Machine learning approach to support ticket forecasting from software logs

Matti Haukilintu

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 10.08.2022

Supervisor

Prof. Arto Visala

Advisor

MSc Petri Pyöriä

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Matti Haukilintu

Title Machine learning approach to support ticket forecasting from software logs

Degree programme Automation and electrical engineering

Major Control, Robotics and Autonomous Systems **Code of major** ELEC3025

Supervisor Prof. Arto Visala

Advisor MSc Petri Pyöriä

Date 10.08.2022 **Number of pages** 36+1 **Language** English

Abstract

Your abstract in English. Keep the abstract short. The abstract explains your research topic, the methods you have used, and the results you obtained. In the PDF/A format of this thesis, in addition to the abstract page, the abstract text is written into the pdf file's metadata. Write here the text that goes into the metadata. The metadata cannot contain special characters, linebreak or paragraph break characters, so these must not be used here. If your abstract does not contain special characters and it does not require paragraphs, you may take advantage of the abstracttext macro (see the comment below). Otherwise, the metadata abstract text must be identical to the text on the abstract page.

Keywords Machine learning algorithms, Robotic process automation, log analyzing, random delay, Microsoft Azure ML Studio

Aalto University
School of Electrical
Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Matti Haukilintu

Työn nimi Sovelluslokien ja vikatikettien yhteyden löytäminen koneoppimista hyödyntäen

Koulutusohjelma Automaatio- ja sähkötekniikka

Pääaine Ohjaus, robotiikka ja autonomiset järjestelmät **Pääaineen koodi** ELEC3025

Työn valvoja Prof. Arto Visala

Työn ohjaaja FM Petri Pyöriä

Päivämäärä 10.08.2022 **Sivumäärä** 36+1 **Kieli** Englanti

Tiivistelmä

Tiivistelmässä on lyhyt selvitys kirjoituksen tärkeimmästä sisällöstä: mitä ja miten on tutkittu, sekä mitä tuloksia on saatu.

Avainsanat Koneoppiminen, koneoppimisalgoritmit, ohjelmistorobotiikka, loki, lokin analysointi, satunnainen viive, satunnaisviive, Microsoft Azure ML Studio

Preface

Terve,
ja kiitos kaloista.

Otaniemi, July 7, 2022

Matti Haukilintu

Contents

Symbols and abbreviations

Symbols

$\mathbf{P}$	placeholder-symbol
$\mathbf{A}$	another placeholder-symbol

Operators

$\nabla \times \mathbf{A}$	curl of vectoring $\mathbf{A}$
$+$	yep, a plus

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
HML	Hybrid Machine Learning
RPA	Robotic Process Automation
SQL	Structured Query Language
JSON	JavaScript Object Notation
CSV	Comma-Separated Values
GDPR	General Data Protection Regulation
ADA	Anomaly Detection Algorithm

1 Introduction

Artificial intelligence (AI) and machine learning (ML) has found their way into more and more fields of business. In banking business they are already used in fraud detection, risk management and service recommendations.[1] Even though these modern big data utilizing technologies are widely used abroad, in Finnish banking field AI and ML are not popularly utilized. Instead, many self-acting solutions are being used to streamline manual labor which could be called intelligent, but are merely highly automated processes and thus cannot be included in the AI category. One of these technologies used in Finnish banking systems is Robotic Process Automation (RPA).

RPA operates "on the user interface of other computer systems in the way a human would do",[2] but is strictly bounded by predefined operations thus being prone to unforeseen situations such as faulty input. RPA, like generally all other software, produces log to "register the automatically produced and time-stamped documentation of events, behaviors and conditions relevant to a particular system"[3]. Logs don't have any standards or form guidelines to follow which tends to make log analysis and log based problem-solving troublesome. This is also the case with RPA's developed by Oy Samlink Ab.

Oy Samlink Ab (Samlink from now on) was founded in 1994 and is now owned by Kyndryl. From the early years while going by the name of Samcom the company was owned by several Finnish banks developing all sorts IT solutions for them. Nowadays, Samlink offers a wide variety of banking solutions from basic banking system to end user targeted software such as Codeapp-mobile application.

Besides banking, Samlink develops multiple other IT solutions to extensive range of customers, for example entertainment platform solutions for DNA. Even though Samlink can be considered a modern technology company, the most modern AI technologies has not yet been adopted in the variety of tools used in development. However, RPA has been actively used in some banking solutions to reduce the amount of manual labor required from banking clerks.

In addition to continuous development as well as product maintenance services, Samlink also offers a technical help desk regarding the software solutions produced. As no IT solutions comes without bugs or misbehaviour, Samlink service desk has to use considerable amount of labor to resolve the possible reason behind the technical support request tickets received. In many cases, the problem-solving starts by reading the log and analyzing the data written by processes in question.

In this study, we aim to find if it is possible to utilize machine learning methods in analyzing logs created by Samlink RPA's. Ultimately, we intend to train ML which is able to predict the arrival of a technical support ticket thus giving a warning for developers about possible issues in the production.

1.1 Background and motivation

In the field of information technology logging is one of the most important methods in problem-solving, be it software or operating system related.[3] Typically, at least

in Samlink processes, logging is a bit more verbose than it needs to be. This is usually because when the problem occurs it is easier to already have the verbose logs available than trying to replicate the issue after setting logging to more verbose mode. Too verbose logging, however, leads into two problematic issues for developers.

First of, the size of log is huge and finding the critical parts related to the problem in hand takes more time. Of course, with more strict logging pinpointing the issue from within the logs would be faster, but then again, solving the problem with only critical error messages could be more time-consuming if crucial context is missing.

Secondly, a well-designed software is able to retry the process after first failure, but logging is done in real time, not after final results of the process has been determined. This means, that each process failure is logged even though said process eventually succeeds. Thus, logs may include dozens of rows of information about a problem encountered, which are not critical information after all. These issues make log analyzing considerably laborious.

Production logs are usually not viewed if everything is presumably working as intended. Technical support tickets are both last and most visible indicator that something is wrong. When a technical support ticket is received from banking clerks it means that something is wrong in a very visible way. Roughly speaking, technical tickets that are due to clear misbehavior of the RPA systems and not, for example, user errors, can be divided into two categories. First are the tickets that uncover an unknown bug in the system which can be either fixed or instructed to user how to avoid. Second type of tickets are somewhat pre-known issues that occur from time to time and are either fixed with updating parts of the system or by rebooting the process.

Typically, in software systems, if issue is known and can be fixed by rebooting something, developers can create log monitors that search for certain keywords and raise an alert if encountered. Developers can either run a reboot manually after a log alert has been received or set up an automated script to do it immediately when such keyword has been found. However, when it comes to RPA's and technical tickets considering them, it is hard to say what type of issue is in question by reading the RPA logs word by word without context. New kind of issues can be more frequent than already confronted ones, and clear keyword linked to a certain problem may not exist without considerable amount of false positive matches.

Machine learning algorithms are widely used to find patterns from massive amount of data making it an ideal tool for log analyzing. Patterns, however, need a connection to a visible issue to be useful. If RPA system has encountered an error but is able to retry successfully, then no issue has practically happened that needs immediate concern. Hence, RPA log analyzing with ML can find meaningful patterns only if they relate to actual technical tickets.

If Samlink support has received a help request the issue behind the request is not fresh anymore. In the event of RPA job failing, it takes some time for the clerk to notice the issue, write a help request to first support level, which then redirects the ticket to the corresponding team. Furthermore, if the issue is noticed during friday, it takes few more days to be handled by RPA developers due to weekend. This leads to noticeable delay in processes that were supposed to be dealt by RPA but which

now have to be manually taken care of by said clerks.

If a correlation between logs and tickets received exists and an ML algorithm is able to find it, it could be possible to create an ML-based log analyzer that can send an alert to developers about an ongoing issue before banking clerks encounter it. With automated scripts set up to receive such alerts, some issues could even be fixed in the production automatically without human interaction. This would reduce significantly the time and labour needed from developers and bank clerks alike.

Comment: This chapter is good to go!

1.2 Research objectives

This research aims to pave the way for machine learning application developers inside Samlink. Multiple obstacles need to be tackled as most of the phases in this study has not yet been encountered inside the company.

First and foremost, it is crucial to construct some basic rules considering the format of log data to make it usable by ML algorithms. Log data formatting is one of the key elements in automatic log analyzing applications as it is not for just machines but also for people to read.

As today more and more concern is set on anonymization not only due to GDPR, the data used for machine learning must be sanitized. Because of this, one major objective is to create a clean dataset that is safe to use in cloud environment without raising concern around security and privacy issues. In addition to this, data must also be clean enough so that ML algorithms are able to process it.

As mentioned, Samlink has not yet developed ML applications. In order to ease the for future ML application developers, this study aims to document the process well enough to create a simple guide to follow in the possible future ML projects in Samlink.

Finally, the main question this research is aiming to answer is: *is there such a correlation between RPA run logs and technical support tickets that ML algorithm is able to find it, and can this correlation be used to forecast a ticket arrival*?

1.3 Scope

In order to limit the study to feasible length and content, it is necessary to define the scope for the thesis. Before diving in to the scope of research objectives, we must first make one assumption regarding the data that ML is going to find some meaning from. In order to find a connection between log data and support tickets, we make a hypothesis that errors leading to tickets are visible to or parseable by ML algorithm. In addition, as we are going to utilize anomaly detection algorithm in log analyzing, we must also assume that these errors in log are, in fact, anomalies. We will, however, compare the results against these assumptions to test this hypothesis.

Data anonymization

Anonymization in the context of this thesis refers to data sanitization process purposed to edit the data into more secure form in the privacy point of view. In this study we aim to create a dataset usable in ML training. In this respect, anonymization is not in the main focus of the study but only treated as a sub-phase of the data preprocessing in whole. Nevertheless, anonymization is from the privacy perspective the most important phase of data preprocessing.

Keeping this in mind, anonymization is covered rather superficially, only enough to explain the reasons behind actions taken during anonymization process.

Azure setup

The ML training and result scoring is done in Azure ML environment. Azure is used because...

As one of the objectives is to create an initial guideline for ML process commissioning the Azure setup phase is documented in such detail reflecting the importance of this information for future developers starting Azure ML projects in Samlink.

Data requirements

Data purity in a sense of how easy it is to be used by ML algorithms creates challenges at the beginning of ML training. If data is not consistent, has lots of missing values or is formed in unanticipated way, it requires considerable amount of preprocessing slowing the training process and causing errors in pipeline runs.

In order to create a baseline for Samlink ML projects this study aims to give basic criterion what is required from the data, so it is easily analyzable by ML algorithms.

Machine learning methods

Several different machine learning algorithms exist that are aimed for different applications in mind. For example, to make an algorithm that can predict the price of an apartment listed[4] we could be using linear regression, and in order to detect possible cyber threats from network traffic[5] a two-class support vector machine could be utilized. These two methods are very different in usage and has their pros and cons in different applications.

As different methods can be used in creative ways in very different applications depending on how the data is presented and how the ML problem is formed, this study focuses on just a few easily approachable training methods that were seen suitable to answer the study objectives.

When it comes to anomaly detection algorithms (ADA), only principal component analysis (PCA) is considered because PCA-based Anomaly Detection component is the only one usable from existing two anomaly detection algorithms in Azure ML Studio. As purely Azure ML Studio is used during this study, no other anomaly detection algorithms are debated. The other ADA-component, One-Class Support Vector Machine, is discussed briefly to explain its unsuitableness for current case.

1.4 Structure

In the **Introduction** section we explained the research motivation, main objectives and study scope. The next section, **Background**, explains the general machine learning concepts and terms relevant to this study case. We also discuss these topics from the perspective of existing studies. The third section, **Research material and methods**, explains in detail the data format and contents as well as the steps used in each... **Results** section reveals how the selected algorithms performed and how well the research questions could be answered Finally, in the section **Summary**, we summarize the research outcomes, evaluate the results, and discuss what could have been done better.

2 Background

Machine learning, or ML, is a subcategory of the AI field and data science. Typically, ML refers to a set of technologies used to "build computers that improve automatically through experience".[6] This is generally considered a machine way to simulate human learning process. ML usage has become more common and is nowadays widely used in many fields, not just in general information technology and computer science. This is because data can be gathered from anywhere, and where there is data to be processed, ML can be there to process it. Computer algorithms are able to find statistical correlation and patterns from places overlooked by human mind, or where amount of data is just too much for people to process. This is why ML has proved its power in various empirical science fields, such as biology, cosmology or social science.[6]

In this section, key concepts of ML are explained briefly and several ML features are explored that are most relevant to this study. We also discuss shortly about data sensitivity and how it had to be addressed during this study.

2.1 Machine learning algorithms and training

Algorithm means a finite sequence of (typically) mathematical operations that are used to solve a specific problem, typically by repetition of some steps until the problem resolves.[7] Algorithms are the main component inside ML. By iterating through all the data points algorithm is able to, for example, find repeating patterns, mathematical or logical connections, or unusual anomalies that would be seemingly normal for human eye.

Algorithms operate on set of rules that are tunable parameters. In order to utilize an algorithm to solve a problem, algorithm is first trained by tuning these parameters. Usually, ML algorithms can be trained in three ways: supervised, unsupervised, and reinforced learning.[6] Even more training methods exist that usually combine those mentioned.[8, 9] For the sake of simplicity, we focus on those three main methods.

In **supervised learning**, algorithm is given data with ready answers on how the data needs to be interpreted. Algorithm then tries to figure out the rules behind how given data and the correct answers are related.[8]

In **unsupervised learning**, on the other hand, algorithm does not get model data from which to train itself, but instead it tries to find clusters or groups inside the data that are linked together more closely than to other data points.

Reinforced learning refers to a method where a computer program is given a goal and provided feedback as a reward. This reward is what program aims to maximize by adjusting given parameters.

In ML, there are multiple algorithms to solve different problems and no jack-of-all-trades algorithm exists. Each algorithm is suitable for certain type of problem. To simplify, algorithms are usually divided into three categories based on the problem type.

Regression algorithms predict values and are typically used with supervised learning.

TODO: Examples: House market price?

Classification algorithms predict categories. Depending on the algorithm, they can predict between two or several categories.

TODO: Examples?

Clustering algorithms use unsupervised learning to find structures inside data.

TODO: Examples?

Anomaly detection algorithms work also unsupervised and try to find unusual or rare data points from data.

TODO: Examples?

Typically, when training an algorithm, some predefined portion of the data is used as training data.

TODO: Sources? Amounts?

The rest is used to validate the results so that validation data and training data do not overlap. Instead, trained algorithm is given data it has not seen before and the result it produces with it is then validated. For example, in supervised learning the key values the algorithm is trained to find out are hidden from the validation data. The resulting values produced by the algorithm are compared to those hidden values and the difference between the estimate and the real value can be used to determine how well the current trained algorithm compares to others.

TODO: Next one might change a bit depending on the actual results, which are still to be tested.

However, in this study, we are going to break that rule about non-overlapping training and validation data. The reason for this is explained further in section 3.6)

TODO: Something more generally about algorithms... Intro to next section (training).

The full component chain from input to output with algorithm training and result validating is called a machine learning pipeline.

2.2 Cloud ML platforms

TODO: Briefly about Azure, Google and AWS

Machine learning algorithms are not light to operate. Depending on the amount of data, it can be a serious

TODO: something something seriously big

Especially with online applications where real time analysis of new input data is required, cloud computing resources can make a huge difference in terms of processing speed.

TODO: referencing

Online market offers several solutions for ML computing in cloud.

TODO: open up some facts about these. BRIEFLY

Google
Amazon AWS(?)
Microsoft Azure[10]

2.3 Azure ML Studio

TODO: More info about Azure ML studio

Comment: some unorganized text:

Microsoft Azure offers a Machine Learning Studio environment for easy ML pipeline designing. ML Studio gives machine learning designer a possibility to train algorithms and publish cloud endpoints utilizing all Azure resources connecting the power of ML to all other Azure features like data storages, IoT-services and cloud computing.

With drag-and-drop pipeline designer it is easy to get started with ML programming, and visualizing the process helps understand all pipeline components and their relations to each other.

TODO: picture of azure pipeline

TODO: Intro to next subsections

Each component in pipeline can be tuned to a certain extent. ML Studio has a predefined set of ready algorithms to use. In this study we focus on R-script execution component, regression algorithm components, PCA-based anomaly detection component, N-Gram feature extraction component and feature hashing component.

2.4 Regression algorithm

TODO: science and math behind regression algorithm

Regression analysis is typical approach in statistical science. It is used to find relationships with a set of variables.

2.5 PCA-based anomaly detection

TODO: Explain PCA and mention other ADA algorithms

Comment: unorganized text below:

Principal Component Analysis, or PCA, is a machine learning technique used to analyze data and explain the variance inside it.

Other anomaly detection methods exist, but they are not supported by ML Studio in a ready component level.

TODO: Something about Anomaly and Novelty detection differences?

TODO: placeholder picture. replace with mathematical explanation

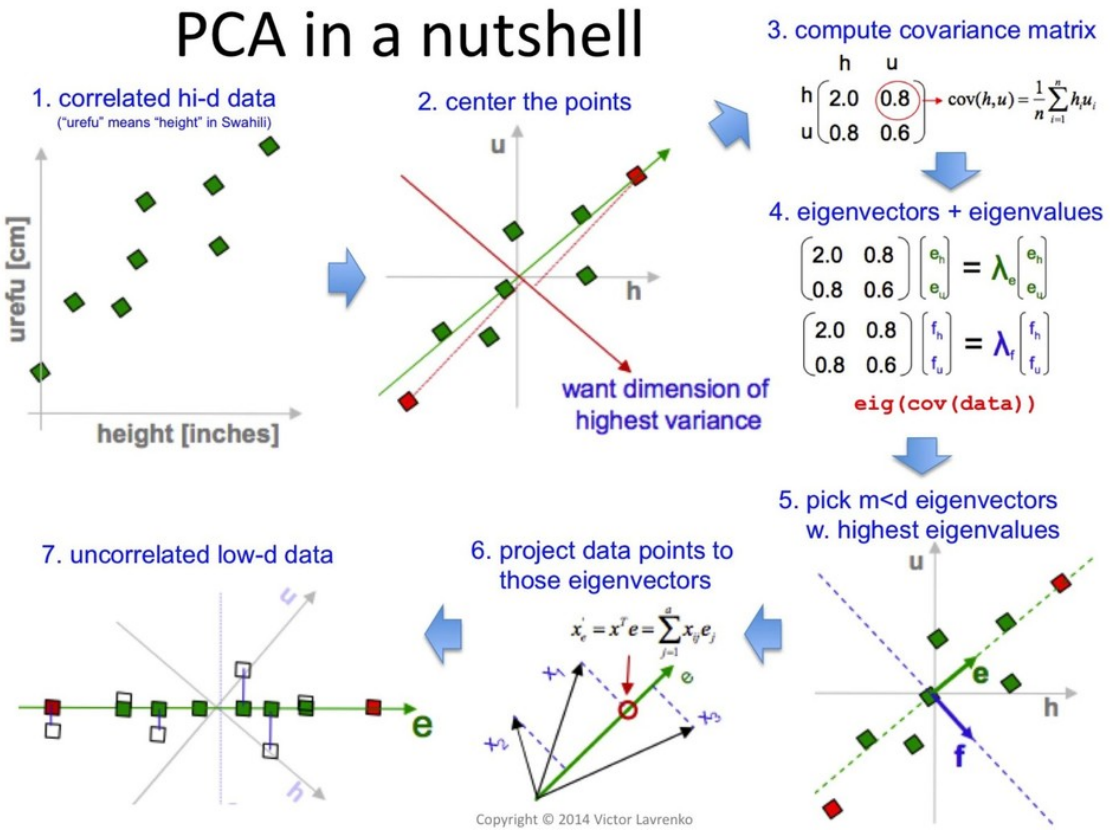

Figure 1: PCA in a nutshell

One-Class support vector machine

Azure ML Studio has also another anomaly detection algorithm to use. This module is called One-Class Support Vector Machine. In our case, however, this module was not deemed suitable as the documentation mentioned that "The dataset that you use for training can contain all or mostly normal cases." Because the content of the data used did not meet this requirement, the usage of this component was decided to skip.

2.6 N-gram features and feature hashing

TODO: cover basic n-gram features and ML connection

Comment: unorganized text below:

As stated before, features are the key elements in ML algorithm training. As textual input does not have any meaning to machines as itself, it is necessary to create a connection between words and features for algorithm. In ML training, one typical approach is to convert textual input to numerical features. For example, by creating a dictionary of words used in the input and assigning each word an identification number, we can express words as a count of certain words used. In addition, as words include meanings not only individually but also with relation to each other and in their order, we can add more information for the algorithm by creating word pairs and groups in the dictionary. These groups are referred as word grams, where **n** in n-gram refers to the maximum number of words in a group of consecutive words in the input sentence.

TODO: explain feature hashing component functionality briefly

As the number of word grams in a dictionary can increase significantly in complex input cases, it is necessary to limit the resource usage by decreasing the features analyzed. One way to do this is use feature hashing. This means that instead of pure n-gram count we use hashed value of several n-grams thus reducing the amount of features. As a drawback, the amount of information might also get reduced as the data is "compressed" but this way we can include more features for algorithm training without significant resource demands.

2.7 Data sensitivity

TODO: some basic stuff about anonymization, data sensitivity and data protection

TODO: Just practicalities, not much about theory or background

During this study, it was necessary to make sure no sensitive data was moved out of the production environment. This was mostly due to regulations described in GDPR.

Data anonymization was executed in production environment with PowerShell script. Several predefined identification features were searched with regular expression (or regex), patterns and replaced with default keys.

Comment:

Here we could explain more about pseudonymization and k-anonymization, but is it necessary as they were not used or considered? It would bring something more to the study, of course, but is it worth the time?

2.8 Log data analyzing with ML

TODO: Some research should exist

TODO: In this section, we look what studies and cases of ML log data analyzing exists in the IT field. Only briefly, nothing too deep.

2.9 Random delay in log event analyzing

TODO: Anything about the topic?

Random delay in input data features is not unusual aspect in time-series forecasting. Time-series in ML context refers to data features that varies over time and is usually affected by past values. As an example, ML algorithm could try to predict future weather based on measured temperature and air pressure. Both these features change over time and also affect to their own future values.

Random delay in such an example could be due to some other local or global features like

This study, however, is not time-series because the majority of the log rows are not affected by previously logged events. Random delay in this case is caused by banking clerks finding the issue and writing a technical support request. This delay can span from hours to several days depending on the weekday and time the issue occurred.

As random delay of such does not seem to be trivial to take into account with ML algorithms, a simple method to solve this was used which we call "time frame compression method". More about this approach is discussed in the section 3.6

2.10 Hybrid machine learning approach in anomaly detection

Hybrid machine learning (HML) refers to an ML technique where two or more ML methods are combined to overcome the limitations of or to boost the estimation capabilities of a single method alone.[11] In this study, we combine PCA-based anomaly detection algorithm with regression algorithm in order to amplify the prediction powers of our ML algorithm when trying to determine the possible ticket count based on log events.

Hybrid machine learning is not rare technique in ML field.[12, 13, 14, 15, 16, 17, 18, 19]

TODO: Something about the HML in ADA

In order to clarify whether hybrid approach is suitable for the current study problem we will compare the results of hybrid ML technique with a single ML algorithm usage.

TODO: Include wireframe model about hybrid model

With ML algorithm utilizing n-gram features combined with time frame compression it is possible to get estimates about the support tickets based on the log events. It is not feasible to use anomaly detection on its own to do this as plain sum of anomalies detected is not correlating with tickets received.

We can, however, amplify our ticket estimating algorithm with anomaly value features. As we first count the anomaly numbers with anomaly detection algorithm and their calculated statistical features with another algorithm, like regression algorithm, we get more relative information to use when creating the final ticket number estimations.

Comment:

Hybrid ML as a term is used here to explain that we use two different ML algorithms in two separate phases. In first phase we try to give an anomaly certainty value for each log row using PCA-based anomaly detection component. In second phase we use this value as a feature to estimate ticket amount in time range by utilizing regression algorithm.

Dual algorithm approach should not be unusual in ML field, but how existing studies or case examples relate to our way is uncertain.

If nothing exists about the topic (at least nothing easily to be found) it should be worth to mention. But if there is a lot of case examples about this, it feels unnecessary to discus about it in more detail.

3 Research material and methods

In the next section we explain in more detail what the data used in the study consists of and what methods were used in attempt to answer the research goals. The content of the section is briefly described below, and the steps of the research are explained.

The data in the research is mainly made of two parts. The most important part is, obviously, the log data produced by the numerous RPA processes. The second part complementing the study is the support ticket data written by clerks of customer banks. In order to use the data safely in the cloud environment it was necessary to sanitize the data from any sensitive information. This was done by anonymizing the log data and using only timestamps from the support tickets.

After confirming the results of anonymization, the data was preprocessed into a better form to make it more usable by algorithms. More processing was done inside the pipeline as ML Studio offered several usable components for this but main cleaning was easier and to execute in local environment. This was also done with PowerShell scripting.

ML pipeline was created in Azure ML Studio and several ML algorithms were compared in order to find the most feasible for our goal in mind. ML training was organized in two different phases in order to find the relation between log anomalies and technical tickets.

Finally, the results of the trained algorithms were validated against newly acquired production data in order to estimate how well the initial goals of the study were fulfilled. These results are presented in the next section 4.

3.1 Support ticket data

Like all other software, RPA components fail from time to time. As described before, RPA logs are verbose making possible error identification from among them hard. Due to that, it is not feasible to create log parsers that would be able to identify critical errors from within thousands of lines of log. When critical error happens causing the RPA process to fail, the banking clerks need to finish manually the job left by the RPA robot. Every time this happens, these clerks then send a support request ticket to Samlink technical help desk and ask to fix the issue.

When clerks send the ticket to technical support a verbose description of the situation is written to help developers to identify the problem. This description often contains sensitive end customer information like bank account details and social security numbers. To avoid privacy issues when processing this data, it was decided to use only timestamps of the tickets. The resulting data was practically a list of date and time values. More about the issue from privacy point of view is described in section 3.3.

3.2 RPA log data

Robotic process algorithms used in Samlink are designed to ease the workload of bank clerks. RPA robots work in behalf of bank clerks executing routine tasks that

require mostly manual labor.

Like other software, RPA also produces log data during runtime. As dozens of RPA robots are running in several bank environment the amount of log produced is also significant, up to over a million lines per week. This log data is not in consistent structure being formed out of typical CSV data injected with even more inconsistent JSON data that varies in contents vastly.

`TODO: refer to appendices, include examples of data`

RPA log data is stored in SQL database. The database is split in live production log that is gathered for few months and then moved to archive that has several years worth of log.

`TODO: check live timescale`

In this study we used archived data as it was easier to acquire in one run without the need to merge different parts together. Archive also had data that was considered as sufficient amount for machine learning algorithm training.

`TODO: how much really?`

Comment:

Next up we'll explain some key features about the log data such as timestamps, robot names, messages etc.

The important part to keep in mind here is that message is in two forms: message and rawmessage. Without rawmessage, the data was in pure CSV-format. However, it was not certain that rawmessage would not hold usable data for ML . In the end, the usage of rawmessage was mostly skipped as it demanded too much resources in ML Studio to process. In the data preprocessing phase the rawmessage was kept along and it lead to some interesting and most likely reusable preprocessing scripts.

Samlink RPA logs have few default features. These are listed in a table below.

The most notable aspect of these features are the *message* and *rawmessage*. *Message* holds the <varsinainen> information written by the RPA agent during automation execution. This includes details about the issue, what part of the process failed, and possible stack trace of the error. *Rawmessage* is JSON-formatted representation of all the default features, including the message and multiple other additional features that RPA agent is able to output regarding the execution. These additional fields are what vary from row to row. Some of the possible fields are listed in a table below, but several other field types exist.

3.3 Data anonymization

Support ticket data privacy

Samlink handles highly sensitive banking customer data in its processes, such as personal identification numbers, home addresses, email addresses and bank account

numbers. All possibly sensitive data had to be removed before data could be transferred out from production environment to cloud. Due to bureaucratic reasons, technical support tickets were under more strict policies. Because of this, they were allowed to be used in the research on condition that no business critical nor customer sensitive information was processed in the first place. Only way to assure this was to select solely timestamp fields from ticket data. Thus, no sanitation for ticket data was needed as ticket data consisted of only list of datetime values.

RPA log data sanitization

Information privacy is one of the key values in Samlink business promise as company develops high security banking applications and processes sensitive customer data. Thus, several aspects were needed to take into consideration before log data could be authorized for thesis study usage. To improve privacy, it was decided to assume that personal customer details are not critical information for ML algorithm training if goal is to find possible problems in RPA runtime and not detect individual customer related problems. This way it was not necessary to achieve adequate security by less secure and more effort consuming ways such as pseudonymization or k-anonymization (explained in the section 2.7), which would have also required strict inspections before data could have been approved for cloud processing.

`TODO: References?`

As production environment is built on Microsoft Server based solution, and because it was highly unrecommended to install additional software to the production server, data acquiring and anonymization tools were chosen based on what was already usable in the RPA production environment. Microsoft PowerShell offers sufficient tools for database SQL querying and stream editing. The amount of data was significant which made straight file editing impossible due to the memory limitations. Thus, stream editing was necessary for finding and replacing sensitive information from the data.

`TODO: maybe references?`

Anonymization took good proportion of the time in workdays as processes were slow, the amount of data was huge and multiple re-runs were needed before the results were seemed adequate.

`TODO: Appendix of the script used. Does this need more explaining?`

3.4 Data formatting

At the beginning of the research, the log data from RPA was in SQL database. However, the database used was not “pure” in a way that typical relational databases are, but some columns included JSON-formatted data in them. For ML algorithms to be able to read the given data with ease this kind of impurities needed to be cleared from the data.

When feeding the log data to anomaly detection algorithm it was necessary that all the rows were as minimally unique as possible in order to use the pattern finding abilities of the algorithm. Too unique data points would have made all of them anomalies compared to each other. Thus, all unique features were stripped from the data, such as the fingerprint value that was unique for each data point.

Comment:

Also, job ID information and timestamps of the rows were removed momentarily from within the rawmessage.

Comment:

Some interesting formatting solutions were used in this phase. Rawmessage was practically created again so that it fitted in the CSV format. This should be explained in some level of detail.

`TODO: check above section so it makes sense`

3.5 Azure environment

`TODO: This section is under construction! Here are some things we are discussing here:`

Comment:

Azure resources, such as virtual networks, storage spaces, connections to ML studio etc.

More info about ML studio, computing clusters, jobs, datasets and such. What kind of resources were usable based on the issue at hand and limitations by the company (cost, basically). We discuss some things about Azure ML Studio but only in extent of what parts needed configuring. More general info about ML Studio should be in chapter 2.

This needs to be formatted with Results-section so that information does not overlap. Maybe this section should be only on one of the sections.

3.6 Machine learning pipeline

Azure ML Studio makes ML pipeline creation easy and comparing different methods and algorithms effortless. Nevertheless, with hybrid approach having two different phases and result comparison being done against anomaly hypothesis, the pipeline drafts started to accumulate in content.

When starting the ML pipeline testing the initial plan was to feed the log data to anomaly detection algorithm and try to get some sort of estimate of possible anomaly count. This plan had several problems. First, as stated, logging is very abundant and several thousands of rows is logged during a single day. Some errors encountered are not critical and RPA agent is able to recover from them finalizing the initial task.

This means that errors that could be deemed anomalous may not result to a ticket in the end.

In addition, one single error case noticed by bank clerks may be linked to several problems in runtime, meaning that one ticket received is, in fact, linked to multiple, dozens or even hundreds of log rows.

Two different algorithms were needed. In phase 1, algorithm defines how likely one datapoint, or log row, is to be considered an anomaly. In phase 2, another algorithm aims to predict how many tickets are to be expected to receive within a time frame. This dual algorithm approach is referred as a hybrid machine learning approach.

In this subsection we discuss each phase and their

Time frame compression and statistical features

To avoid the problem with random delay between log rows and technical ticket timestamps, as stated in the section 2.9, log rows were grouped by time stamp into certain time frame groups with a method we call "time frame compression",

This means that in order to eliminate the effects of random delay we compress some features in certain time frame which is at least as long as the longest estimated delay. Simply put, if we count possible anomalies during one hour of log, we cannot compare this number to actual tickets received at the same hour or the next. What we can do, with time frame compression, is that we count some statistical values of anomaly estimates, for example, the mean and median values of a week, and then compare these numbers with the tickets received during the same week.

Comment:

This part explains some statistical features used when starting regression training in hybrid ML phase 2. These features consist of both log row amounts and PCA-component output values. All these values are grouped together in timescales.

Unconventional training approach

As stated in section 2.1, the approach used in this study is, if expression is allowed, unorthodox. Typically, data points used in ML algorithm training and validating should always be different. Acting otherwise leads to algorithm processing with same data it was trained with thus creating a situation where algorithm already knows what to do with the current data point. If the results were validated after this the algorithm would get unreliable score as it had the validation data already in the training phase. This could be compared to giving some right answers to students during test and scoring test results as if no help was given. However, due to the nature of the study problem and contents of the data, it was decided to test whether bending this rule would provide better results in algorithm training.

Even though the amount of data was large enough to cause issues with memory, the hybrid approach and the time frame compression discussed <later in this section>

lead to significant reduction of data in phase 2. As a general rule of thumb in ML training, only 20–30% of the data is used to validate the algorithm. With the hybrid ML approach, the validation results in phase 1 are what actually form the data used in the phase 2. This data is further compressed to time frame groups leading to only a few dozen data points in phase 2 ML training compared to millions of rows in phase 1.

Because of the way the anomaly detection algorithms work the over-lapping data points is not as big of an issue as it would be with other types of algorithms like regression algorithms. This is why we could use part of the data for training the anomaly detection algorithm as usual and then use all of the available data for validation without overfitting the algorithm. Also, because the main forecasting functionality comes in the phase 2, overtraining in phase 1 may not cause issues.

To verify if this unconventional training method gives good results without issues, the trained algorithms were tested with new production data that had zero overlapping data points with training and validation data. This way we were able to compare different training approaches to determine the best overall pipeline.

`TODO: Under construction below:`

ML Pipeline sketching

`TODO: Something about the pipeline as a full. Or move to results section.`

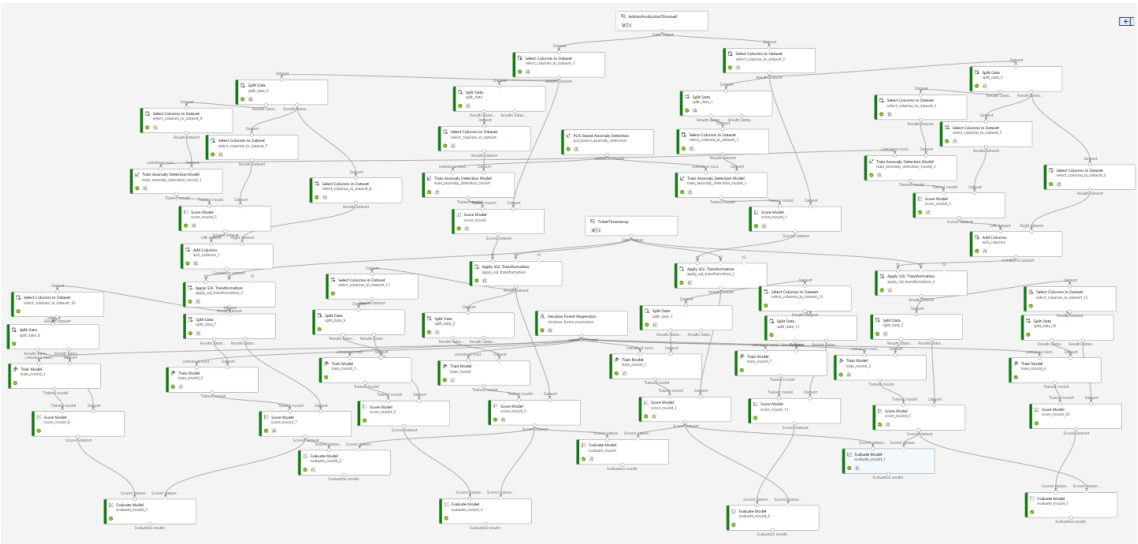

Figure 2: Pipeline in all its glory

Comment:

Next in this subsection, ML studio pipeline used is explained in more detail. This means pictures about the pipeline, explanations about the components used and their parameters, etc. etc.

Some of the content might be wise to move to the Results section. This needs more somewhat more thinking, also based on the results...

Memory issues and limitations

Memory is crucial in ML training as multiple steps happen and data is formatted etc. . While building ML pipeline in Azure ML studio, a memory issue emerged that affected several components and caused serious limitations in terms of usable components and data size. Due to the time limits of this study this issue was not resolved and the problem behind it was not found. As several conditions considering the environment costs were already issued by the company, the issue was declared to be linked with compute instance property limitations. However, this was not certain.

`TODO: Limitations to components`

Feature format for PCA-ADA

In Azure ML Studio there is only one module selectable for anomaly detecting, the PCA-based anomaly detection module, which is explained in section 2.5. However, with textual input like logs it can be used at least in two ways. First, input data can be fed to the algorithm trainer as is, letting the PCA-based ADA component do the work without further modifying the log rows. This way, the component tries to recognize the anomalies based on all the information included in the row. Practically this means that the component processes data in textual format making each row in the input a feature as a whole to consider.

`TODO: PCA-ADA should be explained in background-section`

Second option is to convert the textual features into numerical n-gram features. Each word or n-gram is now a number of said instances found on the row being processed, and each row can be presented as a sequence of numbers indicating the number of those features.

N-grams can in addition have a weight based on the frequency they appear in the entire data. Different weights usable in Azure ML component are listed below.

1. Binary Weight
2. TF Weight
3. IDF Weight
4. TF-IDF Weight

`TODO: Explain different weights and open up more Azure ML Studio PCA-component`

Anomaly probability

Comment:

Here we explain what PCA-component outputs and how the result is used in the pipeline.

The output values of the PCA-ADA component are, as explained in the section 2.5, normalized so the values range between 0 and 1. This anomaly probability value is the main output of hybrid ML phase 1. Based on our initial hypothesis that each anomalous event in the log is linked to a real life support ticket received, the bigger a single anomaly probability value is for a log row the more likely that row is related to a ticket inducing event.

Regression based estimating

Comment:

Here is more information about different regression algorithms used in ML pipeline. Some basic information about all of them is given so the results are understandable by reader in the Result section.

1. Linear regression
2. Decision forest regression
3. etc.
4. etc.

4 Results

We start this section by looking at the Azure resources needed for ML training both in general Azure environment and in ML Studio. Next we analyze the results of different ML algorithms and pipelines.

4.1 Azure and Azure ML Studio

Azure resources

As stated in section 3.5, Azure provides a vast set of tools and resources for different kinds of cloud projects. Resources needed for Azure ML Studio usage depends on the subscription used and security restrictions set by subscription administrators. When starting this study, due to these restrictions, all resources used for ML training in this project needed to be inside the same virtual network. Azure ML Studio environment could be opened from any network, but most of the features were unavailable if computer browsing studio UI was outside this virtual network. Thus, a virtual machine had to be acquired as Azure resource from within the same network as other resources and ML Studio UI had to be opened with the browser on this machine. Later on it was found that Azure Machine Learning Workspace networking feature could be configured to allow public access making possible to access ML Studio UI from all networks.

Azure ML Studio components

During the initial pipeline runs the execution came to an abrupt stop and Azure notified about memory issues. These problems were linked to the data amount which had to be reduced to 600 megabytes before any pipeline could be finished using the data. This reduction was against the initial goal where preferably all the data could have been used.

Considerable amount of time was used to fix or avoid this issue but nothing clear was found that would explain the error received. While working with the issue it was also noted that data needed more cleaning in order to ease the preprocessing phase as described with more detail in section 3.3 Thus, the data had to be imported from log archive and anonymized once more.

To advance the study more efficiently it was decided to trim info-type log messages from data hence reducing the data amount considerably. Final data included 8.6 million log rows which was about 10% of the original data size. Before final cleaning operations the data took 8.1GB od disk space, and after cleaning the rawmessage-field the final disk size was 6.6GB. Even with this data size, some Azure ML components faced this memory issue and forced us to choose such components that were able to handle these data amounts.

In addition to the data we used to train the algorithm, we needed to set up computing instance in Azure ML studio. Some predefined resource limitations affected the computing instance choosing and the memory issue encouraged us to

pick memory prioritized instances. Single computing instance did not work, but we needed to choose a computing cluster instead to be able to run ML training pipeline.

4.2 ML training and validation

In the methods section 3.6 we discussed of different approaches that could be used to get the best results from the ML training. In order to compare different results, some comparable metrics are needed.

Several crossing points caused the pipeline to diverge. First, the error message used to calculate the anomaly probability of a log row had two options. We could either use simple *message*, or more verbose *rawmessage*. This textual data could be fed to the ADA-component in several forms. Most straightforward way was using textual data without any preformatting. Text could also be run through "Preprocess text" -component. N-gram features could have been extracted from the text and these features could have been used instead. Instead of n-gram features, textual data could be converted to numeric with "Feature Hashing" -component. After getting the ADA-component results, the anomaly probabilities were compressed with R-code or SQL. <This concludes the phase 1.>

In phase two, diverging of the pipeline was due to either different regression algorithms used or result comparison to disregarding the results of phase 1. In practice this means that in order to validate the results against our initial hypothesis, we used statistical log data such as row count and unique job ID count without anomaly probabilities to determine whether anomaly metrics provided any insight regarding the ticket data.

Each diverging step, or layer, multiplies the amount of results used in final comparison that would determine the best possible pipeline combination. These layers are simplified in the table 1.

The divergent count implies the number of branches diverging from the previous component. The total count of branch ends, or leaves, would then be the multiplication of all divergent counts, totaling to 192 comparable pipeline combinations. Moreover, n-gram feature extraction and feature hashing have several tunable parameters that strongly influence the end results of the algorithm training. To reduce this amount when considering the best possible pipeline, we simplified this by narrowing down the options based on initial test run results of some of the divergent options.

For example, n-gram feature component suffered greatly from the memory problem and the data amount that the *Extract N-Gram Features from Text* -component was able to handle only 2% of the original data amount. This was deemed as too small amount for training as it would be extremely likely with 98% of the data skipped that also possible rows relevant to the ticket anomalies would get trimmed out.

Divergent layer	**Options**	**Divergent count**
Input text column	message rawmessage	2
Text preprocess	Yes No	2
Numeric conversion	No N-gram Feature Feature Hashing	3
ADA training	Unconventional Proper	2
Validation without anomaly metrics	Yes No	2
Regression algorithms	Decision forest regression Boosted decision tree regression Neural Network Regression Linear regression	4

Table 1: Pipeline divergent layers

Comment: Next is some initial values from pipeline runs. Subjected to change!

N-Gram feature extraction

Using Decision forest regression algorithm.

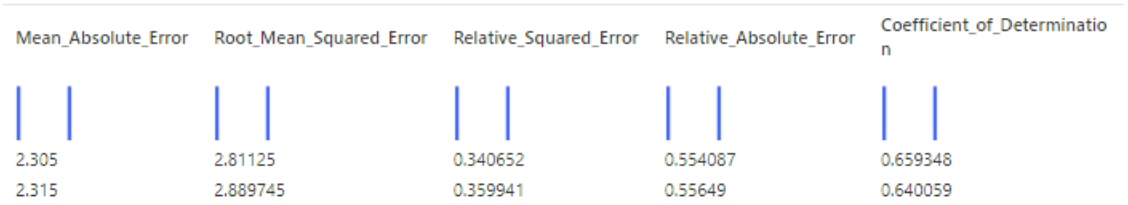

Figure 3: Message with N-Gram feature extraction using unconventional training method in phase 1, Decision forest regression in phase 2, compared to method without anomaly values.

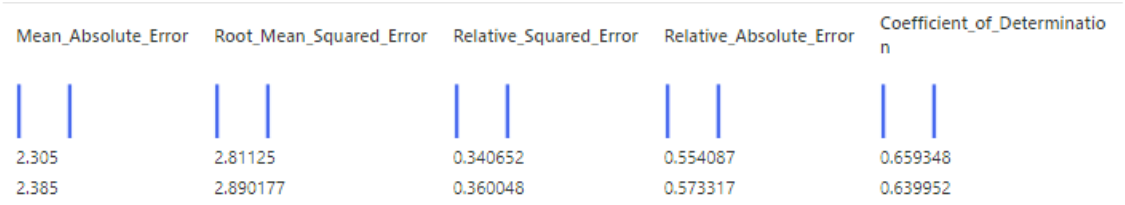

Figure 4: Message with N-Gram feature extraction using proper training method in phase 1, Decision forest regression in phase 2, compared to method without anomaly values.

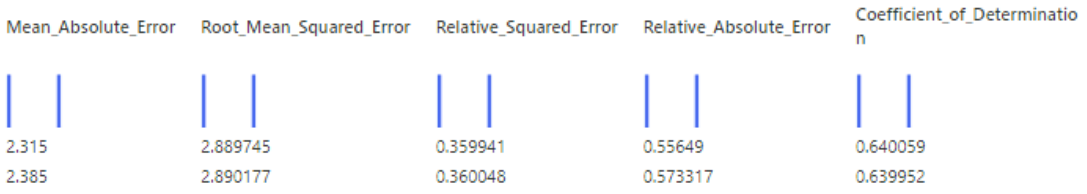

Figure 5: Message with N-Gram feature extraction using Decision forest regression in phase 2, comparing unconventional training to proper training.

Anomaly detection with pure textual data

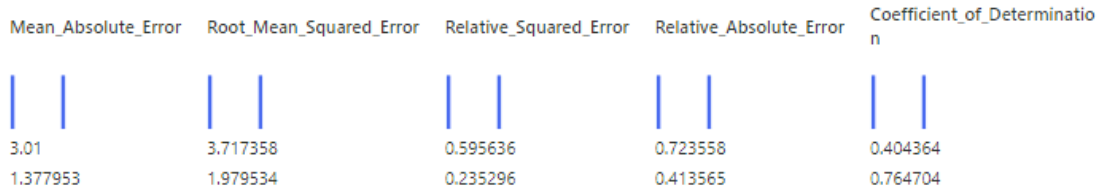

Figure 6: Message fed pure to ADA-component, using unconventional training method in phase 1, Decision forest regression in phase 2, compared to method without anomaly values.

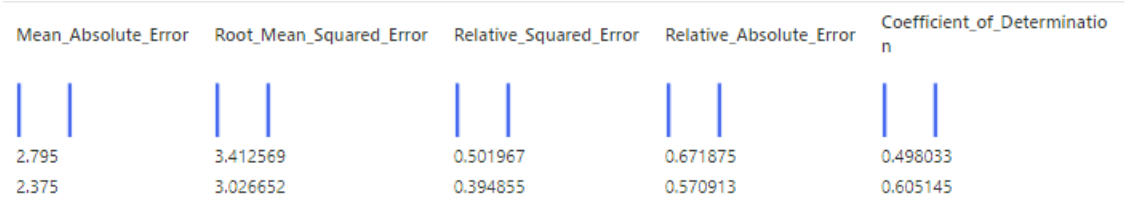

Figure 7: Message fed pure to ADA-component, using proper training method in phase 1, Decision forest regression in phase 2, compared to method without anomaly values.

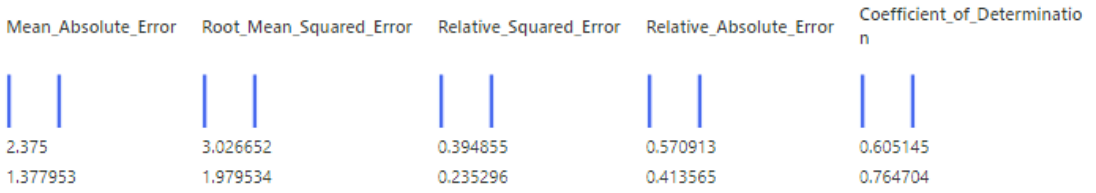

Figure 8: Message fed pure to ADA-component, using Decision forest regression in phase 2, comparing unconventional training to proper training.

5 Summary

TODO: Very under construction...

<Sum up here what we did and why>

5.1 Discussion

<Here some thinking what should have been improved>
Integrating with real time logging?

Data formatting

The most time-consuming tasks in the study was without a doubt the anonymization and preformatting of the data. Although sensitive information may sometimes be crucial in error fixing as problems may consider just one client, it is necessary that the data sanitation is possible to do in order to use the data in less secure environment. By preformatting the data in such way that all different personal information types do not differ between use cases.

Possible ML methods

Some sort of time delay forecasting?[20] Could we estimate the number of tickets based on some log metrics in time frame?

Memory error fixing would have given more options.

TODO: Something else specifically needed?

If log would have been better organized/preformatted it would have been possible to use One-Class Support Vector Machine. This, however, would have needed a manually constructed dataset including only "normal" log events or such events that could have been certain of that they were not part of the ticket inducing issues.

Some more testing would be needed to determine whether splitting data randomly or not in the phase 1 provides better results. By splitting data randomly it is possible to miss important anomalous rows that come in groups which would provide necessary insight to identify anomalous events.

The original hypothesis was that anomalous events in the logs were clearly linked to the tickets received. However, as memory errors due to the size of data forced us to skip info-typed rows, it is possible the data anomalies did not reflect to the tickets. As stated before, multiple error lines in the log may be linked to a single issue, which could make the ticket inducing events more common log feature. Thus, by tuning the statistical values used to take into account more common error messages, it could have been possible to get better results.

References

[1] P. K. Donepudi, "Machine learning and artificial intelligence in banking," *Engineering International*, vol. 5, no. 2, pp. 83–86, 2017.

[2] W. M. Van der Aalst, M. Bichler, and A. Heinzl, "Robotic process automation," pp. 269–272, 2018.

[3] A. DeLaRosa, "Log monitoring: not the ugly sister," *Pandora FMS*, 2018. [Online]. Available: https://web.archive.org/web/20210901031146/https://pandorafms.com/blog/log-monitoring/

[4] W. K. Ho, B.-S. Tang, and S. W. Wong, "Predicting property prices with machine learning algorithms," *Journal of Property Research*, vol. 38, no. 1, pp. 48–70, 2021. [Online]. Available: https://doi.org/10.1080/09599916.2020.1832558

[5] K. Ghanem, F. J. Aparicio-Navarro, K. G. Kyriakopoulos, S. Lambotharan, and J. A. Chambers, "Support vector machine for network intrusion and cyber-attack detection," in *2017 Sensor Signal Processing for Defence Conference (SSPD)*, 2017, pp. 1–5.

[6] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[7] "Definition of algorithm," https://web.archive.org/web/20220510183749/https://www.merriam-webster.com/dictionary/algorithm, accessed: 2022-05-19.

[8] T. O. Ayodele, "Types of machine learning algorithms," *New advances in machine learning*, vol. 3, pp. 19–48, 2010.

[9] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.

[10] "Comparing machine learning as a service: Amazon, microsoft azure, google cloud ai, ibm watson," https://www.altexsoft.com/blog/datascience/comparing-machine-learning-as-a-service-amazon-microsoft-azure-google-cloud-ai-ibm-watson/, accessed: 2022-04-04.

[11] F. Anifowose, "Hybrid machine learning explained in nontechnical terms," *JPT*, 2020. [Online]. Available: https://web.archive.org/save/https://jpt.spe.org/hybrid-machine-learning-explained-nontechnical-terms

[12] T. Shon and J. Moon, "A hybrid machine learning approach to network anomaly detection," *Information Sciences*, vol. 177, no. 18, pp. 3799–3821, 2007.

[13] C.-F. Tsai and M.-L. Chen, "Credit rating by hybrid machine learning techniques," *Applied soft computing*, vol. 10, no. 2, pp. 374–380, 2010.

[14] S. Mohan, C. Thirumalai, and G. Srivastava, "Effective heart disease prediction using hybrid machine learning techniques," *IEEE access*, vol. 7, pp. 81 542–81 554, 2019.

[15] N.-C. Hsieh, "Hybrid mining approach in the design of credit scoring models," *Expert Systems with Applications*, vol. 28, no. 4, pp. 655–665, 2005.

[16] A. Jain and A. M. Kumar, "Hybrid neural network models for hydrologic time series forecasting," *Applied Soft Computing*, vol. 7, no. 2, pp. 585–592, 2007.

[17] H.-j. Kim and K.-s. Shin, "A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets," *Applied Soft Computing*, vol. 7, no. 2, pp. 569–576, 2007.

[18] T.-S. Lee, C.-C. Chiu, C.-J. Lu, and I.-F. Chen, "Credit scoring using the hybrid neural discriminant technique," *Expert Systems with applications*, vol. 23, no. 3, pp. 245–254, 2002.

[19] R. Malhotra and D. Malhotra, "Differentiating between good credits and bad credits using neuro-fuzzy systems," *European journal of operational research*, vol. 136, no. 1, pp. 190–211, 2002.

[20] G. H. Erharter and T. Marcher, "On the pointlessness of machine learning based time delayed prediction of tbm operational data," *Automation in Construction*, vol. 121, p. 103443, 2021.

A Esimerkki liitteestä