure 3.11 shows a design of the whole continuous learning plan.

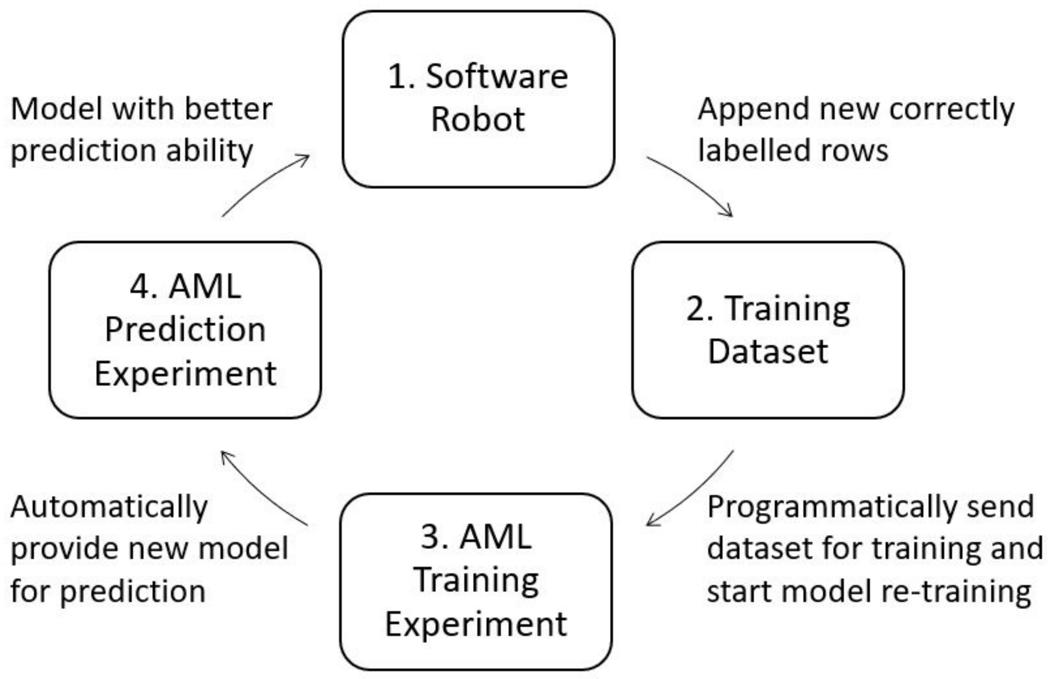

1. Software Robot
Append new correctly labelled rows
2. Training Dataset
Programmatically send dataset for training and start model re-training
3. AML Training Experiment
Automatically provide new model for prediction
4. AML Prediction Experiment
Model with better prediction ability

Figure 3.11: Outline for continuous model updating.

3.4 Software robot

In this section the logic and phases of the created RPA process are described. Going through all the steps in the built logic would be too overwhelming and that is why only the most important steps are opened a bit more. Figure 3.12 shows the main level steps of the created automation. Process logic follows the idea introduced in chapter 2.1.2.

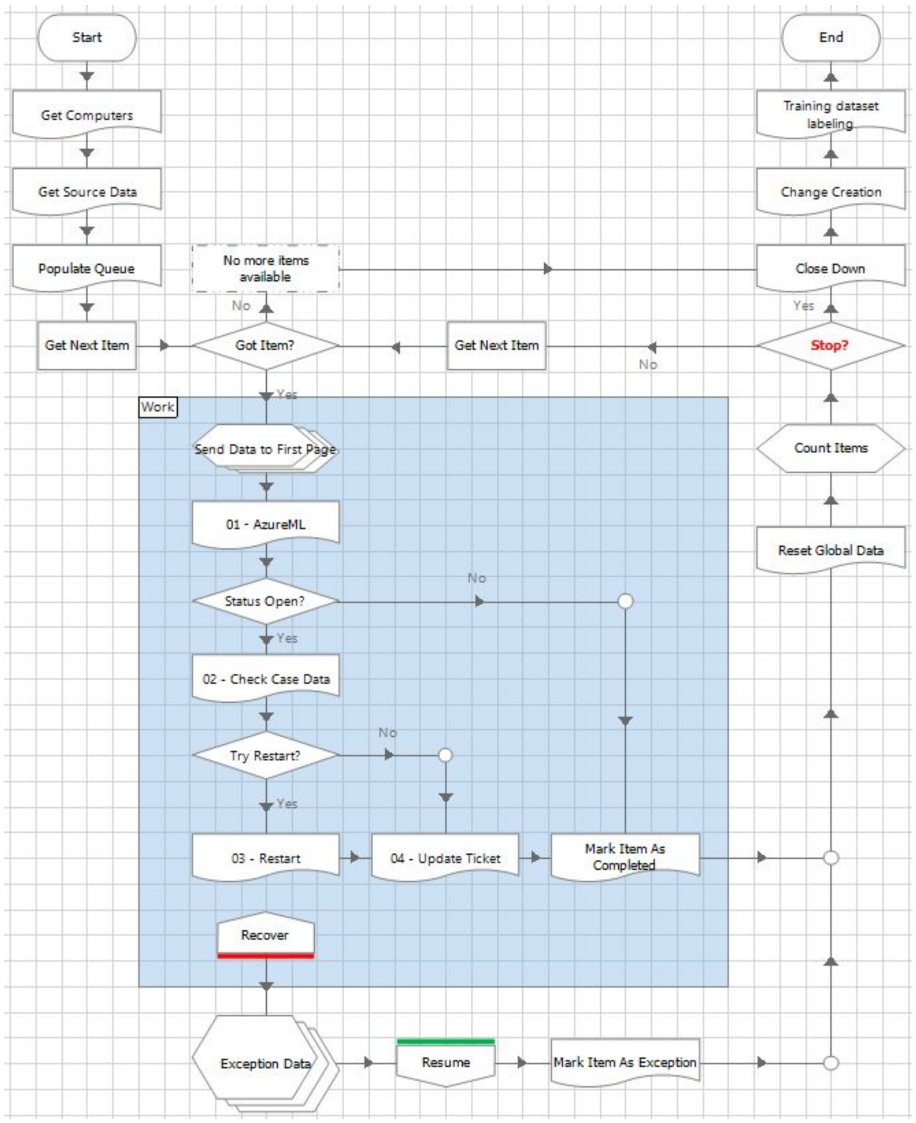


Figure 3.12: Main page of the created automation process.

The flow starts from the pictures top left corner's start-stage. First two tables are read to cache (Get Computers stage). First table consists of details (customer name, IP, server name) of all servers used as control machines (interactive client, IC) in customer environments. Second table has the same variables but from machines where the robotic processes run (resource machines). Data from tables is used for mapping information from tickets with machines so that in restart phase it is possible to connect to the correct IP. Next all open tickets from company's IT service management (ITSM) system are read. For this is utilized an application programming interface that the ITSM provides, but this could have been also implemented with normal software robot techniques by imitating the human way of working. API was chosen because it is faster and same created code blocks could be utilized later when calling machine learning model's API. Last stages in left branch before "Got Item?"-condition are automation platform required steps for adding tickets to an internal queue, where items are picked one by one in the process's loop section.

The loop exists between conditional stages "Got Item?" and "Stop?". Work phases are covered with blue block that defines recovery area. When planned or unplanned exception occurs inside this area, exceptional condition is caught in Recover-stage. Similar recovery handling logic is used in textual programming languages in shape of the "throw-catch"-structure. The loop branch begins by sending ticket message to machine learning model and getting an output from it. The output is in JSON format and consists of binary label value for question "could restart help" and normalized probability value for this. If the ticket's status is open, it is mapped to the ticket fields with the earlier received table values. In match cases a restart is tried. After that, restart or no restart, the ticket is updated in ITSM system by adding a tag for noticing employees that a restart has been tried. Simultaneously ticket status is also changed from open to pending.

After all received tickets have been handled, used systems are closed, possible changes investigated and tickets labelled with correct 0 or 1 value. In chapter 2.1.1 a logic for how a human solves a software robot's crash situation was introduced. From the same picture 3.13 is seen how the automation process logic or to its configuration needs to be changed if restart doesn't help for solving the incident situation.

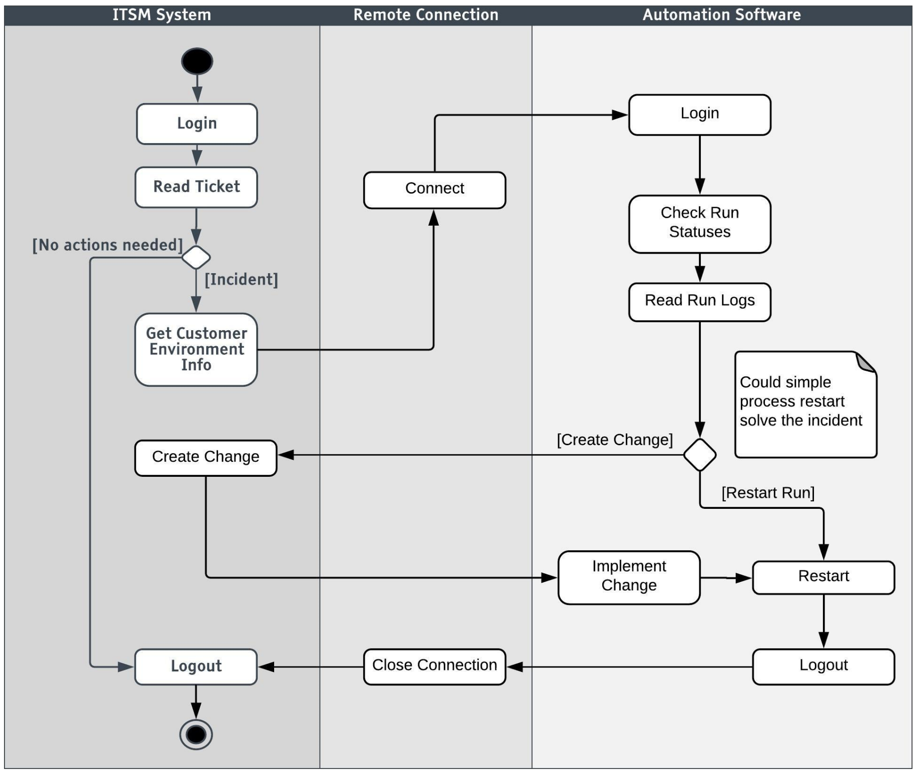


Figure 3.13: How a human check and solves the software robot's crash situation.

In this master's thesis the change creation is solved by tracking if an earlier automatically restart process terminates again or if the ticket queue reveals same termination messages over several days. In these cases, the ITSM's API is used for creating a change request for employees to solve. Correct ticket labelling is handled with similar logic. If new terminations don't appear for earlier restarted termination message the ticket is labelled as 1, otherwise as 0.

Continuous learning part wasn't attached in the main process workflow because the model training takes a lot of time compared to how long the automated process executes the given tasks. Keeping it as a separate part increases automation entity's modularity and enables quicker rotation time for the main automation. Continuous learning process can be therefore later scheduled to run at night-time, but during pilot project it was launched manually.

3.4.1 Results

In this section the statics that were obtained with created automation are explained. In the analysis, only the values that were believed to be beneficial for the company's business unit are investigated. Due to the short project time, the investigated statistics are not based on long term use. Considering this, the selected statistic values therefore are:

1. How long it takes for automation to do initial preparations.
2. How long it takes for automation to handle one ticket without restarting steps.
3. How long it takes for automation to handle one ticket with restart.
4. How long solving one ticket manually takes vs. how long it takes for the software robot.

The RPA process's initial preparations consist of two parts: reading the customer related data from an xml-file and getting open tickets from the company's ITSM software. The first step takes most part of the total time. During implementation it was noticed that the internal loop structures that automation software provides are slow, compared to the normal loop functionalities normal software languages provides. If the time consumed for initial preparations needs to be decreased later, an easy way is to create a custom small block of code using C-Sharp-language's loop structures for reading and picking need information from file. Total time for initial preparations is 21 seconds, from which the reading part is 19 seconds.

With the next metric it is measured how long it takes to get a response from Azure machine learning model and update the ticket without testing the terminated schedule restart. Over 95 percentage of all tickets drops to this category. Going through the main process's loop section thus takes approximately 7 seconds. If process restart is tested, item handling time raises to 46 seconds.

For the last investigated statistical value it is chosen to compare how long it takes for a human to solve one ticket compared to automation. Value for human solving time is based on laboratory condition approximation when one experienced employee does the same steps as the software robot would do. Received values varied from 3-5 minutes depending on the ticket complexity and if a change request to the ITSM system is needed. It is necessary to keep in mind that when these kinds of measurements are made, employees tend to focus more than in normal working situations and full theoretical potential of a full-time equivalent (FTE) can't be reached [36]. Organization for economic co-operation and development (OECD) provides a list for typical yearly FTE values for each of its member country. In 2017 number for average annual hours actually worked per worker in Finland was 1531 hours [37] while the theoretical maximum is 1950 hours (7.5 hours * 5 days * 52 weeks). When using the average value of 4 minutes for human ticket solving time and assuming that people spend 20 percentage of their working time at work to other than effective work, we get that human could solve 18372 tickets in a year. Yearly theoretical maximum run time for one software robot is 8736 hours (24 hours * 7 days * 52 weeks) which equals to 5.7 median Finnish workers. There are always problems with network and system update downtimes with computers, so it as estimated that in a year total downtime for running automated process is 2 percentage (173 hours, 7.2 days). Approximated value for how many tickets software robot could solve in a year is therefore 1116689 tickets. In order to give some sort of ratio for this value, let's imagine that the company's ITSM receives approximately 71 tickets per day. A little bit more than half of these are for the run management team and less than half of the cases are those where the restart could help. If calculated with 40 tickets per day for the RM unit, it would take 76.5 years to reach the approximated maximum value that the automation could handle in a one year with the current speed.

In summary, the obtained results are more than promising if just the numbers are looked. They are consistent with the results introduced in the introduction chapter about the results global companies have been reporting. The transaction volume, i.e. the amount of tickets, is just not large enough for obtaining the similar FTE and ROI results that those researches have. Even if some unknown issues would rise with the automated process, the differences compared to manual work are so significant that a couple percentage increase in the automation downtime still makes it profitable to utilize it. Machine learning model is only used for widening the RPA decision making and making it fuzzier. Calling the model inside the RPA process doesn't require significant extra time compared to normal RPA decision making. By combining machine learning decision making with RPA process it is possible to create RPA processes with wider scope. However, long-term results could not be collected during the master's thesis, so the results obtained must be treated with caution.

Chapter 4

Discussion

This chapter reviews what was done during the project and discusses about implementation limitations and future visions. Future visions are further research opportunities found during the master's thesis. They have not been investigated during the pilot project and are only based on professional expertise.

The goal of this master's thesis was to automate a part of robotic process automation maintenance unit's (Run Management) work by creating a solution that utilizes both RPA and machine learning. Nowadays this combination is called intelligent process automation (IPA). Project was divided into literature review and solution creation. The goal of the literature review was to get enough information to build a working solution. Automation solution part was handled as a pilot project from the perspective of the company for which this master's thesis was done. Because the company is a service provider for two process automation platforms it was clear that the created solution would be done using one of these. The company also provides cloud services utilizing Microsoft's Azure services so it would be good if opportunities provided by Azure services could be used in the created solution. Considering the above listed ideas, the implementation plan was divided under three modular parts: machine learning model, web service and software robot. This generic design is not bound to any technologies or to any precise service provider, instead the best solutions can be chosen.

The end result is a working modular entity that also uses tools that the company provides. By using Microsoft's Azure machine learning studio, the model and the web service parts were possible to combine under one logical entity. Obtained results are promising and with couple of tweaks it is possible to set the created solution to reduce the routine workload of the employees.

4.1 Limitations

During the project a couple of obstacles were encountered. In summary it can be said that the two major limiting things restricting the perfect possible solution where the difficulty to copy a human way of working and the IT security policies that needed to be followed.

Cybersecurity is increasingly present in everyday work [38] and one of the challenges it creates for the run management unit is the use of VPN. Some of the customers require that the unit uses the customer's own VPN solutions for accessing their environment, while others use the general solution provided by the company. For the connection to be created, after feeding the username and password the login needs to be verified with a token code that is sent to a pre-defined mobile phone. This functionality is called a two-factor authentication (2FA). Automating this is possible but it is hard, and it breaks every security rule and the reason for extra identification loses its value. The only way how this could be solved is to move the created automation solution to run inside the company's IT-service network, where it would have access to customer networks without the need for VPN and 2FA. This wasn't done during the master's thesis due to possible changes in company's security policies and it was left out of scope. In a current setup where the process is running on local machine and customer machines accessed outside from company's IT-network, the VPN connection needs to be manually established before the created automation solution can try to restart processes.

Second limiting IT-security policy issue is that the company isn't yet using any methods that enable one user account to have access to multiple customer environments. The most known service with which this could be done is Microsoft's Active Directory (AD) [39]. Currently there is a need to create local Windows user credentials for software robot user for every customer production- and interactive client (IC, control machine). Every customer has also at least two databases for creating new, testing updates and running existing automation solutions, each of which also needs access credentials. For example, if a company had 50 customers and each customer had 10 production machines to run existing processes, 2 IC machines to control the production and 1 database for production, the software robot would need to have 80 (50 production resources + 20 IC resources + 10 production databases) different username and password combinations. Passwords are also set to expire after 30 days and they need to be updated and when new machines are created, users need to be also created there. For employees working in the run management team this means additional tasks. The software robot on the other hand can be programmed to handle password changes automatically. In this master's thesis a table in XML-format that keeps every customer information needed for login to computer and customer database was created. Credential information is kept encrypted in a second place. The software robot was programmed to change the expired passwords automatically and the only manual task is to add details about new computers to the XML-list and to the credential vault.

To access the customer's IC machines to investigate the process runs through a graphical user interface (GUI), employees use Windows's remote desktop protocol (RDP) software. This complicates robotic process automation because only pictures from the remote display move over network and there is more delay. As introduced in section 2.1.2 there are basically two methods how the RPA process interacts with target systems: using software's internal structures for identifying window elements or searching the pictures from screen. The first method is easier to use and more reliable than the second one. Because only pictures are transferred over RDP and elements need to be recognized as pictures from display, interactions over remote connections were minimized so that the implemented process would be as resilience as possible against delays and unexpected future situations. Fortunately, the automation software's service provider offers an ability to control software through command-line, which humans typically never use. A solution for this problem was found by creating configurable executable .bat-files with the automation process. First, they were created to a local computer that calls the automation software with command-line functions. Then the software robot opens a RDP connection to the customer environment with a setting that enables local file access from remote computer. Then it executes the .bat-files there. Without applying this trick, the RPA methods would have been extremely time consuming and the end solution would have been sensitive to new environments.

4.2 Future visions

In this section is discussed about future visions. The review is done from two different perspectives of the company. In the first one is discussed about future visions that the company could utilize with their internal support functions. In the next is discussed about suggestions that could be used in business.

4.2.1 Support functions

During the master's thesis' pilot project, the company has started using Information Technology Infrastructure Library (ITIL) service methodology as a base of their continuous services. ITIL is a widely used set of best practices about how the organization should handle their whole continuous services [40]. In practice, it brings a lot of intermediate steps for daily work, which are irrelevant for the actual solution of solving the disturbance situation. These are mandatory for service measurement and traceability that enable continuous development and better overview for the whole service status. Figure 4.1 introduces ITIL service management flow and its processes [41]. It should be noted in the picture's left side that incident management is only a part of a wider entity and the major idea behind the concept is continuous process improvement.

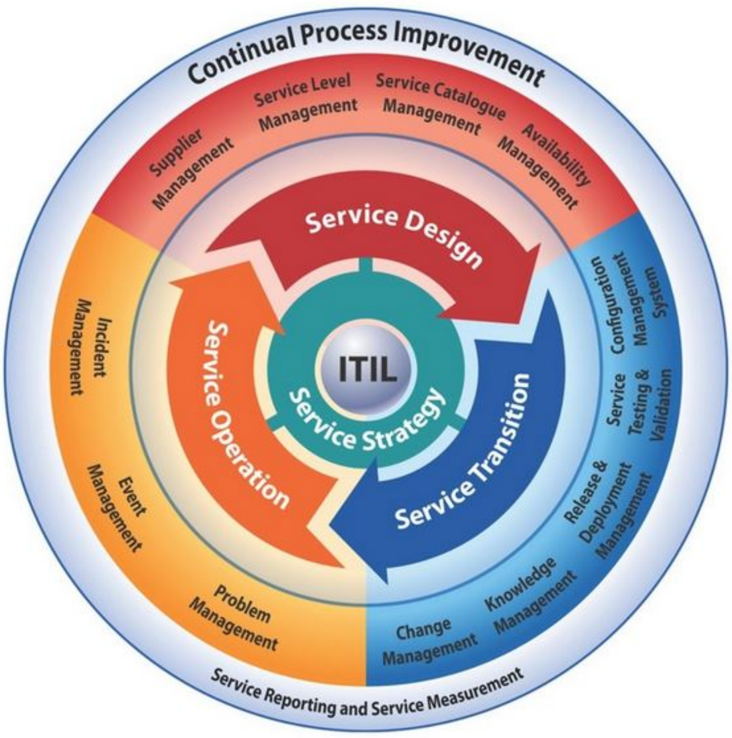


Figure 4.1: ITIL lifecycle and processes.

When humans need to work strictly according to pre-defined processes, they might forget or neglect on purpose some required intermediate steps, which weakens the accuracy of measurement data. It is an unending battle to build continuous monitoring and a companywide control mechanism so that bottlenecks are found and right repairing actions implemented. The technical ways found and used in this pilot-project could be used for ITIL process data gathering and for probability-based decision making. One of ITIL's processes is change management, for which a partial solution was already implemented during the project. Automation process' creates a change request for employees from re-occurring incident tickets. During the master's thesis there arose an idea about drawing all the company's ITIL processes in a similar way the run management's robot termination incident handling was done. By doing this, the process bottlenecks and places where the analysable data exist could be more easily found. The ITIL's reporting requirements for its processes could be automated because RPA solutions save logs and they can be automated to save data in the right format. Gathered data could be later utilized with machine learning algorithms and parts of more advanced ITIL processes could be automated. Idea of this can be seen in figure 4.2. The space between two lines illustrates what was done during the pilot project compared to whole ITIL pipeline. Every ITIL function includes large amount subprocesses which means a lot of potential automation applications.

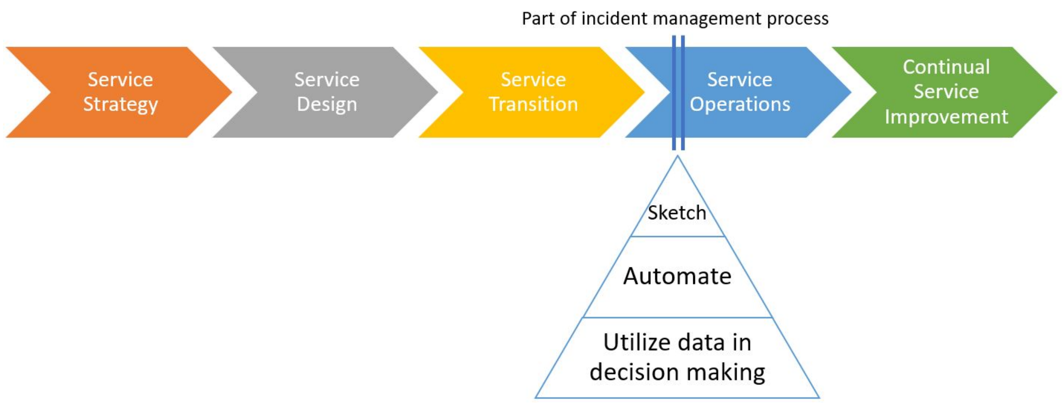

Part of incident management process
Service Strategy
Service Design
Service Transition
Service Operations
Continual Service Improvement
Sketch
Automate
Utilize data in decision making

Figure 4.2: The part of whole ITIL pipeline what was automated during the master's thesis.

4.2.2 Customers

The idea about utilizing the gathered data and logs can also be interesting from the company's business perspective. By integrating machine learning model as a part of an automated RPA process, the company could begin to offer wider and longer process automation solutions and faster lead times. Data moving through automated processes can be saved in correct format to a relational database. This can be seen as pre-built data pipeline for data analysing purposes. Some of the previously built processes already have saved data, which would make the data utilization even easier. By taking advantage of these possibilities, the company could start to sell improvement projects for old processes.

Other option could be to start integrating continuously learning models as a part of new automated processes. The benefit of such a solution would be revealed when enough data is gathered. A sketch of this design can be seen in figure 4.3. When a model starts to give accurate predictions, this part from automation process could be removed to achieve quicker lead times.

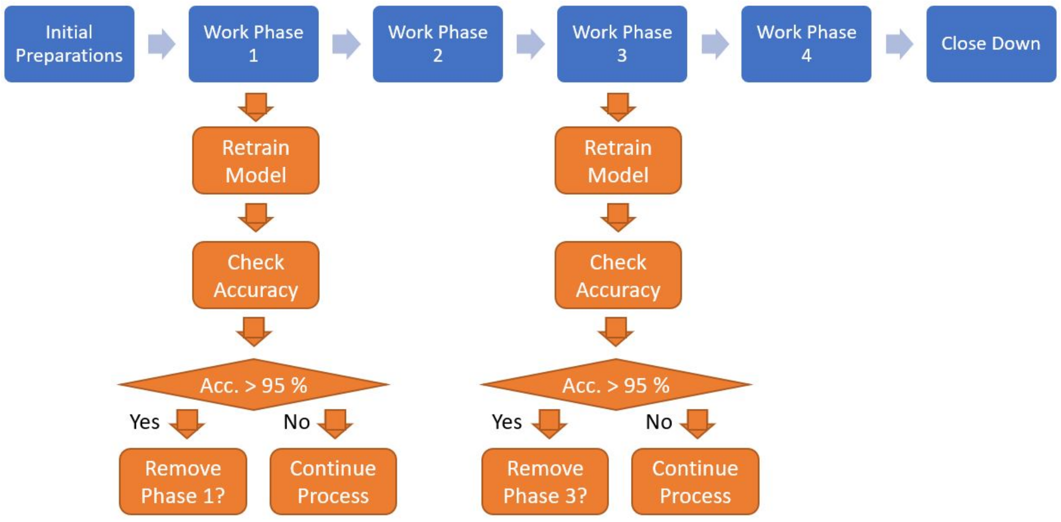


Figure 4.3: Vision about creating continuously learning models as part of RPA workflow.

In last scenario, whole automation software could be removed if there isn't any more need to check all business rule conditions. Instead the process could predict the result based only to input data which in practice would remove all latencies.

Chapter 5

Conclusions

As already stated in the chapter's 2.1 end, the biggest factor affecting to automation project's success is that suitable business processes are chosen and they are improved, so that unnecessary steps are cut out or modified. The part of ticket solving process that was automated in this master's thesis, didn't needed any improvements before automation and it was carefully selected. Still, the reader may wake up to a question like "Why software robot even need an external restart, shouldn't it be possible to add that kind of recovery logic to every process' logic?". There is no clear answer to this question. Every automation process is uniquely build based on customer's needs and requirements. Some processes are extremely business critical while for others it is enough that they work most of the time. Also process implementation quality varies based on the developer's level of experience. This can be seen especially when the field of business is relatively new. The created automation in this master's thesis can be seen as a back-up plan. That is, if despite all the process level verifications, something breaks, it is set back to run mode as soon as possible.

With the results obtained during the project is possible to answer to the initial problem statement question "Is it possible to obtain synergy benefits by integrating machine learning model as a part of RPA?". The answer for this is yes. Robotic process automation suffers from a problem where every decisions needs to be precisely predetermined. This means that a part of the automation potential cannot be realized. In cases where the process decisions can be based on probabilities without a high risk for business if the guess is false, machine learning is a viable alternative. Creating an accurate ML model also requires thousands of rows of valuable data, which is not usually available, at least when the project starts. The workload of creating a predictive model corresponds to automating a part of business process with RPA, but the time spent for work can be reduced by utilizing pre-created solutions that some vendors provide and by copying the solution created in this work. During the master's thesis creation, it was noticed that it is possible to obtain accurate predictions with a relatively small workload but getting exact values requires refining the model repeatedly. The combination has benefits with certain automation possibilities but those can be hard to find. A customer who is considering a combination and a supplier who is selling it, should know the benefits of both techniques and not just try to force it to fit somewhere where it can't bring any value.

Bibliography

[1] Adam Smith. *Wealth Of Nations*. 1776.

[2] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. 2012.

[3] James Manyika, et al. Jobs lost, jobs gained: Workforce transitions in a time of automation, December 2017. `https://www.mckinsey.com/global-themes/future-of-organizations-and-work/what-the-future-of-work-will-mean-for-jobs-skills-and-wages`.

[4] Wilhelm Hasselbring. Article: Information System Integration. `https://oceanrep.geomar.de/14608/1/CACM-ISI2000.pdf`. Accessed: 2019-03-20.

[5] Maureen Brown William Anderson. Revealing Cost Drivers for Systems Integration and Interoperability Through Q Methodology. `https://resources.sei.cmu.edu/asset_files/WhitePaper/2009_019_001_29033.pdf`. Accessed: 2019-03-20.

[6] Keith L. Murphy. Robotic Process Automation and Low-Code. `https://www.outsystems.com/blog/posts/robotic-process-automation-low-code/`. Accessed: 2019-03-20.

[7] Craig Le Clair, Alex Cullen, Madeline King. The RPA Market Will Reach $2.9 Billion By 2021, February 2017. `https://www.forrester.com/report/The+RPA+Market+Will+Reach+29+Billion+By+2021/-/E-RES137229`.

[8] Leslie Willcocks Mary Lacity and Andrew Craig. The Outsourcing Unit Working Research Paper Series: Paper 15/02: Robotic Process Automation at TelefonicaO2. `https://eprints.lse.ac.uk/64516/1/OUWRPS_15_02_published.pdf`, 2015. Accessed: 2019-03-22.

[9] Leslie Willcocks Mary Lacity and Andrew Craig. The Outsourcing Unit Working Research Paper Series: Paper 15/05: The IT Function and Robotic Process Automation. `https://eprints.lse.ac.uk/64516/1/OUWRPS_15_02_published.pdf`, 2015. Accessed: 2019-03-23.

[10] Oxford living dictionaries: Definition of ticket in English. `https://en.oxforddictionaries.com/definition/ticket`. Accessed: 2019-03-20.

[11] Michael Hammer and James Champy. *Reengineering the Corporation: A manifesto for business revolution.* 2002.

[12] John Jeston and Johan Ellis. *Business Process Management: Practical guidelines to successful implementations.* 2008.

[13] S. Liman Mansar H.A. Reijers. Best practices in business process redesign: an overview and qualitative evaluation of successful redesign heuristics. `https://www.sciencedirect.com/science/article/pii/S0305048304000854`, 2005.

[14] Matthias Kunze and Mathias Weske. *A Pragmatic Guide to Business Process Modelling, Second Edition.* 2016.

[15] Matthias Kunze and Mathias Weske. *Behavioural Models: From Modelling Finite Automata to Analysing Business Processes.* 2016.

[16] The Institute of Electrical and Electronics Engineers, Inc. IEEE Guide for Terms and Concepts in Intelligent Process Automation, October 2017.

[17] Matthias Kunze and Mathias Weske. *Service Automation: Robots and The Future of Work.* 2016.

[18] International Electrotechnical Commission. *International Standard IEC61131-3: Programmable Controllers - Part 3: Programming languages.* 2003.

[19] Santiago Aguirre and Alejandro Rodriguez. Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study, August 2017.

[20] Remote desktop protocol. `https://docs.microsoft.com/en-us/windows/desktop/termserv/remote-desktop-protocol`. Accessed: 2019-02-22.

[21] Sholom M. Weiss, Nitin Indurkhya and Tong Zhang. *Fundamentals of Predictive Text Mining, Second Edition.* 2015.

[22] Rafael E. Banchs. *Text Mining With Matlab.* 2013.

[23] Charu C. Aggarwal, ChengXiang Zhai. *Mining Text Data.* 2012.

[24] Roger Barga, Valentine Fontama, Wee-Hyong Tok. *Predictive Analytics with Microsoft Azure Machine Learning, Second Edition.* 2015.

[25] W3C: Web Services Glossary. `https://www.w3.org/TR/2004/NOTE-ws-gloss-20040211`. Accessed: 2019-03-20.

[26] Lewis Berman Bharat S. Rawal and Harold Ramcharan. Multi-Client/Multi-Server Split Architecture. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6496712`. Accessed: 2019-03-22.

[27] Chapter 1: Service Oriented Architecture (SOA). `https://web.archive.org/web/20160206132542/https://msdn.microsoft.com/en-us/library/bb833022.aspx`. Accessed: 2019-03-20.

[28] Hypertext Transfer Protocol – HTTP/1.1. `https://tools.ietf.org/html/rfc2616`. Accessed: 2019-03-22.

[29] R library tm. `https://cran.r-project.org/web/packages/tm/tm.pdf`. Accessed: 2018-03-14.

[30] Create R Model azure machine learning. `https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/create-r-model`. Accessed: 2018-08-29.

[31] Initialize Classification Models azure machine learning. `https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/machine-learning-initialize-model-classification`. Accessed: 2018-08-29.

[32] Tune Model Hyperparameters azure machine learning. `https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/tune-model-hyperparameters`. Accessed: 2018-08-29.

[33] Cross-Validate Model azure machine learning. `https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/cross-validate-model`. Accessed: 2018-08-29.

[34] Two Class Support Vector Machine azure machine learning. https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-support-vector-machine. Accessed: 2018-08-29.

[35] Two Class Neural Network azure machine learning. https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/two-class-neural-network. Accessed: 2018-08-29.

[36] Glossary: Full-time equivalent. https://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:Full-time_equivalent_(FTE). Accessed: 2019-02-21.

[37] Average annual hours actually worked per worker. https://stats.oecd.org/index.aspx?DataSetCode=ANHRS. Accessed: 2019-02-21.

[38] Making a secure transition to the public cloud. https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/making-a-secure-transition-to-the-public-cloud. Accessed: 2019-02-22.

[39] Active directory domain services. https://docs.microsoft.com/en-us/windows/desktop/AD/active-directory-domain-services. Accessed: 2019-02-22.

[40] Claire Agutter, Alison Carlidge, Ashley Hanna, Stuart Rance, Colin Rudd, John A Sowerby, John Windebank. *ITIL Foundation Handbook*. 2012.

[41] The ITIL Service Lifecycle. https://www.quanta.co.uk/blog/2012/04/itil-service-lifecycle. Accessed: 2019-03-18.