UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Samreen Mahak Hassan

Classification and Prediction of Business Incidents Using Deep Learning for Anomaly Detection

Masters's Thesis (30 ECTS)

Supervisor:	Aleksandr Tavgen
Supervisor:	Martin Kiilo
Supervisor:	Raimundas Matulevičius, PhD

Tartu 2019

Classification and Prediction of Business Incidents Using Deep Learning for Anomaly Detection

Abstract: Companies today use a number of software systems to carry out various business activities. Such enterprise standard software solutions consist of a large number of components usually developed by different teams and/or different software vendors using various technologies. In such complex software systems, there can be various issues ranging from problems in the software itself to issues in network.

In order to measure the operational performance of applications and infrastructure as well as key performance indicators (KPIs) e.g. new customers, revenue, that evaluate the success of the organization, a lot of business metrics is collected. These metrics have certain data patterns which represent normal business behaviour. Anomalies are the unexpected changes within these data patterns such as degradation or a sudden surge in business metrics values. Additionally, a small change in software system configuration can cause unexpected behaviour in business flows. Version upgrades of different components can introduce compatibility problems. These problems could lead to a change in the normal behaviour of business metrics and cause anomalies. These anomalies if not resolved quickly results in business and financial losses. Therefore, it is necessary for businesses to take proactive steps to manage such business incidents before they can adversely affect it. This brings us to the need for an analytics platform which can analyze patterns of data streams, identify and differentiate normal behaviour of a business metric from anomalous behaviour and could generate notifications.

The current anomaly detection and alert system in Playtech plc uses a simple anomaly detection technique that follows a rule based approach and it is observed that it is not efficient. Thus, a more robust, modular and efficient business incident/anomaly detection solution based on advanced machine learning techniques is needed that could work in conjugation with the current system. This thesis proposes, describes and evaluates a business incident/anomaly detection system based on deep learning approach that categorises and predicts the business incidents/anomalies using the available business metrics information.

Keywords:
Anomaly Detection, Business Incidents, Machine Learning, Deep Learning, Convolutional Neural Networks, Classification, Predictive Analytics, Playtech

CERCS: P170 Computer Science, numerical analysis, systems, control

Äriintsidentide klassifitseerimine ja prognoosimine kasutades süvaõpet anomaaliatuvastusel

Lühikokkuvõte: Tarkvarasüsteemid omavad tänapäeva ärieettevõtetes elutähtsaid funktsioone ja nad on tihti ka äritegevuseks primaarse tähtsusega. Taolised süsteemid võivad koosneda väga suurest hulgast komponentidest, mis on arendatud erinevate meeskondade või ettevõtete poolt ning enamasti ka kasutades erinevaid tehnoloogiaid. Keerukate süsteemide korral võivad olla vead nii rakendustes kui ka võrgus.

Probleemid võivad ilmneda konfigureerimisel, mis võib põhjustada ootamatuid pöördeid ärivoos, samuti võivad versiooniuuendused tekitada kooskõlaprobleeme. See kõik võib põhjustada ärile maine- ja finantsilist kahju. Seetõttu on ärile vajalikud proaktiivsed sammud, et tulla toime äriintsidentidega enne nende ebasoodsat mõju teistele komponentidele. See toob kaasa vajaduse analüütilise platvormi järele, kus oleks võimalik eristada süsteemi normaalset käitumist anomaalsest meetrika alusel.

Playtech plc kasutab taoliseks automaatseks tuvastamiseks ja häirete tõstatamiseks tüüpilist anomaaliate tuvastamise lähenemist: reeglitel põhinevat tuvastamist. Playtech plc, tarkvarasüsteemides jälgitakse tuhandeid meetrikuid, alustades infrastruktuuri ja süsteemitarkvara ning lõpetades rakenduste ja ärimeetrikutega. Samas on tarkvara paigaldatud ja opereerib rohkem kui 40-s asukohas, igas neist erinevate lõppkasutajate ning ärimudelitest tulenevate erinevustega. Lisaks sellele, on tarkvara pidevas muutumises, nädalaste arendustsüklite tulemustena uuendatakse igal teisel nädalal komponente üle kõigi asukohtade ja paigalduste. Reeglitel põhinev lähenemine on piisavalt efektiivne tuvastamise kiiruse ja täpsuse osas, kuid nõuab palju inimressursse reeglite haldamise ja täppisseadistamise tõttu sellises muutuvas keskkonnas. Seetõttu nähti vajadust leida lahendus mis suutaks automaatselt kohaneda muutuvas keskkonnas ning erinevates tarkvara seadistustes ilma inimese pideva sekkumiseta. Antud töö eesmärk ongi masinõppel põhineva mudeli väljatöötamine ja treenimine, mis tuvastaks ja kategoriseeriks taolisi intsidente. Töö kirjeldab detailselt, kuidas kasutatakse anomaaliate tuvastamise ja süvaõppe tehnikaid täiendamaks olemasolevat lahendust intsidentide tuvastamisel ja klassifitseerimisel.

Võtmesõnad:
Anomaalia tuvastamine, äriintsidendid, masinõpe, süvaõpe, konvolutsioonilised närvivõrgud, klassifitseerimine, ennustav analüüs, Playtech

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

This chapter describes the problems that motivate this thesis, the research questions and briefly mentions the proposed solution. Section 1.1 describes the problem of business incidents on an application server. It mentions the importance of detection and categorisation of such incidents. Section 1.2 gives an overview of the goals to be met, the contributions made during the course of this work and the research questions. Section 1.3 gives the outline of this thesis and the following chapters.

1.1 Motivation and Problem Statement

An anomaly means a deviation from normal behaviour or occurrence of something unusual [1]. Any unexpected change in the data patterns or an event that does not conform to the usual operation or expected actions is considered to be an anomaly and often termed as business incidents, exceptions etc. An organization observes different business incidents like unavailability of web services, unreasonable high number of failed financial transactions, unusual user logins, high error rate, web server's CPU usage, network issues, abrupt drop in the organization's business key performance indicators (KPI) and metrics in a day to day operation and so on.

The increasing number of business incidents have become a top concern for organizations as they lead to disruptions in normal operations and causes financial losses. This could also have adverse effects on the organization's reputation. Therefore, it is important to detect such business incidents.If these incidents are not detected and acted upon, they could cause the other related components in the system to misbehave or stop working completely. Moreover, there could be temporary overloads on other parts of the system which causes more time, effort and costs to maintain the normal business KPIs and metrics.

Delayed, slow or no detection of business incidents can have serious business impact. For example, according to ITIC's 2017-18 Global Server Hardware, Server OS Reliability Survey [2], unplanned outage of an organization business server resulted in an average cost of 300K US dollars. Therefore, it is necessary for businesses to detect and identify such business incidents. An organization does this with the help of some monitoring, profiling, logging and other tools to detect anomalies. In a practical scenario, business incidents are inevitable, so it is important that an organization dedicates efforts to detect and categorise such incidents in time. Fast and correct categorisation of business incidents is important to make sure that high priority incidents are managed first to reduce the overall impact on the business. For example, it is important to deal with anomalies in financial transactions before as compared to lower priority incidents like reduced server average response time.

Playtech plc[1], a leading gambling software provider company, also faces similar challenges. It offers a variety of products such as software for Casino, Poker, Bingo etc. Its platform is maintained and managed by the Integrated Management System (IMS)[2] team. It is observed that there are certain deviations from normal behaviour like unusual drops or surges in their business metrics and KPIs at times. Playtech plc aims to proactively manage these anomalies and business incidents before they affect their customers and lead to financial losses. Hence, there is a need to detect anomalies as well as to take necessary steps like raising alert to the concerned team and applying corrective measures.

Playtech plc initially used a Hewlett Packard's business service management software solution called Service Health Analyzer (SHA) [3] for anomalies detection. According to the teams using the SHA system in Playtech plc, as the number of business metrics increased, it was not possible to rely on just SHA as it gave a lot of false positives. Moreover, a lot of effort was required in the configuration. Therefore, Playtech plc, in 2017, decided to migrate to an advanced and quicker approach using rules matching engine (see Section 5.1) to search for any anomalies and send out notifications to the concerned team. Consequently, it was found that rules matching approach has certain disadvantages such as managing all the syntactic rules manually, adding violation messages etc. Also, the rules matching engine does not give the capability to automatically classify incidents based on severity. The manual process is repetitive and time-consuming. Thus, Playtech plc decided to employ machine learning techniques to automatically detect and categorise business incidents/anomalies.

The idea is to discover if these machine learning methods work for this business problem. It is a research-based project aiming at solving a real-life industry problem by using a research-driven approach and methods. The approach for this proposed solution, which uses deep learning, is discussed over the course of this work. Such a solution would help to identify the business incidents, reduce the time taken to fix the issues and avoid disruptions in day to day business operations.

1.2 Research Questions

Anomaly detection and categorisation of business incidents is quite an extensive topic. So in this work, we limit ourselves to the below-mentioned goals.

The main objective is to design and develop a solution that identifies and categorises anomalies/business incidents. This thesis explores the possibility of using deep learning approach i.e. Convolutional Neural Network (CNN) to achieve the main objective and determine how effectively CNN functions in this real-life operational setting.

The project lasted for about eight months and comprised of various activities includ-

[1] https://www.playtech.com/

[2] https://www.playtech.com/technology/

ing a clear understanding of business requirements and use case. The activities involved was to collect, clean and transform large data, implementing, training and evaluating the chosen machine learning model. In the course of the project, the following main question (MQ) was addressed:

MQ - How to detect anomalies/business incidents and categorise them using machine learning?
While achieving the main objectives of this thesis, it is anticipated that the proposed solution also contributes to answering additional sub-questions (SQs) as follows:

SQ1 - What are the current approaches and solutions that exist for anomaly detection?
Study of different research-based approaches as well as commercial anomaly detection solutions to learn and understand the current state of the art.

SQ2 - How to design and develop a viable solution for the detection of anomalies and categorisation of business incidents?
The information from SQ1 is used to design and develop a solution using a deep learning approach.

SQ3 - What is the efficiency of the proposed solution for detection of anomalies and categorisation of business incidents and how is it measured?
This SQ incorporates what are the methods to evaluate the model and the developed solution.

1.3 Thesis Organization

The rest of this thesis is structured as follows and contributes to providing a detailed answer to the main question (See **MQ** in 1.2).

Chapter 2 discusses the state of the art anomaly detection solutions. It focuses on the different related research works carried out in this field along with few commercial solutions.

Chapter 3 provides a brief explanation of the important terms and background knowledge necessary to understand the thesis. It also describes the environment setting where the proposed solution is to be used and the preliminary system currently used in Playtech plc.

Chapter 4 describes in detail the software requirement specification (SRS) by specifying functional and non-functional requirements that is expected from the proposed solution.

Chapter 5 discusses the design and architecture of the proposed solution, the involved workflow and different individual components. It also describes the structural properties of the data and necessary pre-processing of data along with the assumptions and scope.

Chapter 6 describes the CNN based model used in the proposed solution.

Chapter 7 gives the details of a prototypical implementation of the proposed solution along with the interaction among various components. It details the data pre-processing techniques and the implementation of the CNN based model.

Chapter 8 illustrates the evaluation of the CNN based method and the proposed solution. It also includes the experimental setup, performance evaluation data and the results.

Chapter 9 finally concludes this thesis and presents the outlook of the future work.

2 Related Work

This chapter discusses the state of the art and describes different related works done in this field. It provides an answer to the first sub-question (**SQ1** in Section 1.2). To answer this question in detail, it is broken down into three sub-questions: 1) What is anomaly detection? 2) What are the research-based approaches? 3) What are the current commercial solutions available? Sections 2.1, 2.2 and 2.3 provide answers to these questions respectively.

2.1 Anomaly Detection

An anomaly can be described as an event that does not conform to an expected pattern in the data [1]. It is the set of values that does not seem regular or unfit in a usual dataset. In the context of an organization's network and application servers, these anomalies can be a sudden increase or an unusual drop in certain activities like delayed average response time or high error rate which is not commonly observed.

Authors Schwartz and Jinka define anomaly detection as a set of techniques and approaches to find unusual behaviours and/or states in systems and their observable signals [1]. Anomaly detection is the identification of items, observations or events that differ significantly from the majority of data [4]. There are three main broad categories of anomaly detection: 1. Unsupervised which detect anomalies in an unlabeled dataset with the assumption that most of the data points have normal usual values. 2. Supervised anomaly detection techniques which detect anomalies in a labelled dataset with both normal and abnormal class labels [5]. 3. Semi-supervised anomaly detection where only the normal class data has labeled instances.

An anomaly detection software solution enables the organizations to automatically or semi-automatically detect anomalies. Such detection software compare events or deviations in patterns against already defined normal behaviour. Solutions from different anomaly detection software providers use different techniques such as statistical methods, cluster-based techniques and machine learning approaches [6]. These software also exist for domain specific detection of anomalies like network intrusion detection in software security domain, fraudulent transaction detection in financial domain etc. An anomaly detection software monitors events and derive patterns from the metrics. It can then identify certain deviations and notify the users by sending alerts. This software can be integrated with the monitoring systems and dashboards can be used for visualizing the metrics.

This thesis aims to design and develop a solution for the detection and categorisation of business incidents/anomalies. This is an extensive research area and the next Subsection describes the two broad categories of existing solutions: 1. Research-based solutions 2. Commercial solutions.

2.2 Research-based Anomaly Detection Solutions

Anomaly detection is an active research topic and there exists a plethora of research in different domains for example in security domain for intrusion and fraud detection, for fault detection in safety-critical systems, in health care and so on. Hence, in this section, only a few categories of the researches done in this field which is partly related to the solution present in this thesis are explained.

Patcha et al. [6] provided a comprehensive survey of anomaly detection systems and hybrid intrusion detection systems. They described various statistical methods, data-mining based methods, machine learning based techniques and hybrid approaches proposed for anomaly detection along with the advantages and drawbacks.

Chalapathy et al. [7] in their work reviewed deep learning-based anomaly detection in various domains such as in social networks, system logs analysis, time series analysis and assessed the effectiveness of the used methods.

The following works describe the use of unsupervised, supervised, graph-based and CNN approaches for anomaly detection in different domains. In their study [8], Lane et al. presented a traditional machine learning approach for anomaly detection with the goal to automatically detect violations of security policy of a website hosted on UNIX server.To learn characteristic patterns of actions of the users, their proposed system used the sequence of actions as the fundamental unit of comparison. The underlying hypothesis is that a user responds in a similar manner to similar situations, leading to repeated sequences of actions [8].

According to Görnitz et al. [9], the predictive performance of purely unsupervised anomaly detection often fails to match the required detection rates in many tasks and there exists a need for labeled data to guide the model generation. So they devised a semi-supervised anomaly detection algorithm for their work. They proposed a very interesting active learning strategy to automatically filter candidates for labeling and observe that this methodology requires much less labeled data while achieving higher detection correctness.

Schindler et al. [10] analyzed real-world log data and tried to detect anomalies to defend against advanced persistent threats. They used a graph analysis based approach to successfully detect simulated attacks by analysing the log data of a simulated computer network. It is shown that the proposed solution significantly reduces the detection time of breaches and react faster to newer attack vectors [10].

In [11], Chouiekh et al. used a Deep Convolutional Neural Network (DCNN) for detecting fraudulent activities in the mobile communications domain. They measured the performance of CNN against traditional machine learning algorithms.

CNN is used widely for image recognition tasks but lately, research has been carried out to use them with other types of data such as time series data. In this thesis, the data is business metrics data and the focus is to use CNN based model to detect anomalies/business incidents in a real enterprise environment. Few works which employ CNN based models on different types of data are listed below.

Zhengyao et al. [12] proposed an extensible reinforcement-learning framework solving the general financial portfolio management problem. They used a CNN based approach in their experiments to realize the framework. The input to the network is a price tensor and the output is the portfolio vector. They used CNN in evaluation process for Reinforcement Learning applied to time-series data (financial data) and by results, it was concluded that CNN implementation had the best performance in several evaluations.

In another similar work [13], Wang et al. used a Fully Convolutional Network (FCN) for time series classification without any heavy pre-processing of the raw data and it showed better performance than other state of the art. In a different work [14], Oats et al. followed an approach to encode time series data as different types of images namely Gramian Angular Fields (GAF) and Markov Transition Fields (MTF). They used Tiled CNN to identify structure in time series and to classify GAF and MTF representations on 12 different datasets.

In their work [15], Yang et. al proposed a systematic feature learning method for human activity recognition problem. This method adopts CNN to investigate the multichannel time series data and automate feature learning from the raw inputs in a systematic way. There are 18 classes in this activity recognition task and the proposed CNN method consistently performs better than the baseline methods.

Cui et. al [16] propose a new model based on CNN called Multi-scale Convolutional Neural Network (MCNN) to carry out feature selection and classification of time series data in a single framework. The first layer of MCNN contains multiple branches that perform various transformations of the time series, including those in the frequency and time domains, extracting features of different types and time scales.

2.3 Commercial Anomaly Detection Solutions

In this section, commercial solutions for anomaly detection and business analytics such as Anodot[3] and IBM Watson[4] are discussed.

Anodot is a software as a service (SaaS) AI platform from Anodot Ltd. that monitors time-series data from different sources and uses machine learning techniques to discover anomalies in real time. Anodot is an autonomous analytics platform. This aim of this thesis is to create a prototypical implementation that looks somewhat similar to Anodot but which uses CNN based classifier.

Anodot makes use of their patented machine learning method and system to analyze business data in real time and alert whenever an incident occurs [17]. It identify and notifies the anomalies at an early stage before they lead to a major issue.

[3] https://www.anodot.com/

[4] https://www.ibm.com/watson/

Unlike traditional anomaly detection tools which monitor only infrastructure-centric data sources such as log files, CPU and memory metrics, Anodot helps to look for spikes in the data that might indicate a problem. It analyzes anomalies within the patterns of anomalies themselves [18]. Anodot can further distinguish between important and not-so-important anomalies. It also allows to create sophisticated composite functions to track advanced business logic that ties directly to KPIs. The result is business intelligence in real-time, leveraging all available data sources relevant to business performance.

Anodot follows a five-step process for anomaly detection: 1. Collection of business metrics 2. Learning normal behaviour 3. Learning abnormal behaviour 4. Topological study of normal and abnormal behaviour 5. Acquiring and processing data in real-time [18].

Furthermore, Anodot claims to be a data agnostics solution. It is able to input any type of time-series data in real time and immediately detect anomalies. It analyses input data to determine its normal range and detects anomalies. The anomalies are assigned a significance score depending on how unusual the anomaly is or for how long it lasts. Anodot chooses the most appropriate algorithm which best describes the data pattern. It can handle complex patterns such as trends and changing data behaviours.

IBM Watson is another commercial solution which is capable of doing advanced data analytics and computer vision. Although, anomaly detection is not the focus of IBM Watson platform but it could be used for the same. It is a machine learning based software platform of business-ready AI services and applications [19].

IBM Watson Machine Learning (WML) [5] is a service integrated with IBM Watson Studio that uses open source libraries such as Scikit-learn [6] and Apache Spark MLlib [7] for various machine learning modeling and statistical methods. WML works with data from different domains and allows user to create customized data analysis and prediction pipelines.

Watson Analytics (WA) [8] is another product of IBM Watson platform that provides data analytics and visualization services that one can use to quickly discover patterns and properties of large dataset. It enables automated predictive analytics and cognitive capabilities such as natural language dialogue to get the answer[20]. WA leverages deep content analysis and evidence-based reasoning to accelerate and improve machine learning decisions, reduce operational costs, and optimize outcomes [20]. It helps to analyze a trend or to visualize report data in a dashboard which can be used further to identify anomalies.

Besides the above described solutions, there are many other commercial solutions that exist for anomaly detection as well as for non-specific data analytics similar to IBM

[5] https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/ml-overview.html/

[6] https://scikit-learn.org/stable/

[7] https://spark.apache.org/mllib/

[8] https://developer.ibm.com/watson-analytics/

Watson and some of which are briefly mentioned below.

Loom Systems [9] gathers and analyzes different types of logs and metrics automatically and over time learns their behaviour. It detects and reports anomalies and trends including the root cause and resolutions for the anomalies. In Loom systems everything is done automatically without any manual data pre-processing or feature selection.

Numenta [10] is based on principles of the 'neocortex'. They argue that it is ideal for large-scale analysis of continuously streaming data sets and excels at modeling and predicting patterns in data for detection of anomalies.

SAP Predictive Analytics [11] uses automation to build sophisticated predictive models that can be embedded in business processes to help in anticipating future behaviour, predicting outcomes and lead to profitable decision-making across digital businesses. One aspect of the solution is clustering analysis to analyse groups of products, customers, employees and improve decision making. SAP Predictive Analytics provides an automated module for clustering in the Automated Analytics interface which helps in defining and generating clustering models [21]. SAP Predictive Analytics solution has many other applications relative to the needs of the user.

2.4 Summary

In this chapter, the state of the art and commercial solutions are discussed to answer the research question "What are the current approaches and solutions that exist for anomaly detection?" (**SQ1** in Section 1.2).

[9] https://www.loomsystems.com/

[10] https://numenta.com/

[11] https://www.sap.com/products/predictive-analytics.html

3 Background

To achieve the goals defined in Section 1.2, it is required to understand some background on key technologies and mechanism that are tightly integrated with this thesis. This chapter describes the key topics and terms like Machine Learning, Machine Learning Process, Convolution Neural Networks, Time series etc. Section 3.3 describes the method and the environment setting of this thesis explaining existing Playtech plc anomaly detection system and the used technologies. It is recommended to go through it to better understand the later chapters.

3.1 Time Series

Time series is a continuous sequence of values of a variable or events which are collected at equally spaced fixed time intervals [22]. Time series are analyzed to understand the long term trend, to forecast the future values and trends or perform various other analysis. A time series can have increasing, decreasing or seasonality trends (variations specific to a particular time frame) [23]. Real-time surveillance systems and network sensors generate time series data which can be collected for valuable insights.

Time series data is used to obtain an understanding of the underlying forces that produced the observations, fit a model and aid in the monitoring and forecasting. Time series analysis has been used in multiple applications like sales forecasting, stock market analysis, census analysis and so on and so forth. Each of these applications has a chronologically listed data points like the number of sales, the value of stock etc. This data usually shows a trend line or some metrics which can be studied in order to forecast after the analysis.

A time series is stationary, if the mean and variance of the series is constant [24]. Figure 1 shows stationary and non stationary time series. If the time series is not stationary, one cannot build a model for time series. Modeling is done using moving statistics for mean and variance. A moving average is kind of a low-pass filter that passes signals with a frequency lower than a certain cutoff frequency. It is used to make the resulting time series data smoother and remove the noise but it leaves intact the main trend lines [23]. This approach has been used to monitor systems effectively and it helps to detect outliers and anomalies.

3.2 Machine Learning

Machine learning [25] is a field of computer science where a computer learns using some sample data without being programmed explicitly. Machine learning focuses on the development of algorithms and processes which can learn and make predictions for future data from the data provided/already available data. The two main types of machine learning are supervised and unsupervised learning. Supervised learning [26] is

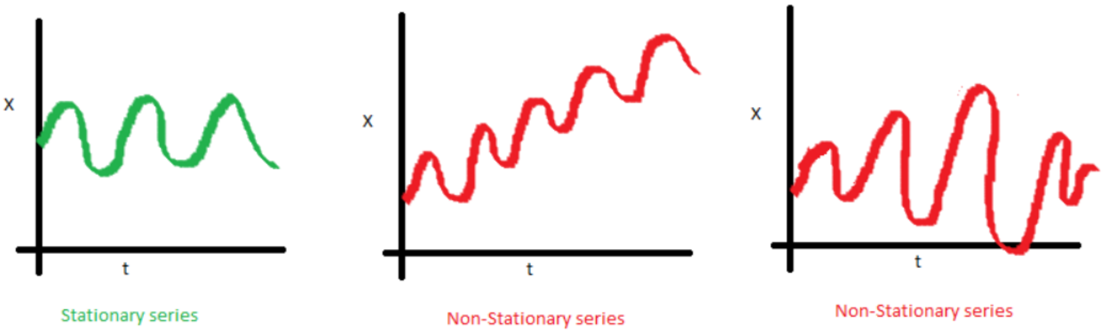


Figure 1. Stationary and non-stationary time series (adapted from [24])

used when the data has labeled inputs. This type of learning is primarily used to solve regression and classification problems. In supervised learning for classification, the main goal is to train the model to learn and classify an input accurately. Unsupervised learning [26] is used when the data has no predefined set of labels. It focuses more on exploratory data analysis and uses methods such as pattern matching and clustering. Clustering [26] is a technique where similar examples/instances in data are grouped together in clusters and are used to discover patterns in data. It can also be used to classify an unlabeled dataset which can further be used for supervised learning.

3.2.1 Machine Learning Process

Machine learning process is an iterative process which requires choosing and developing a machine learning model, evaluating the model and then repeating the steps until the required results are achieved. Figure 2 represents a standard machine learning process.

The first step is to get the data and structure it the way as required by the process. Data cleaning and pre-processing [28] is one of the tedious and most time-consuming parts of the whole machine learning process but it is important so that the process could understand the data. It is necessary to check for outliers, redundant values, missing values and tackle them either by filling them with suitable values or removing such rows altogether. The next step is to visualize the available structured data so as to have a clearer picture of the data and come across any obvious patterns in the data. The next task is feature engineering where important features are selected, extracted and engineered, to be used as input for the machine learning models. Once the features are available, the algorithms could be applied to the selected features from the data. This is an iterative process. The result of the algorithm applied data is a candidate model meaning the first most appropriate model that is trained. In course of time, several candidate models are produced through the iterative process and evaluated based on certain metrics until the model is finalised and deployed. The final model can then be used to test further data

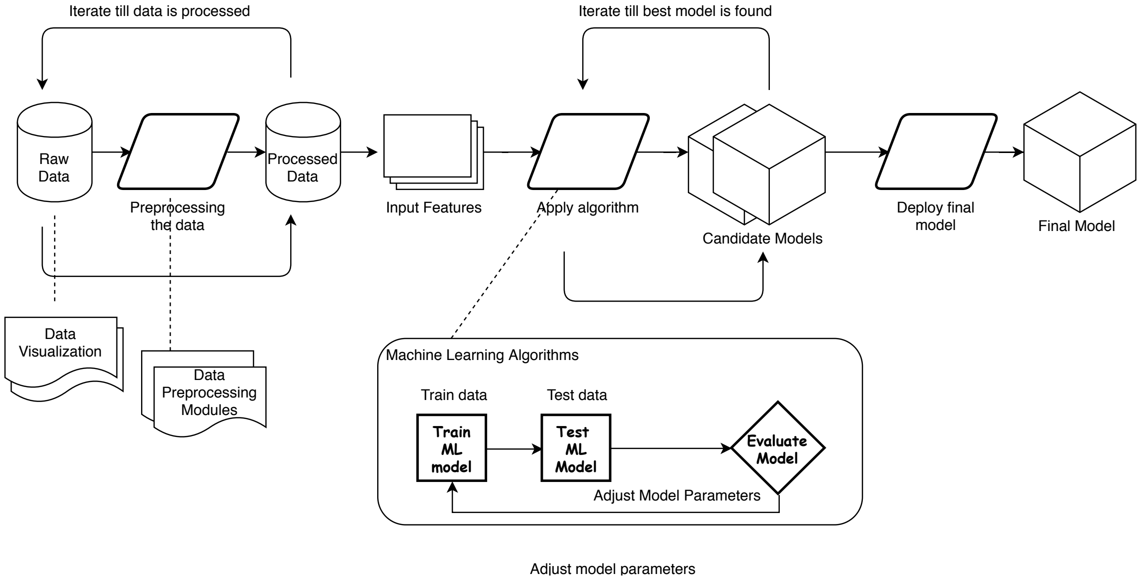


Figure 2. Machine learning process (adapted from [27])

and predict future data values.

3.2.2 Traditional Machine Learning

Traditional machine learning [29] models have been used far and wide to carry out the regular machine learning tasks like regression, classification and prediction. For classification and regression problem, there are multiple choices of machine learning models like Naive Bayes Classifier, Decision trees, Random Forest [29]. Each of these models follows a different algorithm approach and performs differently under different data set and parameters. Traditional machine learning algorithms need a lot of domain expertise and human intervention. In traditional machine learning techniques, the features should be identified by a domain expert so that the complexity of the data is reduced and the patterns are more visible to learning algorithms to work. In these techniques, the problems are broken into different parts to be solved first and then their results are combined at the final stage while deep learning tends to solve the problem end to end. Below, a few of the methods are described in brief.

Naive Bayes [30] is one of the many techniques to construct a classifier that assigns class labels. The Naive Bayes model is based on the Bayesian theorem and is used when there is a high number of input dimensions. The Bayesian analysis uses the concept of prior probability, conditional probability and maximum likelihood. Naive Bayes is called 'naive' as it assumes that all of the features in a dataset are equally important and independent.

A decision tree [30] is a technique that uses a tree-like structure representing the possible consequences and decisions to be taken at each step, including the chance of every event outcome. The internal nodes represent a condition or a decision test and each of the branches represents the outcome or consequences. The leaf nodes of the tree represent the final class labels and the path from the root of the tree until the leaves represent classification rules or decision rules. It makes use of the ID3 algorithm.

Another very important and widely used approach is the Random Forest Model [31]. It is an ensemble method for both classification and regression tasks [32]. It operates by building multiple decision trees while training and outputting the class (in classification task) or value (in regression task).

3.2.3 Artificial Neural Networks

Artificial Neural Network (ANN) [33] is a collection of human-brain inspired supervised learning models which are intended to replicate the way humans learn [34]. A neural network is composed of three main layers: an input layer, a hidden layer which consists of units that transform the input into something that the output layer can use and an output layer [35]. Each layer is composed of neurons which are interconnected to all the neurons in the next layer to construct a fully connected network as shown in Figure 3. They are very useful for finding patterns too complex for a human brain to recognize easily. The more advanced networks today are deep learning [36] neural networks in which the different layers extract different features until it can recognize the pattern it is looking for.

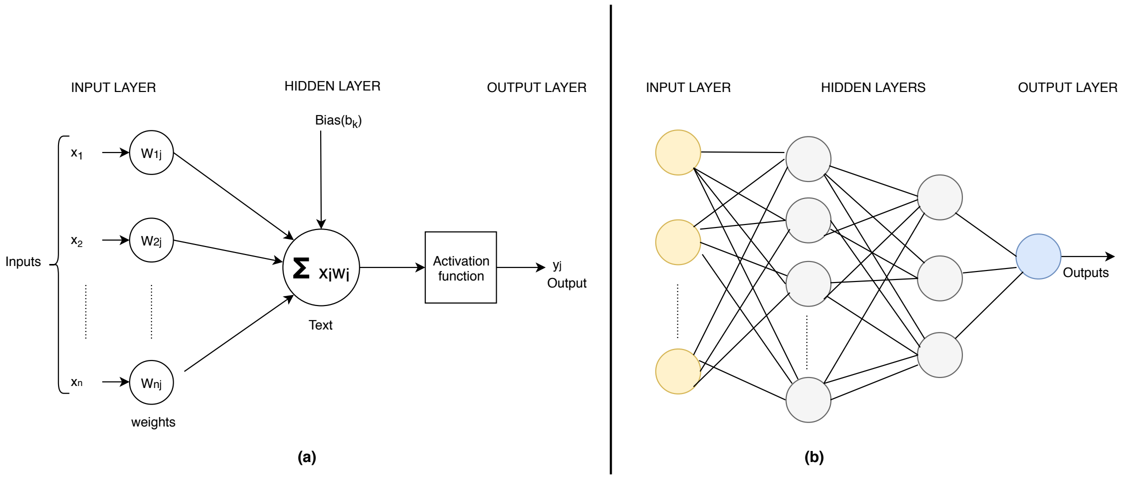


Figure 3. (a) A single neuron (b) 2-layered artificial neural network (adapted from [35])

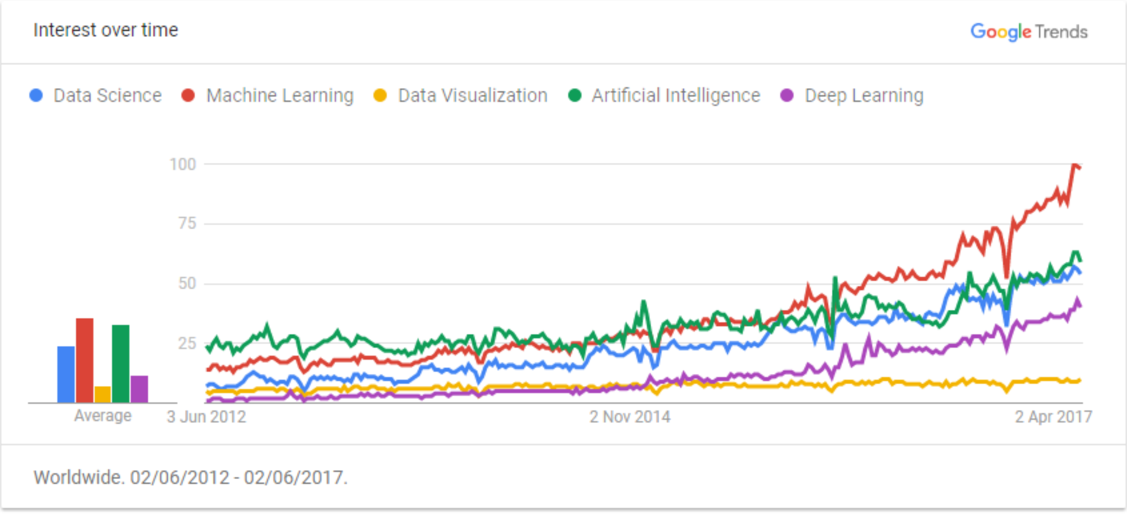


Figure 4. Google trends for machine and deep learning over the years (taken from [38])

3.2.4 Deep Learning and Convolutional Neural Networks

Deep learning [36] and Convolutional Neural Networks (CNNs) or ConvNets [37] have been used widely for complex pattern matching and recognition. It has gained popularity over the years [38] as shown in Figure 4. One of the reasons is the increased computation power and the ever-increasing amount of generated data [39].

There are multiple CNN architectures that have emerged over time like LeNet-5 [40], AlexNet [41], VGGNet [42], GoogLeNet [43] and many other. All these were trained on the ImageNet data [44] to classify high-dimensional patterns like handwritten digits. CNN also has a similar structure as a regular neural network having an input layer, a hidden layer made up of hidden units called neurons and an output layer. But it is more complex and the layers of a CNN have neurons arranged in 3 dimensions i.e. width, height, depth.

CNN process the input data through hidden layers made up of units which perform non-linear transformation of the input data in order to compute the output. The layers help to automatically detect patterns and more complex patterns can be detected with increasing layers. With the increase in the number of layers, the complexity of the network increases as well as data abstraction also increases [45]. It has become even more popular cause of the multi-use quality i.e. this deep learning method can be used to solve supervised, unsupervised or semi-supervised problems. CNN architecture and model is explained in more detail in Chapter 6.

3.3 Environment Description

This thesis is done in collaboration with Playtech plc in Tartu, Estonia. In this section, the Playtech Integrated Management System (IMS) unit and other relevant setting of the system is described. It also includes the explanation of the state of the art system in Playtech plc and the need for change to a new system for anomaly detection.

3.3.1 Playtech plc IMS unit

Playtech plc was founded in Tartu, Estonia in the year 1999 by entrepreneurs from the casino, multimedia and software engineering industries. From then onwards, it has grown significantly and has been a leading company in providing software and licenses of web and mobile application gambling gaming software to the digital gaming industry. It offers many different products e.g. Live Casino, Bingo network, iPoker network, land-based offering, Videobet, Mobile Casino etc., and has many customers.

The Playtech plc IMS [12] unit offers ancillary services such as online marketing, customer support, CRM system, fully-managed poker and bingo gaming networks, sports betting trading room services, hosting and disaster recovery services, payment processing and advisory services management tools needed by the customers to interact and manage their players throughout their life cycle. This unit serves as the backbone of the product as it provides the customers/licensees with all the tools required by them to run their businesses. IMS enables the licensees to manage their operations in an efficient and profitable way as it unifies all the products into one and provides capabilities to licensees for accessing and using the products.

3.3.2 Anomalies Detection in Playtech plc

An anomaly can be defined as an abnormal occurrence that does not conform to an expected pattern or to the normal behaviour of any system, as described earlier in Section 2.1. Playtech plc also faces anomalous behaviour and deviations in their business metrics. So, there arises a need to detect these anomalies and take necessary steps like alert the concerned team, use curative measures etc. Playtech plc used a Hewlett Packard solution which will be discussed in Section 3.3.3 and then migrated to a new approach using rules matching engine which is discussed in Section 3.3.4. As the approach using rules matching engine has certain issues, the idea is to employ deep learning to detect anomalies and determine if its a suitable solution for such scenarios.

[12] https://www.playtech.com/technology

3.3.3 State of the art in Playtech plc

Initially, Playtech plc used a solution from Hewlett Packard (HP) called Service Health analyzer (SHA) which over time did not provide very convincing results and is considered to be reaching its limit in providing even satisfactory results. SHA[13] uses the Run-Time Service Model (RTSM) and enables the user to analyze SHA anomalies using the familiar topology and metric views. SHA uses a run-time analytics engine that can anticipate IT problems before they occur, by analyzing abnormal service behaviour and alerting IT managers of real service degradation before an issue impacts the business [46].

(1) SHA, according to HP, takes data streams from multiple sources and uses predictive algorithms to alert and diagnose problems before they actually occur. **(2)** SHA uses a self-learning algorithm to analyze historical data, learn normal behaviour, and create baseline thresholds. **(3)** SHA detects an anomaly, reports on the current state of abnormal IT services and sends out an SHA event using abnormal metrics and topology information.

According to the Service Operations team experts, the **problem** with this solution is the configurations where one can not tune or change any parameters or settings. So in case of any issues, one needs to contact HP for solutions which takes a long time and with no effective results. Another problem was the increase of false alerts. The number of alerts rose upto 1,000 alerts per week, 90% of which were false alerts. This means that there were wrong event detections and sometimes even if the event occurs, no detection was done.

SHA's granularity was 15 minutes. This means that the minimum time SHA takes to detect a problem was 15 minutes. But 15 minutes is too long from a business perspective as such a delay can cause tens of thousands of revenue loss.

3.3.4 Current Alert System in Playtech plc

The architecture follows a service oriented approach and is very complex due to the complex business logic and financial transactions. Therefore, there was a need for an alert monitoring system to quickly detect and notify about the anomalies. As mentioned earlier in Section 3.2 HP Service Health analyzer was not efficient enough in terms of configuration ease, cost and reaction speed. So there was a need for an alternative monitoring and alerting system.

Monitoring a distributed system is not a trivial task. It requires a lot of expertise and knowledge to be able to monitor efficiently and effectively [47]. Low-level monitoring includes monitoring of the infrastructure, CPU, disk and memory usage, networking, etc. In this case, simple measures such as using threshold values, peak analysis can be used. But in high-level monitoring where the monitoring metrics are KPIs or business metrics such as logins, payments, transactions, it is quite challenging to have good monitoring

[13] https://support.hpe.com/hpsc/doc/public/display?docId=emr_na-c03111081/

metrics and results. The need was to develop a simple and fast solution for Playtech plc specifically, which works best and is effective for their system in particular. The main task of the new alert monitoring system were [48]: **(1)** search for any anomalies in the metrics and send out notifications if such anomalies are found. **(2)** The next task was to utilise this information received to make decisions whether these represent any real problems or not. **(3)** The alert system used a rules matching engine to do this task [48].

The Rules Matching Engine is the main component where all the rules are created, matched and any anomalies are detected. It receives data from all events. YAML (human-readable data serialization language) is used for defining the rules. An example of a rule can be *'Send an alert if one metric is decreasing'*. These rules are set using simple prefixes or suffixes for the metrics names. An abstract syntax tree is built and each violation message is checked against it [48]. If a rule matches and is triggered, an alert is sent out to notify. All the Rules Matching Engine computations are performed in memory, so the average time taken in checking one violation message against the rules is less than 1ms. This means that 1,000 messages per second can be checked. Moreover, every Rules Matching Engine's process is independent, so it is very easy to shard it [48].

Figure 5 shows the count of incidents detected in Playtech plc from 2016 to 2019 quarterly. The red color region shows the count of incidents detected by customers and the green color region shows the incidents detected by monitoring systems in Playtech plc. Once the current solution was deployed during the third quarter in 2018, the number of customer detected incidents have decreased and the number of incidents detected by monitoring systems have increased. In 2016 and 2017, SHA was used and in mid of 2018, the current solution was deployed in production.

Implementation of this solution is done in Python programming language and the metrics are stored in InfluxDB[14] database. The solution has an event-driven design and all communication is done asynchronously using message queue to achieve high throughput. Some details of the components of the system is given in Section 5.1.

This solution was an improvement from the previous solution that uses HP's SHA as can be seen in Figure 5. It provided better results and the response time was about 3 minutes (to detect anomalies and send an alert) which is quite less in comparison to 15 minutes taken by SHA. As this solution was based on rules matching, the idea was to try and use deep learning approach and see if it is a suitable approach for the same problem of anomaly detection and classification of business incidents. So, the requirement was to have a second component or Machine Learning Engine (see section 5.1) which can do the same task of decision making without using the rules matching approach and this work provides a solution for the same.

[14] https://www.influxdata.com/

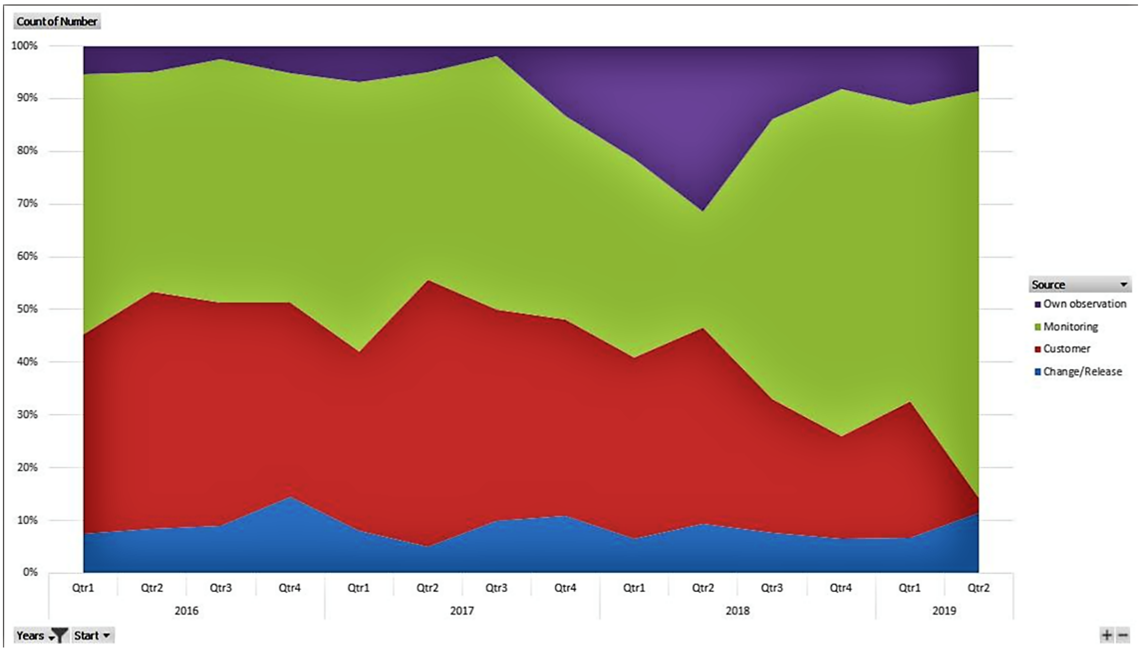


Figure 5. Count of detected business incidents quarterly (taken from Playtech plc internal IT services platform ServiceNow)

3.4 Summary

In this chapter, the necessary terms and knowledge for the understanding of this work are explained. This contributes to answering the research question "What are the current approaches and solutions that exist for anomaly detection?" (**SQ1** in Section 1.2).

In the next chapter, we proceed to create a requirement specification for the solution.

4 Requirements

This chapter describes the requirement specification for the proposed solution and answers the second sub-question (**SQ2** in Section 1.2). To answer this question better, it is broken down into the following six sub-questions: 1) What are the functional and performance requirements of this proposed solution? This chapter lists and describes the requirements.

The remaining sub-questions: 2) What is the overall design and system architecture of the proposed solution? 3) What are the data collection and transformation techniques? 4) What are the assumptions and scope of the proposed solution? are answered in 5. Chapter 6 answers the sub-question: 5) What is the method used for analysis of the fetched data? Chapter 7 answers the last sub-question: 6) How are the different components of the proposed solution implemented and how do they interact with each other?

4.1 Functional Requirements

Functional requirements include the main tasks, the actions and activities to be performed by the solution. Figure 6 illustrates the use cases of the required system. Each of the use cases[15] is mapped to a functional requirement.

- F1: An analyst shall be able to fetch the data within a given time range: The data is fetched from the database within a given time range and for particular sites. The use case UC1 is mapped to this requirement.
- F2: An analyst shall be able to transform the data: Once the data is collected, the data is pre-processed, meaning the data is cleaned and the missing values are filled. The data values are then normalized and thus transformed into features which can be used further as train and test data. The use case UC2 is mapped to this requirement.
- F3: An analyst shall be able to detect anomalies using the proposed solution: The developed solution should implement a CNN based machine learning model for the detection of anomalies and business incidents. The machine learning model shall be able to perform analysis of metrics data for multiple sites and detect business incidents. The use case UC3 is mapped to this requirement.
- F4: An analyst shall be able to classify business incidents categories: The developed solution should implement a CNN based machine learning model to categorise business incidents. The machine learning model shall be able to detect and categorise business incidents into different categories. The use case UC4 is mapped to this requirement.

[15] The use cases are defined using use case textual templates (Tables 9-14 in the Appendix I)

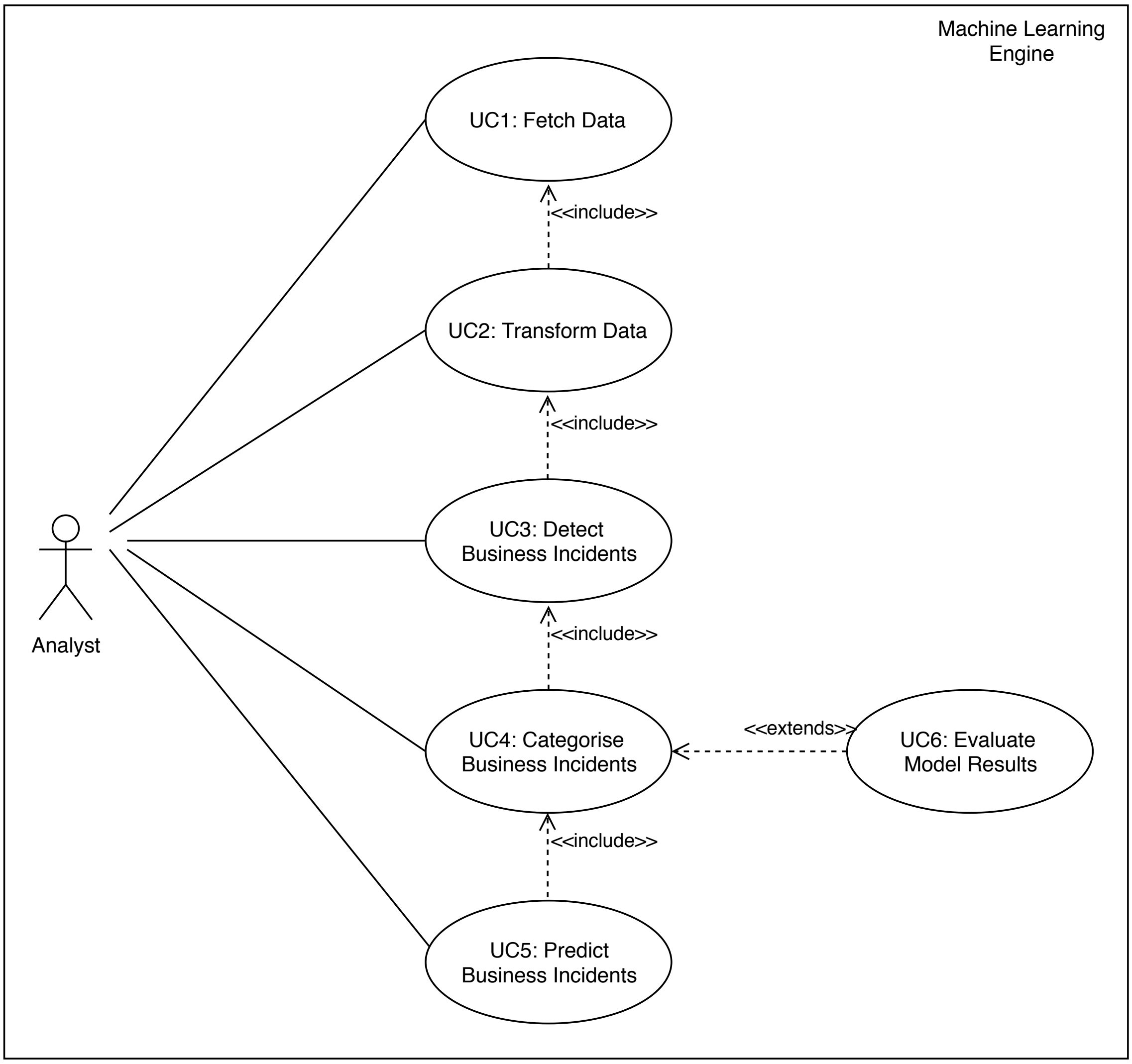

Machine Learning Engine
UC1: Fetch Data
<<include>>
UC2: Transform Data
<<include>>
UC3: Detect Business Incidents
<<include>>
Analyst
UC4: Categorise Business Incidents
<<extends>>
UC6: Evaluate Model Results
<<include>>
UC5: Predict Business Incidents

Figure 6. High-level use case diagram of the proposed solution (use case description templates in Appendix II)

- F5: An analyst shall be able to predict business incident categories: The developed solution should implement a CNN based machine learning model to categorise and predict business incidents. The machine learning model shall be able to predict business incidents into different categories based on probability values. The use case UC5 is mapped to this requirement.

- F6: An analyst shall be able to evaluate the model results: The developed solution has a CNN based machine learning model which is used to categorise and predict business incidents. The results from this model i.e. the categorised incidents should be evaluated. The use case UC6 is mapped to this requirement.

It is important to mention that the solution, which detects anomalies, categorises and predicts business incidents, is to be used in conjugation with the current monitoring and alert system in Playtech plc.

4.2 Performance Requirements

Performance requirement specifies the extent to which the functional requirements must be executed and measured in terms of quantity and quality.

- P1: Memory: The developed solution should be efficient in terms of memory to have practical use and so that it could be adopted by Playtech plc. The memory requirement is set based on the data size and is less than 6GB. The typical machine learning phases like training the model must be supported seamlessly.

- P2: Computation Speed and Efficiency: The time taken for data collection, preprocessing and training the proposed CNN based classifier should be less than 10 hours. The developed solution must yield optimized results and this should be measured by metrics like accuracy, precision, recall and f1-measure.

- P3: Extensibility: The proposed solution should allow the addition of new statistical and machine learning algorithms and be upgradable to multiple sites when required. The system shall be built as a set of loosely coupled individual components which makes it possible to extend the system and add features independently without the problem of breaking something.

Security is not the focus of this work but certain aspects of security are taken into consideration while designing the proposed solution such as data confidentiality. Only the authorized user can read and process the intended data (in this case Playtech plc). A secure connection to the database (DB) is established for data collection.

The traceability between requirements and design of the proposed solution is given in Appendix II.

4.3 Summary

In this chapter, the requirements for the proposed solution are gathered and a requirement specification is created. This contributes in answering the research question "How to design and develop a viable solution for the detection of anomalies and categorisation of business incidents?" (**SQ2** in Section 1.2).

After gathering the requirements, in the next chapter, we proceed to discuss design of the proposed solution.

5 Proposed Solution Design

This chapter describes the architecture of the proposed solution and contributes in answering the second sub-question (**SQ2** in Section 1.2). The following sub-questions are answered in this chapter: 2) What is the overall design and system architecture of the proposed solution? 3) What are the data collection and transformation techniques? 4) What are the assumptions and scope of the proposed solution?

Section 5.1 discusses the system architecture in brief and then the individual components of the proposed solution are explained. Section 5.2 describes the data fetching and feature engineering mechanism. The assumptions and scope of the proposed solution are mentioned in Section 5.3.

5.1 Architecture Overview

This section describes the system architecture of the proposed solution. It details the individual components/modules of the proposed system such as the data fetcher and preparation modules, Analysis module.

The proposed solution consists of a Machine Learning Engine. This engine uses a CNN based model for the detection of anomalies and categorisation of business incidents []. The data comprises of various business metrics such as user logins, payments, transaction amounts which is fetched from the InfluxDB. This engine collects and transforms the data, perform certain analytical computations, categorises the business incidents and predicts probabilities for business incidents. In this thesis, a CNN based model is used to categorise the business incidents and the model is evaluated using metrics such as accuracy, precision.

Figure 7 shows the system architecture of the Playtech plc monitoring and alert system (consisting of Rules Matching Engine and other related components) along with the proposed Machine Learning Engine.

The solution is designed as a set of loosely coupled components. The communication between system components is done using a message queue and in an asynchronous way to achieve higher throughput. In this case, ActiveMQ[16] is used but any message queue following Simple Text Oriented Messaging Protocol (STOMP) can be used [48].

Python programming language, with its various data analysis and computation libraries such as Pandas[17], NumPy[18], Scikit-learn[19], is chosen for implementation. The business metrics are stored in InfluxDB as it is quite fast and reliable for time series [49].

[16] http://activemq.apache.org/

[17] https://pandas.pydata.org/

[18] http://www.numpy.org/

[19] https://scikit-learn.org/stable/

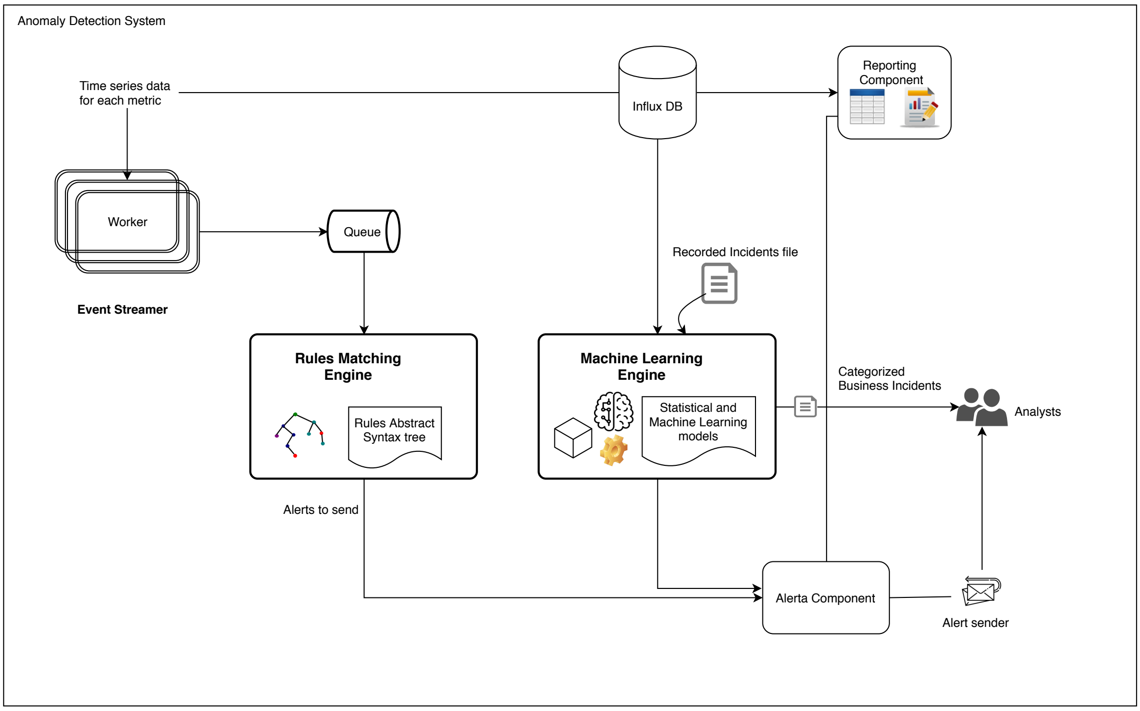


Figure 7. High level system architecture (adapted from [48])

The Event Streamer, Worker, Rules Matching Engine and Reporting components (although not part of the proposed solution) are discussed in brief as they are necessary for the understanding of the current system in Playtech plc and the information flow.

Worker: It is the main working unit that has a set of models along with meta-data. The worker performs I/O operations in the InfluxDB. It has a handler and a data connector [48]. The handler receives data from the data connector and tests it against a set of models using a specific strategy.

Event Streamer: This component has all the workers and fetches data regularly.

Rules Engine/Rules Matching Engine: This component receives the information provided by the Event Streamer as input. This is done by subscribing to a specific topic in ActiveMQ. The Rules Matching Engine is subscribed to a specific topic and every violation message is added to the tree of dictionaries that is associated with every site (a tag is used in the InfluxDB) [48]. The system stores all the events received within a specific time range. After new messages are received, the previous old events are deleted.

Every message is sent to the Rules Matching Engine where the analysis is done.

YAML Ain't Markup Language (YAML), a human readable data serialization language, is used for defining the rules. A rule can be any metrics based decision (For ex: 'An alert should be sent, if a particular metric is decreasing and another one is increasing'). The rules are set by using regular expressions or prefixes for the names of the metrics. At the start of the Rules Matching Engine, an abstract syntax tree is created and every violation message is checked against this tree. If a rule is matched, an alert will be sent [48]. There is a possibility that an event triggers more than one rule. In that case, the rule hierarchy defines which rule is more important and what alert should be sent.

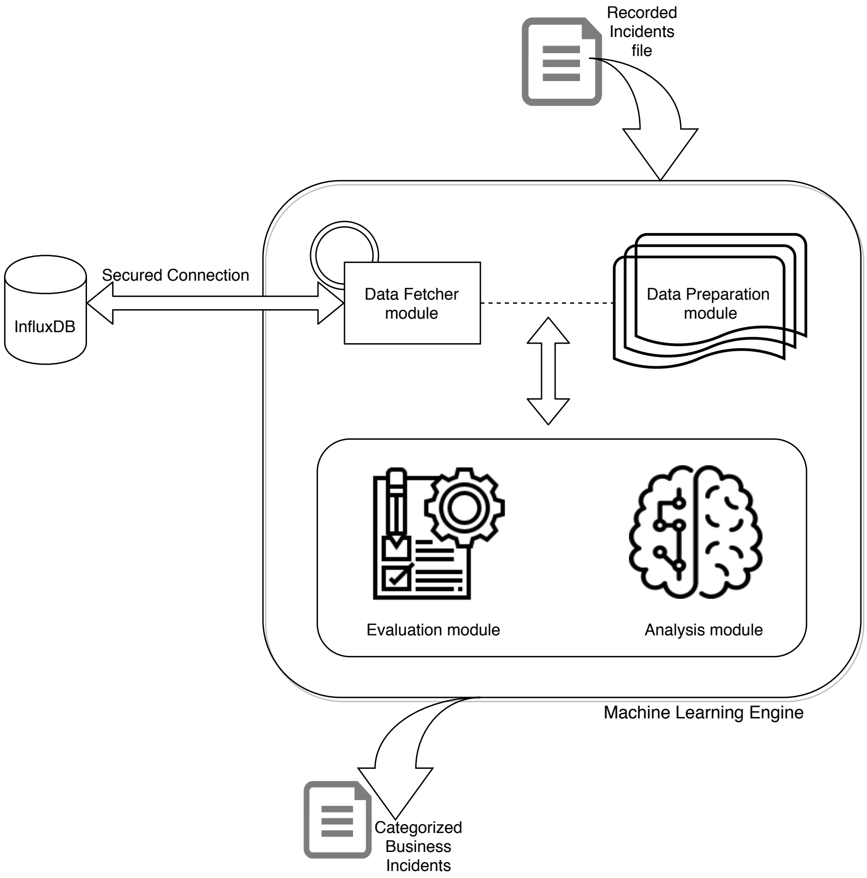


Figure 8. Block diagram of the proposed solution

Here, the speed and acceleration of the degradation of the metrics with respect to time is important, as it shows the severity of the incident. Speed is a discrete derivative or an angular coefficient of a particular metric which is calculated for every violation. Acceleration is the second order derivative. The speed and acceleration values are set in the rules and used in the overall assessment of an incident [48].

The average time of checking one violation message against the rules is around 1 ms for the configuration [48]. This means that more than 1,000 messages per second can be checked.

Reporting: This component creates dynamic reports which record all the alerts and all the metrics that are related to the rules. A report link is generated, when an alert is fired by the Rules Matching Engine [48].
As the rules matching engine had certain disadvantages for ex: creating a new rule would require code changes. This led to the requirement for another Machine Learning Engine where the detection and categorisation could be done automatically.

Machine Learning Engine: The Machine Learning Engine is the main focus of the design and implementation of the proposed solution. Figure 8 depicts the block diagram of the Machine Learning Engine. It consists of four main modules: 1. *Data Fetcher module*, 2. *Data Preparation module*, 3. *Analysis module* and 4. *Evaluation module*.

1. **Data Fetcher module:** This is the module that fetches the data from the InfluxDB through an established secure connection as shown in Figure8. The InfluxDB is used as it is a fast access database and is quite reliable [49]. The tasks performed by this module are explained in Subsection 5.2.2 and the implementation details are given in Chapter 7.

2. **Data Preparation module:** This module, as the name suggests, is responsible for performing the pre-processing and transformation on the fetched data. There are three main tasks performed in the module. The first task is to clean the fetched data and fill any missing values. The missing values are replaced by 1 to avoid division by zero error during the next normalization step. The next task is to normalize the data and make sure that all the data values are in between the range of 0 and 1. The last task is to slice the normalized data into small windows which can be used as input to the model. The tasks performed by this module are explained in Subsection 5.2.3 and the implementation details are given in Chapter 7.

3. **Analysis module:** This module comprises of the CNN based model. This model will be discussed in detail later in Chapters 6 and 7. The CNN based model is used to detect, classify and predict business incidents. For this a *recorded incidents file* is also used as shown in Figure8, which assists in implementing the supervised learning model.

4. **Evaluation Module:** This module, as the name suggests, is used to evaluate the model implemented by using various evaluation metrics such as accuracy, precision, recall. The results of the evaluation are provided in Section 8.2.

5.2 Data Collection and Pre-processing

In this section, the method to collect and pre-process data is described. the Subsections 5.2.1, 5.2.2 and 5.2.3 include the details about the time series database (TSDB) used and the data flow during the fetching and pre-processing stages. The business metrics data is collected using multiple sources which retrieve data and feed to the DB as well as the event streamer and workers. The focus of this section is data fetching from the DB as the other sources and event streamer is out of scope for this work.

5.2.1 Data Structural Property

The data is business metrics data. It consists of metrics for each site or licensee such as number of login and sign-ups, number of payments and transactions carried out, various casino gaming win and bet counts etc. It is organized such that for each metric say *casino gaming*, there are many aggregate values such as win counts, bet counts, games count etc. Another metric *payments* has aggregate values such as number of withdraw requests, withdraw count, deposits count and so on.

TSDB is used to store time series data efficiently. It is a database that is optimized for handling time series data or any serial data indexed by time [50]. The efficiency considerably improves based on the fact that time is considered as a discrete quantity rather than a continuous one. A TSDB enables to insert, read, update and delete time series data easily and helps to organize it better than relational DB.

As mentioned before, InfluxDB is used for managing the time series business data available in Playtech plc. It is a cross platform, open source and MIT licensed TSDB developed by InfluxData [20]. InfluxDB is specifically optimized for fast retrieval of time series data. It promises a high-availability storage option and is popular in the fields of operations monitoring which suits the needs of this solution [50]. It is also a good option as database for real-time analytics and fast processing operations.

InfluxDB provides a SQL-like language called InfluxQL with built-in time-centric functions for querying a data structure composed of measurements or series or data points [51]. Each point consists of several key-value pairs called the fieldset and an associated timestamp. These are grouped together to define a series, as a set of key-value pairs called the *tagset*. These series can be grouped together by a string identifier to form a *measurement*. The points are indexed by their associated timestamp and tagset. The queries run periodically and results are stored in a target measurement.

An example of a tagset, in a measurement, is as follows: A tagset comprises of a root tag and an aggregate tag. The root tag has the *site* (licensee name) name and say the aggregate tag consists of an *operation*. This operation can have metric values related to *user logins, casino gaming metrics, payments metrics* etc. An example for value of the aggregate tag (termed as aggregate value in this solution) can be *failed login count*.

[20] https://docs.influxdata.com/influxdb/v1.7/

5.2.2 Data Fetcher module

The business metrics are collected regularly from multiple sources and stored in the DB. This is done by specific data fetching operations which collect business metrics after every 1,800 seconds. The details of this business metrics collection is out of scope of this thesis.

The *Data Fetcher module* fetches the business metrics data stored in InfluxDB for individual sites. A list of available sites (licensees) is maintained and the data is fetched within a given time range. For this, a secure connection to the database is established by specifying the database name, username, password and other parameters which is discussed in detail in the implementation Section 7.2.1. Once the secure connection is established, the measurements of the business metrics are retrieved.

Each measurement consists of various business metrics which are retrieved and read as follows: The root tag value is the name of the *site*. The aggregate tag consists of the metrics name *casino_gaming, logins* and their values. The aggregation list comprises of *casino, clientplatform* and *operation*. The metric name, operation name and aggregate value combines to form the aggregate tag. For each site, there can be multiple metrics. Therefore, the dictionary *sites_metrics* stores all the site names as keys and the metrics as values. Once the aggregate tag is available, it is used to query and retrieve results from the metric table by providing the site name. The results are retrieved in a list. For each individual query, a 1,000 records dataframe with a unique field/column name containing aggregate tag and metric values are retrieved. The data is collected for a specific time range using the start and end date.

Data Normalization: Normalization is done to make the data uniform so that it can be used as input to the CNN model. The data is subsetted within a given time range and normalized. Normalization is used to adjust the data so that it is approximately of the same scale. Here, the data is rescaled to have values between 0 and 1. This can be achieved by the equation 1.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Once the data is normalized, the subsets are combined to form a single data frame. This data frame, with multiple fields/columns specifying the different metrics values, for a particular site is then returned to the *Data Preparation module*.

5.2.3 Data Preparation module

Once the data frame is retrieved from the Data Fetcher module, the *Data Preparation module* performs certain transformations and convert it into a suitable form for input to the CNN model. As stated in the Section 5.2.2, on retrieving the subset of the data within a given time range, the data is normalized and the maximum values are set to be 1.

Filling Missing Values: Another important operation is to fill the missing values with some common value or a mean/median value. The value chosen here was 1 to avoid division by zero error. Using the mean value doesn't quite provide any advantage as the data being used is already normalized.

Data Slicing: The next task is to *slice data* and create different matrices of data. The data subset or slice with 5 rows and 12 columns is used. Each of these slices' rows is traversed and the timestamps are matched against the incident times from the recorded incidents file. The *incident flag* is set to True or False accordingly. Furthermore, the sliding window method is used. A sliding window is a sub-list that runs over an underlying collection. For ex: for a list of size 7 [A, B, C, D, E, F, G], a sliding window of size 3 will have elements like [A, B, C], [B, C, D],..., [E, F, G]. Following the same approach, a window with parameters *window_height* and *window_width* containing data points is created and then the window is moved one step at a time for each step of iteration, until the end of data points is reached.

The window size parameters are modifiable according to the requirements. In this solution, a window size of 5x5 meaning *window_height* = 5 and *window_width* = 5 is chosen for the transposed slice of data. These window slices of size 5x5 are then converted into a matrix/2-D array which has values between 0 and 1. The *incident flag* is again checked and the slice matrix is added to the *incident marked list* or *non-incident marked list* based on the value of *incident flag*. Thus, using the above mentioned approach, the data is sliced into smaller matrices using a 5x5 window and checked for any incidents or anomalies. Finally once all slice matrices are segregated, they are labeled using '1' for the incidents and '0' for non incidents. The final *result list* is combined from the two lists: the *incident marked list* and the *non-incident marked list*.

5.3 Assumptions and Scope

The following assumptions are made while designing and developing the proposed solution:

- It is assumed that the historical data of business metrics and events are already present as a time series and stored in a database.
- In order to avoid memory errors during the training of CNN model on a single machine, it is decided to set the maximum data size to be trained around 400 MB. Although, the final implemented solution in production is to use large dataset i.e. in orders of GBs.

- It is assumed that the training dataset is balanced i.e. count of incidents and non-incident event is similar so as to avoid the classifier being biased towards one.
- It is assumed that the proposed solution will exist in conjugation with the rules matching engine which already exists as a current solution in Playtech plc.

The other supporting components like metrics collection component and DB management is out of scope of this thesis. The dashboards to visualize the alerts are not included in this work as well.

The main focus of this work is only on designing and developing a solution for data pre-processing, implementing a CNN based machine learning model and evaluation of this model.

5.4 Summary

In this chapter, the architectural design of the proposed solution, the data collection and data transformation methods are described. This contributes in answering the research question "How to design and develop a viable solution for the detection of anomalies and categorisation of business incidents?" (**SQ2** in Section 1.2). The assumptions and scope of this work are also mentioned here.

6 Analysis Module

This chapter describes the CNN based model and contributes in answering the second sub-question (**SQ2** in Section 1.2). The question: 5) What is the method used for analysis of the fetched data? is answered in this chapter. To analyse the data, a deep learning approach is followed which uses Convolutional Neural Network.

6.1 Convolutional Neural Network (Deep Learning Approach)

The idea was to use a deep learning approach for the task. A Convolutional Neural Network is chosen as it is best known for its ability to recognize patterns. This suits the main task in the proposed solution that is to detect reoccurring patterns and/or anomalies. In this chapter, the design of CNN, the types of layers and the different hyperparameters that can be used are discussed. It also includes discussion about layers and parameters relative to the proposed solution's CNN model. Chapter 7 describes the implemented CNN model.

CNN or Convnet is a sequence of layers where every layer transforms one volume of activations to another through a differentiable function [52]. Classification using CNN includes many steps and several layers. It has multiple hidden layers which perform a set of mathematical operations for extracting low-level features through applying convolutional filters, activation functions, max-pooling operation, resizing the layer, transforming layer to fully-connected layer. The neurons inside the hidden layers use weights on inputs to produce the output. Figure 9 shows an example of a CNN.

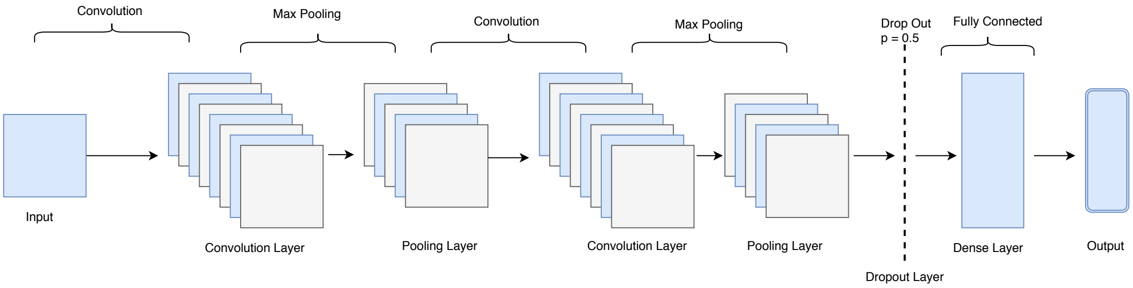


Figure 9. An example of a convolutional neural network (adapted from [40])

The Analysis module (as mentioned in 5.1) comprises of the CNN based model. Although CNNs can have various network architectures, they contain similar components which are described below:

Convolutional/Conv Layer: It is the core building block of a CNN that does most of the computations. A 2-D convolutional layer is used which applies sliding convolutional filters to the input i.e. applying matrix filter. The computations include dot multiplication between two matrices and sum up the received values [53]. One of these matrices is the set of learnable parameters known as a kernel and the second matrix is the receptive field's values. The size of the Convolution kernel is much smaller than the input image [53]. Figure 10 shows an example of convolution on input matrix data using a 3x3 kernel matrix/filter.

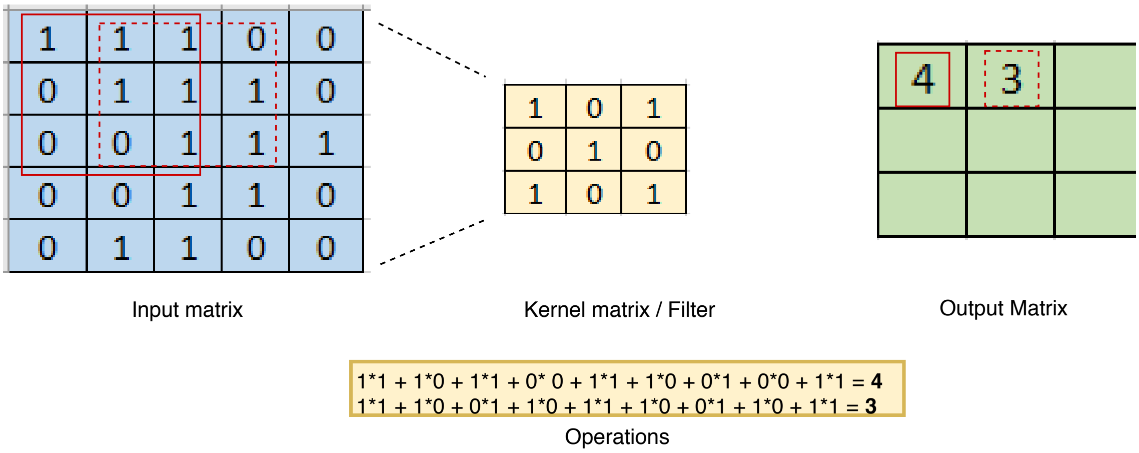
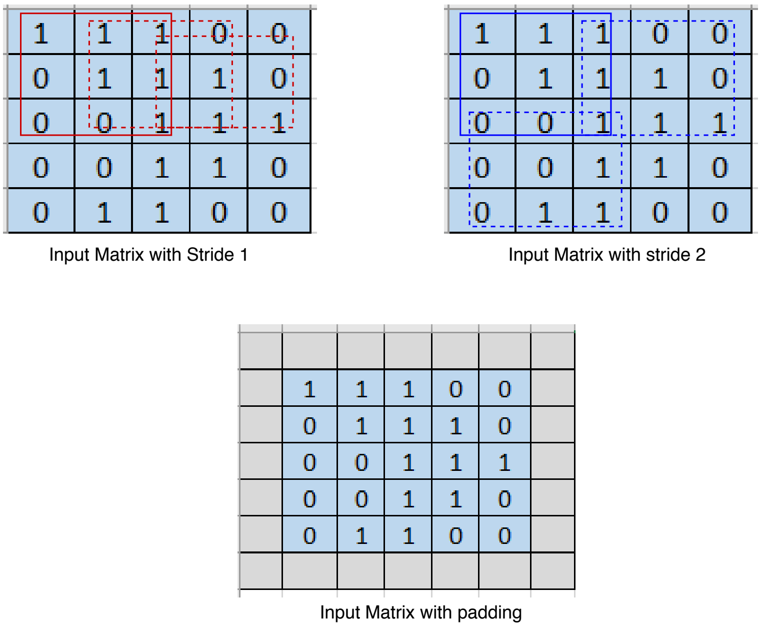


Figure 10. An example of convolution on input matrix using a 3x3 filter (adapted from [54])

The first convolutional layers in the networks usually detect low-level features such as lines, edges and curves [55]. Once the network becomes deeper through max-pooling operations and convolutional layers, they can detect more complex features. For every convolution filter, we must specify the *filter size* that is how many pixels are used in a filter. A filter is any algorithm that starts with some image I(x, y) and computes a new image I′(x, y) by computing for each pixel location x, y. The template that defines both this small area's shape, as well as how the elements of that small area are combined, is called a filter or a kernel [56].

Stride and Padding: Stride specifies the number of pixels by which the convolution filter is shifted at each step during convolution [54]. The default value is one. As we increase the stride, the output feature map reduces in size as some potential locations are skipped in this process. In order to maintain the same size/dimensionality, padding is used. As the filters are square and if there is no padding, the filter might be outside the input matrix, so additional padding is used which extends the margins of input matrix and fills those margins with zero-values [54]. This helps to preserve the size of the feature maps or else they would shrink at each layer. Figure 11 shows the strides and padding techniques.



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input Matrix with Stride 1

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input Matrix with stride 2

	1	1	1	0	0	
	0	1	1	1	0	
	0	0	1	1	1	
	0	0	1	1	0	
	0	1	1	0	0	

Input Matrix with padding

Figure 11. Strides and padding (adapted from [54])

Activation Functions: Activation functions are applied after Conv layer to transform/modify the output values in desired form [53]. The layer with activation function maps the resulting values depending on the activation function used. Figure 12 shows the different activation functions. The sigmoid function curve takes a S-shape. The sigmoid function is used for models where we have to predict the probability as an output and as probabilities can be between 0 and 1, the function also ranges between 0 to 1. The softmax function is a logistic activation function that is used for multi-class classification. The tanh function is also like the logistic sigmoid but the range is from (-1 to 1). The ReLU is half rectified i.e. f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero. The range is [0 to infinity). All the negative values are transformed to zero. It is applied to the output of the Conv layer to extract linear and non-linear relations in the data.

In this work, ReLu activation function is used for convolutional layers and sigmoid function is used to categorise whether a business incident is detected or not.

Pooling Layer: This layer is inserted between successive convolutional layers [53]. It is applied to the output from the ReLU to extract the most meaningful information. It progressively reduces the spatial size of the representation and thus reduces the amount

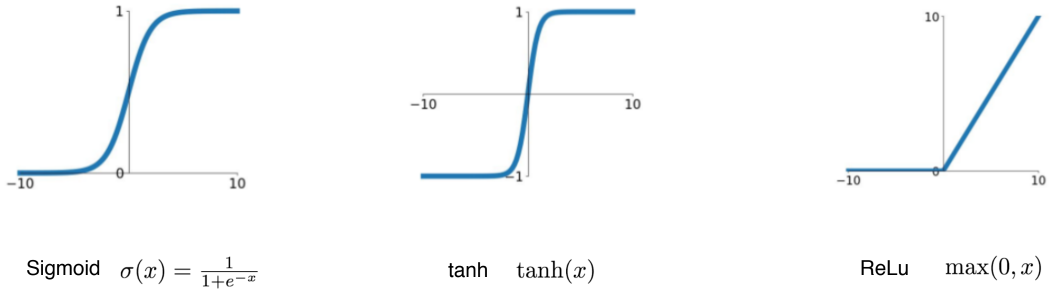


Figure 12. Different activation functions (adapted from [53])

of parameters and computation in the network. This helps in controlling overfitting [52].

Fully connected Layer: This layer acts as a classifier[53]. In this layer, each neuron is connected to every neuron in the previous layer, and each connection has its own weight [52]. The connection pattern makes no assumptions about the features in the data. It is very expensive in terms of memory (weights) and computation (connections). The convolutional layers provide a meaningful, low-dimensional space and the fully-connected layer learns a function in that space.

Dropout Layer: This layer randomly sets the input elements to zero with a given probability. It deactivates a certain percentage of neurons for the specific layer during the training. It is used to prevent overfitting in the network [55].

Output Layer/Classification Layer: This layer computes the cross entropy loss for multi class classification with mutually exclusive classes. It uses softmax or sigmoid functions.

In this work, a sequential model is created using Keras[21] and convolution layers are added. There are two convolutional layers in the model. Both layers have the same number of hidden units i.e. 64, the ReLu activation function and a default stride value of 1x1. The number of hidden units were chosen to be 64 as it is between the size of the input layer and size of output layer. Other option e.g. 128 was also tried. It did not have any significant impact on the final model results instead increased the training time. So, it was decided to use 64 hidden units to avoid underfitting as well as overfitting and to reduce the training time.

The filter size in both the layers is different. It is 4x4 and 3x3 in the first and second convolutional layer respectively. The idea is based on the fact that as the patterns are small, it is recommended to use small filters relative to the input matrix size (5x5). As

[21] https://keras.io/

the incident list consists of slice arrays of size 5x5, kernel size was chosen to be 4x4 and 3x3.

Two pooling layers are used with the same pool size 2x2 and stride 1x1. For the fully connected layer, 32 hidden units and the ReLu activation function is used. The dropout is set to 25% to avoid overfitting. The batch size is kept at 64. Although other batch sizes e.g. 32 were also tested but in this case, 64 suits the best. The parameter values along with other implementation details will be discussed in 7.2.2.

6.2 Summary

In this chapter, the CNN model used for detection and categorisation of business incidents, used in the *Analysis module* as part of the proposed solution, is explained. This contributes in answering the research question "How to design and develop a viable solution for the detection of anomalies and categorisation of business incidents?" (**SQ2** in Section 1.2).

7 Proposed Solution Implementation

This chapter describes the implementation of the proposed solution and contributes in answering the second sub-question (**SQ2** defined in Section 1.2). This chapter answers the question: 6) How are the different components of the proposed solution implemented and how do they interact with each other? Section 7.1 explains the interaction between the individual components of the proposed solution. Section 7.2 gives the implementation details. Section 7.2.1 describes the data pre-processing along with code snippets from the implemented solution. Section 7.2.2 describes how the CNN model is implemented along with code snippets.

The proposed solution is implemented in Python programming language and uses several other open source libraries e.g. Pandas library is used for data pre-processing and transformation. The deep learning library Keras is used for the implementation of CNN based model.

7.1 Component Interaction

In this section, the different components of the proposed solution and their interaction among each other are discussed. The data and information flow among different modules is also described. Figure 13 and Figure 14 depicts the working of various involved modules and components. Figure 15 shows a detailed view of the relationship between the modules. The work of the Data Fetcher and Data Preparation modules is separated from the Analysis module and Evaluation module for modularity, easier maintainability and understanding.

Data Fetcher and Data Preparation module: Figure 13 illustrates the data collection and transformation process. The main function initiates the data preparation phase. The *Data Fetcher module* initiates the connection to the DB by providing the username, password, URL and other necessary arguments over secure connection. Once a connection is made to the DB, it performs a select operation to fetch data for a particular business site and returns the result set to the *Data Preparation module*. The data comprises of various business metrics like login counts, games count, win counts, bet counts, payments, withdraw count, closed fund transfers etc. The fetched result set data is then cleaned. This operation is performed iteratively until data for all sites is fetched and cleaned. Afterwards, when the data from all sites is fetched, the result dataset is passed to the *Data Preparation module*.

The *Data Preparation module* now initiates a call to the Incident Data Manager and receives the *recorded incidents file*. This file contains the historical list of incidents, along with time values, which is used to generate labels. Once the recorded incidents are received, the *Data Preparation module* uses this recorded incidents data as well as the cleaned fetched data and perform certain transformations which is described later. Data

Slicing operation (as described in 5.2.3) is performed and the incident times from this *recorded incidents file* is used to label the data. This data is passed to the main function as a combined list of incidents and non-incidents data, with normalized metrics values and the associated labels.

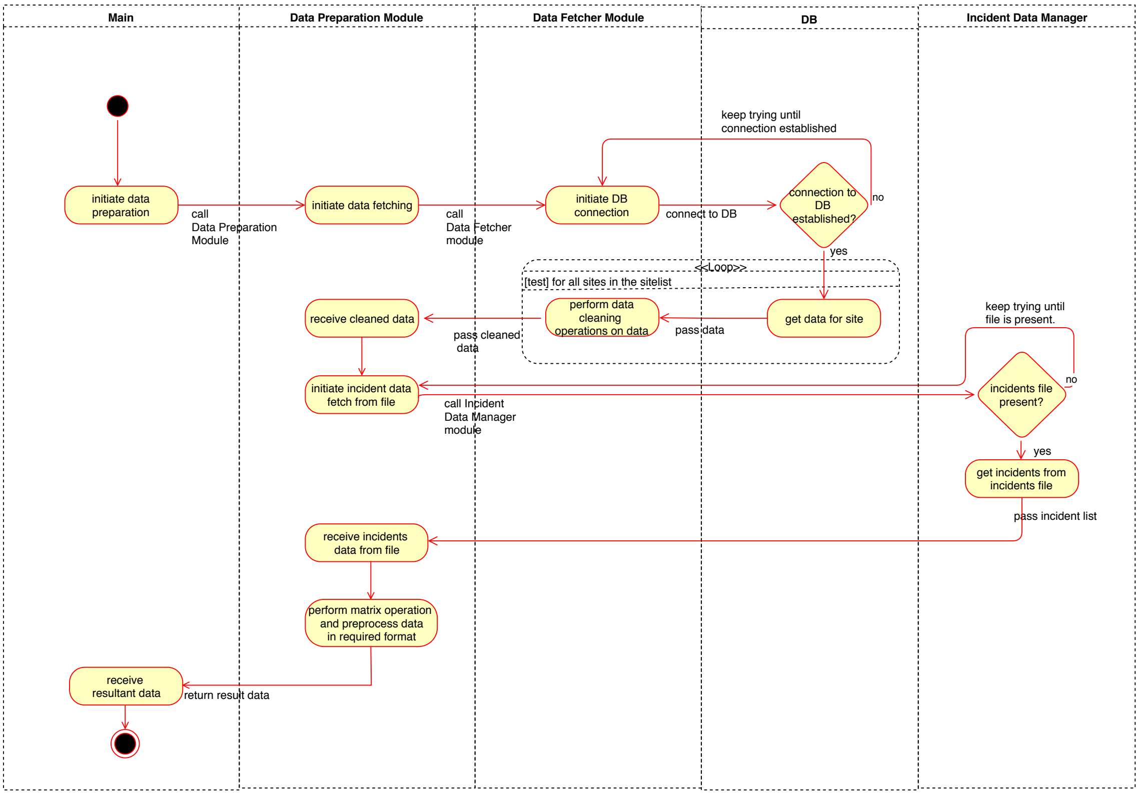

Main
Data Preparation Module
Data Fetcher Module
DB
Incident Data Manager
initiate data preparation
call Data Preparation Module
initiate data fetching
call Data Fetcher module
initiate DB connection
connect to DB
keep trying until connection established
connection to DB established?
no
yes
<<Loop>>
[test] for all sites in the sitelist
get data for site
pass data
perform data cleaning operations on data
pass cleaned data
receive cleaned data
initiate incident data fetch from file
call Incident Data Manager module
keep trying until file is present.
incidents file present?
no
yes
get incidents from incidents file
pass incident list
receive incidents data from file
perform matrix operation and preprocess data in required format
return result data
receive resultant data

Figure 13. Data fetcher and data preparation modules

Analysis and Evaluation Module: Figure 14 illustrates the Analysis module and Evaluation module. It shows the information flow between these two modules. The analysis of data is initiated by calling the *Analysis module*. This module comprises of the CNN based model. The earlier cleaned and transformed data received from the *Data Preparation module* is shown in Figure 13 fed to the CNN model. The business incidents are then detected, categorised and predictions are made. The results, including the predicted values, from this model is returned to the main function in the form of lists. These results are then passed on to the *Evaluation module* for evaluation. The evaluation of the model is done on the basis of metrics such as precision, recall, f-measure, accuracy and finally the evaluation results are passed to the main function.

Figure 15 shows the detailed structure of the proposed solution. The diagram although does not show each and every functions, yet the most important ones which are required

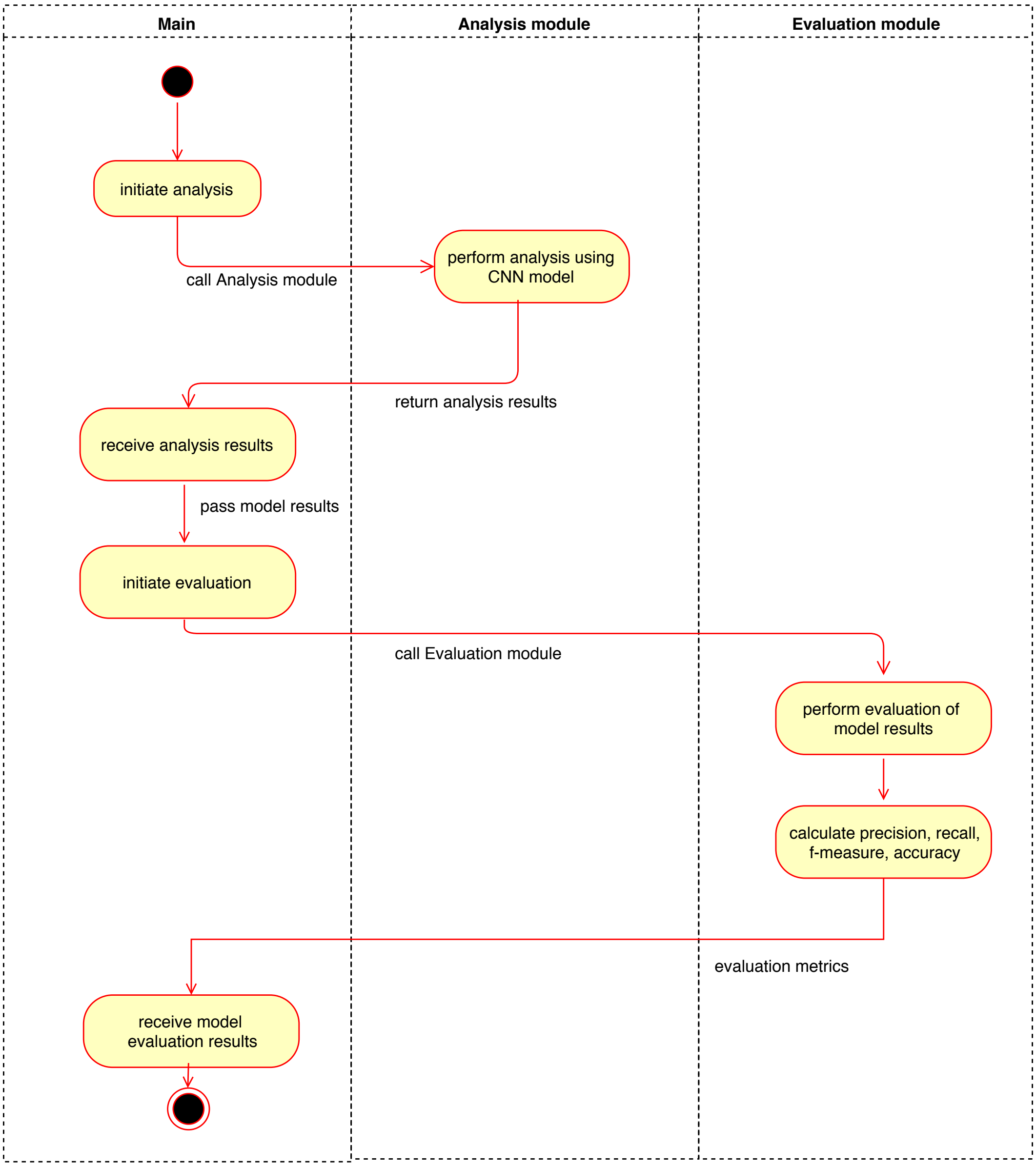


Figure 14. Analysis and evaluation modules

to understand the overall information flow are shown.

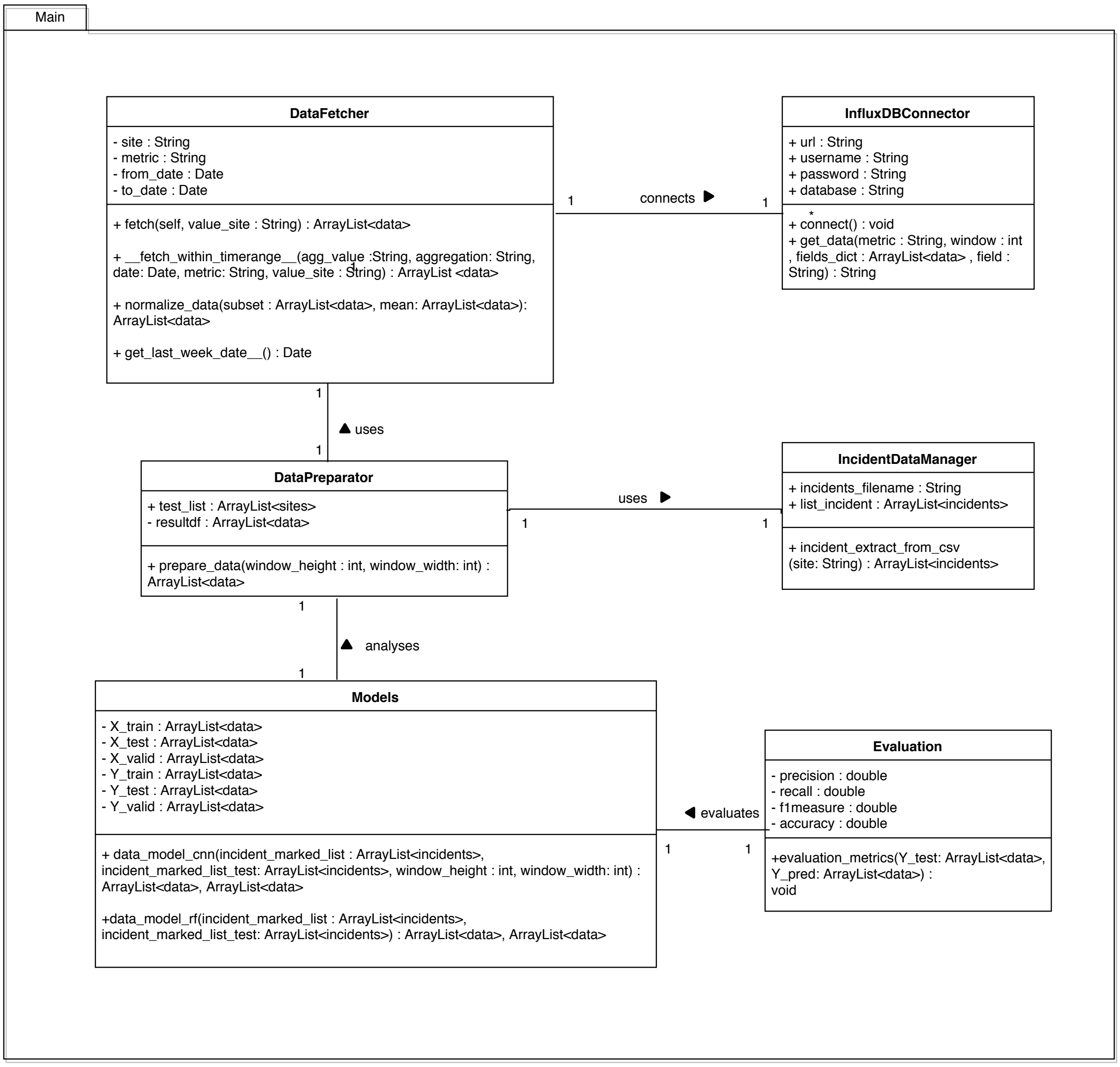

Main
DataFetcher
- site : String
- metric : String
- from_date : Date
- to_date : Date
+ fetch(self, value_site : String) : ArrayList<data>
+ __fetch_within_timerange__(agg_value :String, aggregation: String, date: Date, metric: String, value_site : String) : ArrayList <data>
+ normalize_data(subset : ArrayList<data>, mean: ArrayList<data>): ArrayList<data>
+ get_last_week_date__() : Date
1
connects ▶
1
InfluxDBConnector
+ url : String
+ username : String
+ password : String
+ database : String
+ connect() : void
+ get_data(metric : String, window : int , fields_dict : ArrayList<data> , field : String) : String
1
▲ uses
1
DataPreparator
+ test_list : ArrayList<sites>
- resultdf : ArrayList<data>
+ prepare_data(window_height : int, window_width: int) : ArrayList<data>
uses ▶
1
1
IncidentDataManager
+ incidents_filename : String
+ list_incident : ArrayList<incidents>
+ incident_extract_from_csv (site: String) : ArrayList<incidents>
1
▲ analyses
1
Models
- X_train : ArrayList<data>
- X_test : ArrayList<data>
- X_valid : ArrayList<data>
- Y_train : ArrayList<data>
- Y_test : ArrayList<data>
- Y_valid : ArrayList<data>
+ data_model_cnn(incident_marked_list : ArrayList<incidents>, incident_marked_list_test: ArrayList<incidents>, window_height : int, window_width: int) : ArrayList<data>, ArrayList<data>
+data_model_rf(incident_marked_list : ArrayList<incidents>, incident_marked_list_test: ArrayList<incidents>) : ArrayList<data>, ArrayList<data>
◀ evaluates
1
1
Evaluation
- precision : double
- recall : double
- f1measure : double
- accuracy : double
+evaluation_metrics(Y_test: ArrayList<data>, Y_pred: ArrayList<data>) : void

Figure 15. Low level diagram of the proposed solution

The main classes which correspond to the main modules of the proposed solution are shown in Figure 15. It consists of a *DataPreparator* class which uses the *DataFetcher* class to collect the data. The DataFetcher class connects to the database using the *InfluxDBConnector* class in order to fetch the data. The Data Preparator class also uses the *IncidentDataManager* class to import recorded incidents file. The DataPreparator class transforms the fetched data and then passes the prepared data lists to the *Models* class (*Analysis module*) which performs the analysis on the data and categorises and predicts the business incidents. Lastly, the *Evaluation* class performs evaluation of the categorised business incidents.

7.2 Implementation Details

In this section, the implementations details of the different modules and the main function are described along with code snippets. The data pre-processing implementation is detailed in 7.2.1 and the CNN model is described in 7.2.2.
Each of the four modules in the proposed solution, as described in Section 5.1, is implemented as a separate package in an object-oriented programming paradigm to have separate functional responsibility. The different packages are then tied together with a main file.

The 'main.py' file contains the main script which controls all the modules and manages the information flow between different modules. A code fragment from this file is shown in listing 1. Here, the following operations are performed:

- Specify the site (licensee) names for which the business metrics data should be collected in *test_list*.
- The parameters *window_width* and *window_height* are used to define the window size for input data matrix in the CNN model.
- The IncidentDataManager class ('incident_management.py') creates a list of already occurred incidents using the incident_time_manager object. These incidents are read from a csv file which has manually recorded incidents during the period of March and April 2018.
- The fetcher object of class DataFetcher (file 'fetcher.py') is passed to the DataPreparator class.
- The data_preparator object of the DataPreparator class (file 'data_preparator.py') is initialized with the *test_list* sites, the fetcher object and the incident_time_manager.

7.2.1 Data collection and pre-processing Implementation:

As we discussed data pre-processing in Section 5.2, here the implementation details are given for the same including code snippets for illustration.

Data Fetcher module:
In the DataFetcher class, the important method is *fetch()*. The *influx_connector* is used to connect to the InfluxDB and retrieve the business metrics data. The following steps are performed in this method.

```
incident_time_manager = IncidentDataManager(incidents_filename)

fetcher = DataFetcher(from_date, to_date)
data_preparator = DataPreparator(test_list, fetcher,
incident_time_manager, type = 'train_')

result_list = data_preparator.prepare_data
(WINDOW_HEIGHT, WINDOW_WIDTH)

data_preparator = DataPreparator(test_list, fetcher,
incident_time_manager, type = 'test_')

result_list_test = data_preparator.prepare_data
(WINDOW_HEIGHT, WINDOW_WIDTH)

dm = Models(result_list, result_list_test)
Y_test, Y_pred = dm.data_model_CNN(result_list,
result_list_test, WINDOW_HEIGHT, WINDOW_WIDTH)

eval = Evaluation(eval_metrics=None)
eval.evaluation_metrics((Y_test[:,0]), Y_pred)
```

Listing 1. Code snippet from the file 'main.py'

- The first step is to check if the *site_metrics*'s value i.e. the list of *metric, aggregation, agg_value* is already present in *site_metrics* dictionary or not. If not, an empty list is added to this value. Next for all aggregation values, from the *aggregation_list* i.e. the aggregate tags like *casino, operation*, the following steps 2 - 4 are followed.

- For each *agg_value* in *aggregation_values_unique_list*, we append the *sites_metrics* dictionary values and increment the count of *all_metrics*.

- Next step is to fetch the data within a specific time range. For ex: if we want to get values between last week and current date, we specify the last week date using the *__get_last_week_date__()* method and *to_date* accordingly. The *__fetch_within_timerange__ ()* method is used to get the subset of data by specifying the *agg_value, aggregation, date, metric* and *value*. The metric values obtained are extracted into *dictsubset* using subset operation on the 'measurement_value'

```
for site in self.test_list:
    fetched_dataframe = self.data_fetcher.fetch(site)
    fetched_dataframe = fetched_dataframe.fillna(1)
    fetched_dataframe.to_pickle(self.type + site + '.pckl')
    fetched_dataframe.index = pd.to_datetime(fetched_dataframe.index)
    incident_times = self.incident_time_manager.incident_extract_
    _from_csv(site)
```

Listing 2. Code snippet from the file 'datapreparator.py'

column and converting it into a dictionary. This is carried out in a while loop and is then concatenated into a resultant dataframe *result_df*. The date parameter in the *__fetch_within_timerange__ ()* method is updated by deducting 7 days in each iteration till the *to_date* date is reached.

Finally, the *metric_name* is formed by concatenating the *value, metric, aggregation* and *agg_value*. This is then used to rename the *'measurement_value'* column of the dataframe to the new column name which is the *'metric_name'* and the *result_df* is then returned.

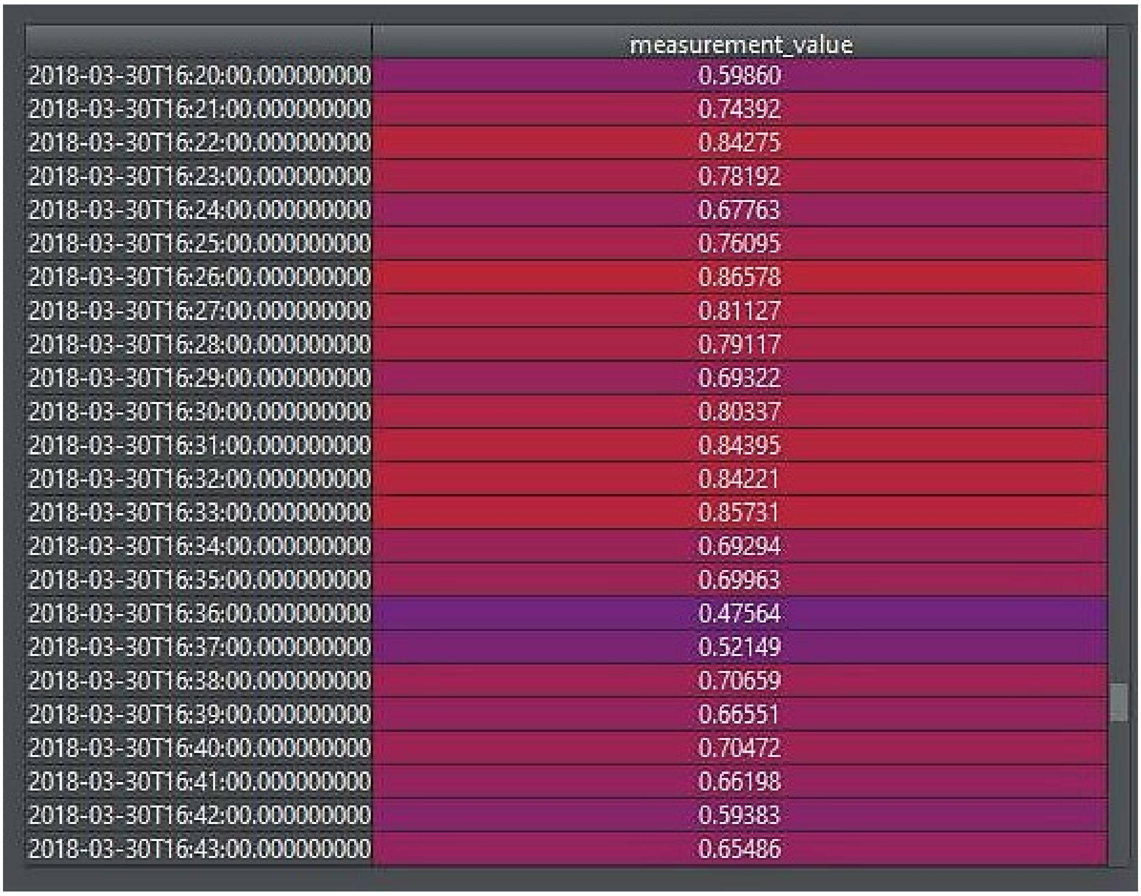
An example of the data fetched is from the site 'Megasport' which has around 120,000 rows (timestamp values) and 12 columns (metrics).

The *__fetch_within_timerange__ ()* method has two subsequent methods. The *get_model()* method is used to create a model from the *metric, value, aggregation, agg_value* etc. The *normalize_data()* method uses the subset data (subset of 'measurement_value' column) and normalizes it using the div operation by mean (mean of 'measurement_value' column). If there are missing values, they are filled with '1s' and all values greater than 1 are replaced by 1 as max value. The *__fetch_within_timerange__ ()* method basically gets the *date, start_week* and *end_week* using the *get_week_range ()* method. It calculates the mean and generates a dataframe. The dataframe values where the aggregation column values are same as *agg_value* are then aggregated and grouped by column 'time'. Thus the subset data frame is created by normalizing the data values.

Figure16 shows the normalized measurement values for site 'Megasport' alongwith timestamps.

Data Preparation module:

In the *DataPreparator* class, the most important method is *prepare_data()*. It takes as input the *window_width* and *window_height* parameters as specified in file 'main.py'. The following steps are performed here.



	measurement_value
2018-03-30T16:20:00.000000000	0.59860
2018-03-30T16:21:00.000000000	0.74392
2018-03-30T16:22:00.000000000	0.84275
2018-03-30T16:23:00.000000000	0.78192
2018-03-30T16:24:00.000000000	0.67763
2018-03-30T16:25:00.000000000	0.76095
2018-03-30T16:26:00.000000000	0.86578
2018-03-30T16:27:00.000000000	0.81127
2018-03-30T16:28:00.000000000	0.79117
2018-03-30T16:29:00.000000000	0.69322
2018-03-30T16:30:00.000000000	0.80337
2018-03-30T16:31:00.000000000	0.84395
2018-03-30T16:32:00.000000000	0.84221
2018-03-30T16:33:00.000000000	0.85731
2018-03-30T16:34:00.000000000	0.69294
2018-03-30T16:35:00.000000000	0.69963
2018-03-30T16:36:00.000000000	0.47564
2018-03-30T16:37:00.000000000	0.52149
2018-03-30T16:38:00.000000000	0.70659
2018-03-30T16:39:00.000000000	0.66551
2018-03-30T16:40:00.000000000	0.70472
2018-03-30T16:41:00.000000000	0.66198
2018-03-30T16:42:00.000000000	0.59383
2018-03-30T16:43:00.000000000	0.65486

Figure 16. Normalized measurement values for site 'Megasport'

- The first step is to fetch the business metrics data from InfluxDB using the *fetch()* method which resides in the DataFetcher class (file 'fetcher.py') and storing it in a dataframe. The data is fetched between a given time range and the values are normalized between '0' and '1'. The missing values are filled with '1s'. If we fill them with zeros, on normalizing such zero values, there are a lot of infinite values. Serialization [22] is performed on the dataset and the data is stored as a pickle file along with the site name for quicker read operations later. The dataset can then be read using *read_pickle* into a dataframe and is indexed using the datetime function provided by Pandas library. The *incident_times* i.e. the time ranges where an incident is recorded, are extracted from the csv file. All the above steps are done for each site in the *test_list* of site names. Code fragments from the file 'datapreparator.py' are given in listing 2 which shows the code for data fetching, listing 3 which shows the code for the data slicing operation and listing 4 which

[22] https://docs.python.org/3/library/pickle.html

```
for x in range(0, fetched_dataframe.shape[0] - window_width):
    slice = fetched_dataframe[x:x + window_width]
    for index, row in slice.iterrows():
        if check_for_incident(index, incident_times):
             incident = True
        else:
             incident = False
```

Listing 3. Code snippet from the data pre-processing file 'datapreparator.py'

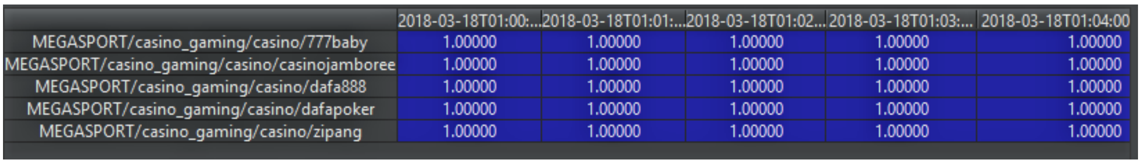
```
for item in incident_marked_list:
     result_list.append((item, 1))
return result_list
```

Listing 4. Code snippet from the data pre-processing file 'datapreparator.py'

shows the code for combining the incidents and non incidents list into a single result list with labels.

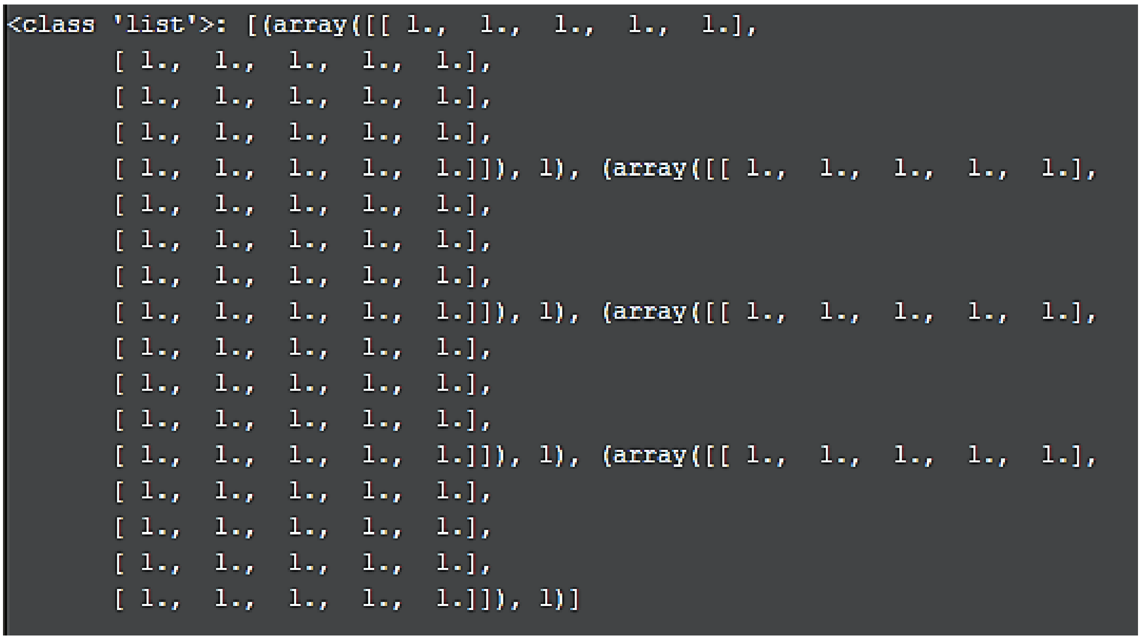
- Next step is to slice the data frame according to the window_width and window_height, iterate over all the values of the slice rows and check for incidents (*check_for_incident()* method) using the *incident_times* which were extracted in the first step. In the *check_for_incident()* method, the whole incident times list is traversed and the timestamps in the business metrics data are compared with the start and end time values of the incident times list. Based on whether there is an incident or not, the incident flag is updated with either a True or False boolean value.

- The slice is then transposed and sliding window technique is followed to create a *window_slice*. Figure17 shows a window_slice of size 5x5. Here, a window size of 5x5 i.e. *window_height* = 5 and *window_width* = 5 is chosen. This *window_slice* is then converted into a *slice_matrix* which is a 2D array containing data values. The incident flag is again checked and the slice matrix is added to the *incident_marked_list* or *non_incident_marked_list* based on the boolean value of the incident flag and the corresponding incident flag values are set as labels. Both, the *incident_marked_list* and *non_incident_marked_list* are a list of tuples having the sliced matrix and flag value (either 0 or 1) indicating an incident or not an incident. A combined result list, containing values from *incident_marked_list* and *non_incident_marked_list*, is then returned back to the main script 'main.py'. Figure 18 shows few tuples of the incident_marked_list which has the slice matrix as well as the label for site 'Megasport'.



	2018-03-18T01:00:...	2018-03-18T01:01:...	2018-03-18T01:02...	2018-03-18T01:03:...	2018-03-18T01:04:00
MEGASPORT/casino_gaming/casino/777baby	1.00000	1.00000	1.00000	1.00000	1.00000
MEGASPORT/casino_gaming/casino/casinojamboree	1.00000	1.00000	1.00000	1.00000	1.00000
MEGASPORT/casino_gaming/casino/dafa888	1.00000	1.00000	1.00000	1.00000	1.00000
MEGASPORT/casino_gaming/casino/dafapoker	1.00000	1.00000	1.00000	1.00000	1.00000
MEGASPORT/casino_gaming/casino/zipang	1.00000	1.00000	1.00000	1.00000	1.00000

Figure 17. Window slice (5x5) for site 'Megasport'



```
<class 'list'>: [(array([[ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.]]), 1), (array([[ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.]]), 1), (array([[ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.]]), 1), (array([[ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.,  1.]]), 1)]
```

Figure 18. Example of incident marked list for site 'Megasport'

7.2.2 CNN based model Implementation

In this section, the design and implementation details of the CNN based model are explained. This model is part of the Analysis module.

In the *Models* class, the method *data_model_cnn()* comprises of the CNN model used for detecting and categorising business incidents. It takes as input the *final_list* and *final_list_test* along with the input matrix *window_height* and *window_width*.

Convolutional Neural Network model: The input to the CNN model is the *final_list* which is divided into train and validation sets in 80:20 ratio. The *final_list_test* is used as the test set. The listing 5 shows the code snippet for the same.

```
train = final_list[:int(len(final_list)*0.80)]
valid = final_list[int(len(final_list)*0.80):]
test = final_list_test[int(len(final_list_test) * 0.100):]
```

Listing 5. Train, validation and test dataset

Listing 6 is a code block depicting the sequential model generated using Keras. The model specifics are given below.

- There are two convolutional layers which uses input of the shape (5,5,1). There are 64 hidden units, the kernel size is 4x4 and 3x3 in the first and second layers respectively and activation unit type is ReLu.
- There are two pooling layers with pool size 2x2.
- The dropout is set to be 0.25.
- Lastly there is a dense layer of output functions which uses number of classes as 2 and the activation function as sigmoid.
- The value of batch_size which signifies how much data is processed in one iteration is 64
- The number of epochs is 10.
- Adadelta optimizer is used.
- The number of classes is 2 (in case of incident and non-incident classification).

The sequential model is generated using the Keras library. The train data and validation data is used for fitting the model. Next the model is evaluated on the test data using evaluation metrics like precision, recall, f-measure and accuracy. A confusion matrix is also plotted to show the true positives, false positives, true negatives and false negatives.

The first fully-connected (dense) layer contains the largest number of trainable parameters in comparison to other layers. The sigmoid activation function is used to detect if an incident is present or not (two-class classification). The probability for every class is also computed and label with the highest probability/confidence is chosen as the predicted label. The results of the experiments are given in 8.2.

```
model = Sequential()
model.add(Conv2D(64, kernel_size=(4, 4), strides=(1, 1),
name='layer_conv1', activation='relu', input_shape=(rows, cols,1),
border_mode='same'))

model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
name='maxPool1'))

model.add(Conv2D(64, kernel_size=(3, 3), strides=(1, 1),
name='layer_conv2', border_mode='same', activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1),
name='maxPool2'))

model.add(Flatten())
model.add(Dense(32, activation='relu', name='fc1'))
model.add(Dropout(0.25))
model.add(Dense(num_classes, activation='sigmoid', name='fc2'))
```

Listing 6. The implemented Convolutional Neural Network model

Evaluation Module:
In the *Evaluation* class, the method *evaluation_metrics()* is used to calculate the metrics such as precision, recall and f-measure. It takes as input the actual test values *y_test* and the predictions *y_pred*. The evaluation metrics are calculated using the confusion matrix values. The formula for calculating precision, recall and f-measure is given by the equations 2,3 and 5 respectively. The accuracy of the CNN model can be measured during the training of the model. The values of the evaluation metrics for different experiments is given in the Section 8.2.

7.3 Summary

In this chapter, the prototypical implementation of the proposed solution including the data pre-processing mechanism and the CNN based analysis model is detailed alng with code snippets. The chapter also explains the interaction between the components of the proposed solution. This contributes in answering the research question "How to design and develop a viable solution for the detection of anomalies and categorisation of business incidents?" (**SQ2** in Section 1.2).

8 Evaluation

This chapter provides an evaluation of the proposed design and the implemented prototype It also includes the evaluation of the implemented CNN based model thus answering the third sub-question (**SQ3** in Section 1.2). This question was subdivided into the following questions: 1) What is the system/hardware configuration? 2) What are the results and means of evaluating the model? 3) How to interpret the results and assess the performance of the proposed solution?

Section 8.1 describes the system and environment configuration. Next, this chapter mentions results in Section 8.2 and explanation of performance measurements using these results. Section 8.3 discusses the requirement evaluation and summarizes the performance and results of the developed prototype.

8.1 Baseline and Methodology

In this section, the system configuration requirements are listed.

System/Hardware Configuration The implementation is done using Python (version 3.6.3) and keras (version 2.1.5) using tensorflow libraries, on a physical machine with the configuration as shown in Table 1.

Table 1. System configuration

Operating System	Microsoft Windows 10.0.15063 LTS 64 Bit (Linux Kernel 4.4.0.x)
RAM Memory	23,8 GB
Processor	Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz, 2 Core(s)
Physical Memory	476 GB

Pycharm IDE [23] (PyCharm 2017.3.2 Community Edition) is used for the implementation of Convolutional Neural Network based model. The data is stored in InfluxDB (version 4.1.1). The main python libraries used are Python pandas and NumPy for data pre-processing and Scikit-learn for the CNN model. The details of the versions of these libraries are provided in Table 2.

[23] https://www.jetbrains.com/pycharm/

Table 2. Main Python packages and libraries used

Packages	**Version**
python	3.6.3
pandas	0.20.1
keras	2.1.5
tensorflow	1.6.0
scikit-learn	0.19.1
ipython	6.1.0
matplotlib	2.1.2
numpy	1.13.3

8.2 Convolutional Neural Network Model Results

This section consists of the results of the CNN based model. The model is run multiple times with different data sizes and parameters. Some of the experiment results including the data size, time taken for the model to train and the different evaluation metrics[24] are listed here. It also includes the confusion matrix and the plots for training and validation accuracy and loss to better understand the model.

Experiment 1

For experiment 1, the data was collected from 17 sites and each site had 12 business metrics. The data was pre-processed and transformed in order to be used for training the model. Table 3 shows the first results. It has the values of the time taken to fetch and preprocess the train and test data, the time taken to train the model and the time taken to calculate evaluation metrics. The table 4 shows the evaluation metrics values for the same trained model on test data. The accuracy value is 48% and the model run time is slight over 16 hours. Therefore, it was decided to try and train the model with 5 sites and check if there was an error in the data pre-processing.

Table 3. First run: time measurements (in mins)

Run #	**Fetching & pre-processing time (Training Data)**	**Fetching & pre-processing time (Test Data**	**Model Run time**	**Evaluation metrics calculation time**
1	224.47	122.48	968.72	1.047

[24] These evaluation metrics used to assess the CNN model results like accuracy, precision, recall are described in Appendix III

Table 4. First run: evaluation metrics (CNN)

Run#	Activation function	Precision	Recall	F-measure	Accuracy
1	Sigmoid	0.29	0.02	0.04	0.48

Experiment 2
As the training time in experiment 1 was ~16 hours, the idea was to reduce the number of sites and see if it reduces the time taken as well as improve the accuracy. So the number of sites was reduced to 5. The table 5 shows the results for 5 sites. It includes model run time and evaluation metrics. It is observed that the model run time has reduced considerably from 16 hours to 1 hour. However, there is no significant change in the accuracy of the model which still remains 48%. The size of the data and the time taken for pre-processing the data is given in table 5 and the evaluation metrics are given in table 6. An addition in these tables is the size of the input data incident points and non-incident points for the sites before it is sent to the model to be trained.

Experiment 3
So from Experiment 2, it is found that reducing the number of sites reduces the training time although it does not affect the accuracy. Therefore, for the next experiment, there were changes made to the data pre-processing and incident detection code. The model was then trained again for 17 sites. Also, it was made sure that the size of the input data incident points and non-incident points for the sites is comparable and balanced. The following tables 7 and 8 give the results and evaluation metrics. The accuracy value is ~58.53%. In table 7, the first row depicts an imbalance in the data for incident and non incidents. The training time is over 4 hours with an accuracy of 54%. The next observation has a more balanced dataset. Figures 19 and 20 show the training and validation loss and accuracy respectively. Figure 21 shows the confusion matrix.

Table 5. Time measurements for 5 sites (in mins)

Run#	Incident data points (Train Data)	Non-Incident data points (Train Data)	Incident data points (Test Data)	Non-Incident data points (Test Data)	Fetching & pre-processing time (Train Data)	Fetching & pre-processing time (Test Data)
1	15927	16097	8063	8607	64.13	22.70

Table 6. Evaluation metrics (CNN) for 5 sites

Run#	Number of sites	Number of epochs	Activation function	Model Run Time (in mins)	Precision	Recall	F-measure	Accuracy
1	5	5	sigmoid	33.65	0.47	0.036	0.067	0.489
2	5	10	sigmoid	56.38	0.42	0.034	0.062	0.485

Table 7. Time measurements for 17 sites (in mins)

Run#	Incident data points (Train Data)	Non-Incident data points (Train Data)	Incident data points (Test Data)	Non-Incident data points (Test Data)	Data Pre-processing time (Train Data)	Data Pre-processing time (Test Data)
1	151227	16059959	50663	6703607	52.75	22.63
2	151227	162311	50663	66914	19.28	7.29

Table 8. Evaluation metrics (CNN) for 17 sites

Run#	Number of sites	Number of epochs	Activation function	Model Run Time	Precision	Recall	F-measure	Accuracy
1	17	10	sigmoid	250.44	0.47	0.03	0.06	0.54
2	17	10	sigmoid	138.71	0.53	0.80	0.64	0.58

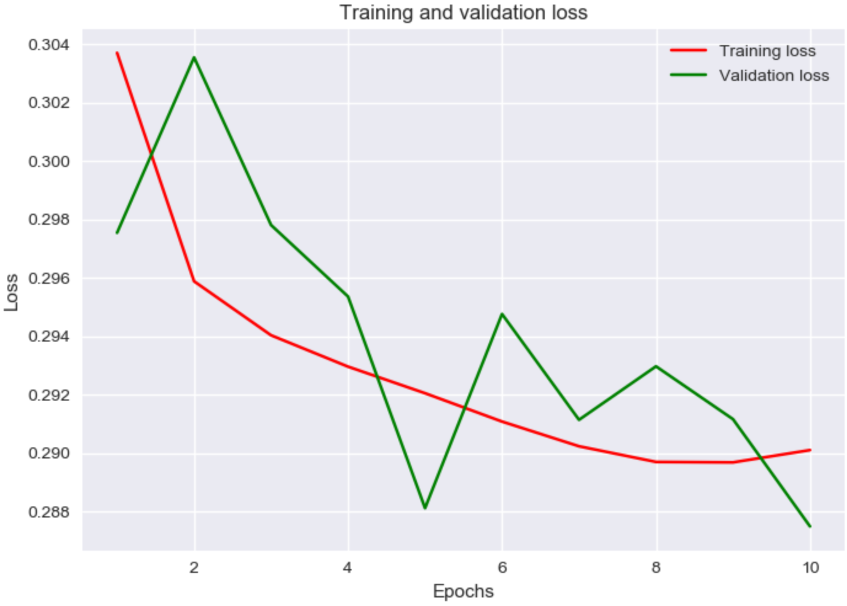


Figure 19. Loss (CNN)

Visualization of Weights in Convolutional Layers: As the CNN architecture is like a black box, the weights of the convolutional layers of the network are visualized as shown in Figures 22 and 23. In the first convolution layer, the size of the matrix is 4x4 while in the second convolution layer it is 3x3. In the Figure 23, one can observe the changing intensities implying that the kernels are changing over time during the training. Some of the highlighted kernels (red) show similarity to Laplacian kernel [56] and Sobel kernel [56] which helps in edge detection. Other kernels (highlighted by green) show linear filters as well as separable kernels. A separable kernel can be thought of as two one-dimensional kernels, which is applied by first convolving with the x-kernel and then with the y-kernel [56].

8.3 Discussion

This section evaluates the requirements of the solution as described in Chapter 4 and discusses the results.

8.3.1 Requirements Evaluation

This section provides an evaluation of our proposed design and the implemented prototype based on the requirements as discussed in Section 4.

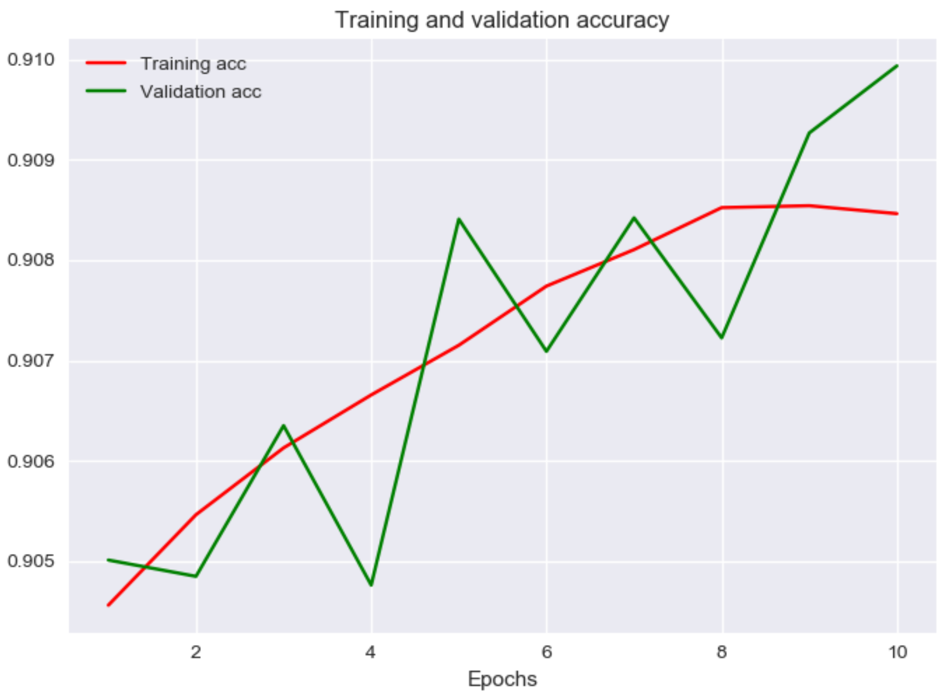


Figure 20. Accuracy (CNN)

- F1: An analyst shall be able to fetch the data within a given time range: The developed prototype supports that the data is fetched from the database within a given time range and for particular sites. The connection is established to the InfluxDB using the necessary parameters for the connection. The data received is then cleaned in the Data Fetcher module.

- F2: An analyst shall be able to transform the data: The developed prototype supports that once the data is collected, the data is pre-processed and transformed in the *Data Preparation module*. The data values are normalized, sliced into matrices and thus transformed as input to the CNN model.

- F3: An analyst shall be able to detect anomalies using the proposed solution: The developed prototype implements a CNN based model for the detection of anomalies and business incidents categorisation in the *Analysis module*.

- F4: An analyst shall be able to classify business incidents categories: The developed prototype supports the categorisation of business incidents. This is done using a CNN based model.

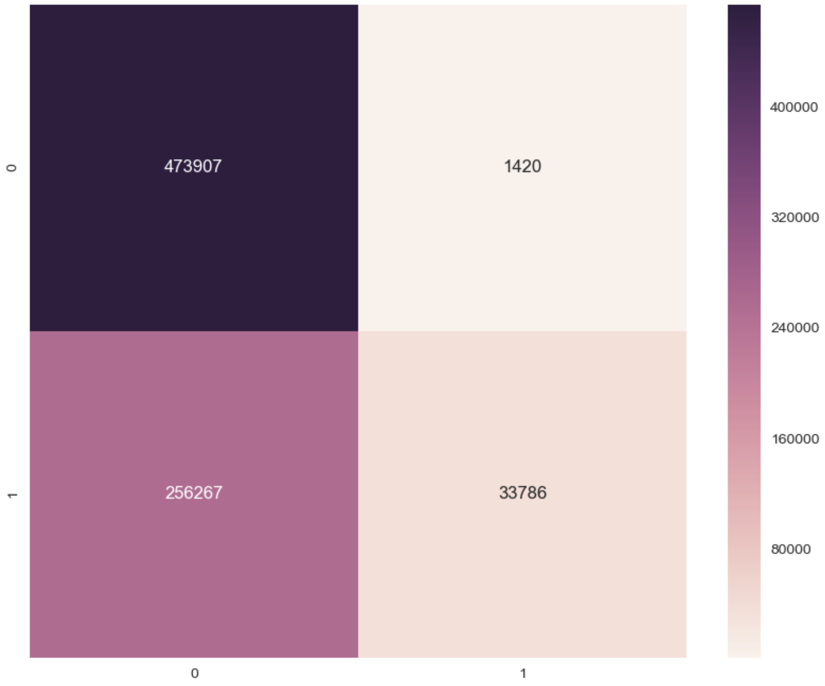


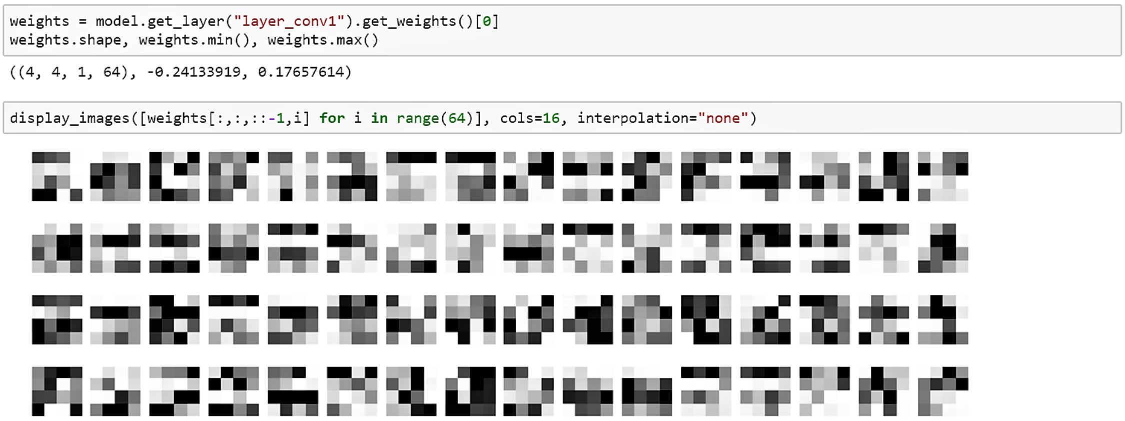
Figure 21. Confusion matrix (CNN)

- F5: An analyst shall be able to predict future business incidents categories: Additionally, the developed prototype supports the prediction of business incidents using the CNN based model.

- F6: An analyst shall be able to evaluate the model results: The developed prototype supports that the implemented model can be evaluated. The evaluation is done on the basis of the categorised business incidents. This is done in the *Evaluation module* .

- P1: Memory: The developed prototype runs on a 24GB RAM and 467 GB machine. The maximum memory requirement based on the data size and computations is roughly between 4GB and 6GB.

- P2: Computation Speed: The speed of the developed prototype is measured in terms of time taken. The optimal time taken for CNN model to prepare data and perform training is approximately between 2-4 hours when the data is read from



```
weights = model.get_layer("layer_conv1").get_weights()[0]
weights.shape, weights.min(), weights.max()
```

```
((4, 4, 1, 64), -0.24133919, 0.17657614)
```

```
display_images([weights[:,:,::-1,i] for i in range(64)], cols=16, interpolation="none")
```

Figure 22. Visualization of weights in convolutional layer 1

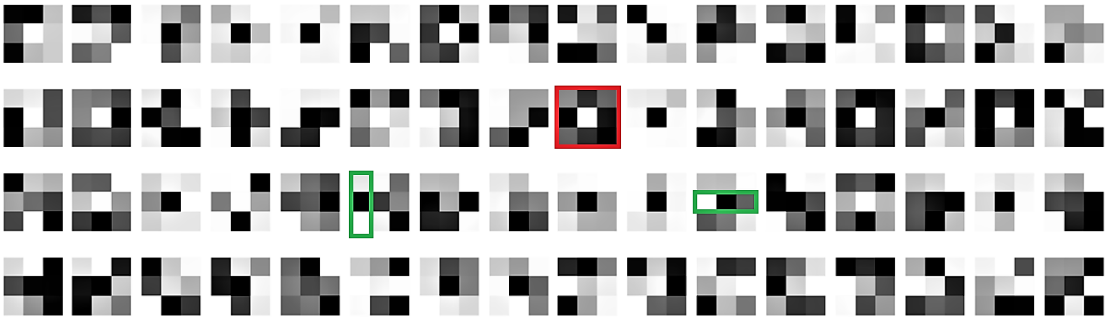


Figure 23. Visualization of weights in convolutional layer 2

the saved pickle files otherwise fetching data from the database takes additional time of about 3-6 hours (depending on number of sites and time range).

- P3: Extensibility: The developed prototype allows easy addition of new statistical and machine learning algorithms as we can provide the fetched input data with minor changes and use it to do the categorisation task. It is easily upgradable to multiple sites by just adding the sites name to the list of sites for which the analysis needs to be performed.

8.3.2 Interpretation of Results

This section discusses and interprets the results. The difference between the solutions used by Playtech plc and the proposed solution is also discussed.

Model Results: The CNN model gives an accuracy of 58% and the time taken for the model to run is approximately 2 hours. The time for data collection and data preparation/pre-processing for training and test data combined is between 3 - 6 hours. These values are for input data from 17 sites. As the number of sites are increased, the data fetching, pre-processing and model run times increase too.

However, it is worth mentioning that this method didn't contain a benchmark of other CNN architectures. In fact, this serves as a benchmark and reference model for future when the prototype will be tested and deployed in production at Playtech plc.

Comparison with Previous Solutions at Playtech plc: As mentioned earlier, according to the service operation team experts, in case of HP SHA, there were very limited option for configuration and for small changes and tuning, one needed to contact HP services. Moreover, there were around 800 false alerts each week and this worsened during daylight savings time transitions. This made the system almost impractical to use in practical scenario.

In case of Playtech plc's current rule based solution, the business incidents are detected and an alert is sent to the concerning team. The number of alerts were successfully reduced to less than 450 per month thus reducing the number of false alerts. Although with the current solution, there is a possibility of configuration but a lot of time and effort is required to setup and prepare the statistical models for each metric of each site. There are around 360 software upgrade deployments per month, in over 40 production sites and on an average, approximately 50,000 components get updated bi-weekly. This kind of modifications in the configuration takes approximately 5 - 6 hours per site. Therefore, it is important to reduce the effort in maintaining the statistical models as the maintenance costs are too high. The proposed solution aimed to improve on this and use a deep learning approach to categorise business incidents. The proposed solution does not require indepth analysis of site metrics, behaviour or configuring any predefined rules which reduces the time and efforts required in configuration.

Next is the speed of reaction to the business incidents. SHA took about 15 minutes to detect any unusual behaviour in metrics while the current solution takes about 3-4 mins to detect anomalies and send out notification. As the proposed solution is to be used in conjugation with the current solution, as a parallel engine. Although it is not yet deployed but the reaction time is the expected as the current prototypical implemented solution. For now, to evaluate the CNN model, the accuracy and time performance of the implementation is taken into consideration and is detailed in the results section 8.2. This will serve as a benchmark for the solution when deployed in production.

Also, according to the service operations at Playtech plc, there were a lot of false positives for incident detection. So, it was required to have a trustworthy solution with less false incidents. The proposed prototypical implementation showed fewer false positives for the test data. In the interviews conducted in Playtech plc for research purposes, the results of the proposed solution were presented and the general consensus was that the results look promising. In conjugation with the rule matching engine, this solution would contribute in reduction of false positives.

The additional improvements for future could be to filter the type of business incidents occurred and where they occur such as in application layer or network layer. The business metrics at a more granular can be used for this purpose.

8.4 Summary

In this chapter, evaluation of the proposed solution and the CNN model is done which helps in providing an answer to the research question “What is the efficiency of the solution for detection of anomalies and categorisation of business incidents and how is it measured?” (**SQ3** in Section 1.2). The system and environment configuration of the proposed solution is outlined. Requirements evaluation is discussed and the results for the CNN based model are provided along with the interpretation of the results and comparison with previous solutions.

9 Conclusion and Future Work

This section concludes this thesis. Section 9.1 answers the main question (MQ) and SQs as discussed in Section 1.2. Section 9.2 gives the concluding remarks along with some limitations of the prototype solution which motivates the next Section 9.3 i.e. the future work possibilities.

In the course of this work, several contributions were made both on a theoretical and practical level. At first, the need for early detection of business incidents and their correct categorisation faced by Playtech plc is explained, thus stating the problem description. The current state of the art and related work is mentioned, discussing the different types of systems and analysis methods in use for similar problems. An overview of the machine learning and deep learning is given, building the background knowledge needed to understand the thesis. Later, the environment setting i.e. the Playtech plc system and their current solutions are outlined.

This thesis then listed the requirement specification for the proposed solution which laid the foundation for the design and the implementation of the proposed solution. The main objective was to detect anomalies and categorise business incidents which is achieved as well as evaluated. For this, the thesis presents a deep learning approach. The main aim here was to use a CNN model and determine if this approach works in this business case scenario. Finally, some results were listed for the CNN model used and illustrated with the help of tables and figures.

9.1 Answers to Research Questions

In the goals and contribution Section 1.2, the main question **(MQ) -**
"How to detect anomalies/business incidents and categorise them using machine learning?" was divided into three sub-questions.

SQ1 - What are the current approaches and solutions that exist for anomaly detection?
In this thesis, anomaly detection and the techniques to do anomaly detection are mentioned. The different research-based anomaly detection approaches and commercial solutions are discussed constituting the state of the art.

The background knowledge necessary for the understanding of thesis such as machine learning process, deep learning, traditional machine learning approaches etc. are outlined. The environment setting i.e. the Playtech plc system and the solutions used by them are also described.

SQ2 - How to design and develop a viable solution for the detection of anomalies and categorisation of business incidents?
The information from the state of the art and requirement specifications is used to propose a solution using a deep learning approach. The solution is designed and a prototype is implemented to detect anomalies, categorise and predict business incidents which used the CNN model(deep learning approach).

SQ3 - What is the efficiency of the solution for detection of anomalies and categorisation of business incidents and how is it measured?
In order to find the efficiency of the analysis method in the proposed solution, various evaluation metrics such as accuracy, precision, recall are used. The model is compared to the current solution for assessment too.

9.2 Conclusion

The developed prototype solution is able to detect anomalies and categorise business incidents with an accuracy value of 58%. The model was trained and run several times using different data sizes and different layers of the Convolutional Neural Network in order to achieve the optimal results. The final results are listed in Section 8.2 for the CNN model. The precision, f-measure and confusion matrix values, kernel visualizations reflect that the model performed well and successfully categorised the business incidents.

The proposed solution overcomes the problem of delayed detection of business incidents which led to adverse financial effects. This solution is viable and could be utilised for faster detection and categorisation of the incidents,This enables the specialists using this solution to react swiftly to the business incidents and take relevant actions for the resolution of the same. This solution is also beneficial for setting timelines and planning for downtimes.

The work done in thesis not only helps Playtech plc to effectively as well as proactively monitor and manage the business incidents but can also serve as research for people working in related fields and building similar projects.

Limitations: There are few limitations or challenges that were faced while working on the proposed solution which are worth mentioning.
The proposed solution is designed and developed for Playtech plc environment setting but the concepts are generic enough to apply it under any other business setting. The developed solution is implemented in Python programming language for quicker prototypical implementation which is comparatively slower than more low level languages e.g. C, Rust.

The most time consuming activity was fetching and preparing the data for input to the CNN model. The data transformations were quite challenging as the business metrics data was not structured in the way that could be provided directly to model for training.

The imbalance in incidents and non-incidents data was quite challenging to remove as the non-incident data data was more than the incident data. Additionally, most of the incidents were new and non-repeating which posed a problem while training the machine learning model.

The amount of time the proposed model takes for initial training is about 3 - 6 hours that posed difficulty to try and implement models with different hyperparameters, kernel values and number of layers.

9.3 Future Work

The proposed solution and the developed prototype could be improved further. The first addition could be to classify the business incidents according to priority into three different categories i.e. Priority A (high), B (medium), C (low) using several additional parameters. This is being developed and partially done in the case of Playtech plc. Although, this solution requires certain business metrics at a more granular level in order to improve the accuracy and efficiency of the solution.

From the results in Section 8.2, it is found that there were some incidents which were either not detected (False Negatives) or misclassified as incidents (False Positives). To tackle this issue, the model could be improved by supplying better labeled data and balance the incident and non-incident data. But providing proper labeled data for an anomaly detection solution in itself is a challenge.

Another important thing to improve on is to balance the training data for incidents and non-incidents. Additionally, as mentioned in the limitations, each business incident differs slightly from the other because its own contextual sense and as mentioned in the problem statement 1.1, all new or non-repeating business incidents might or might not be detected through the machine learning solutions.

As mentioned before, fetching and preparation of training data set took a lot the time while working on this thesis. So, in future AutoML techniques could be used that clean and prepare the features automatically.

Another important addition could be using a graphical user interface which would improve usability. There are many tools available such as Dash[25] which can be used.

A relatively new approach was used for anomaly detection with the hope of getting better results in this business case. There is always a scope for improvement as the Convolutional Neural Networks evolve with time which results in more sophisticated and deeper layers.

[25] https://plot.ly/products/dash/

List of Tables

List of Figures

References

[1] Baron Schwartz and Preetam Jinka. *Anomaly Detection for Monitoring*. O'Reilly Media, Inc., March 2016.

[2] Cost of hourly downtime for enterprises in 2017 - 2018. `https://www.ibm.com/downloads/cas/YGLRKEEK/`. Last checked: 28 Mar 2019.

[3] Hp service health analyzer. `http://1fctr.saas.hp.com/topaz/amdocs/eng/doc_lib/BSM_Help.htm#SHA/UserGuide/C_Welcome.htm`. Last checked: 20 Aug 2018.

[4] Michael Pecht and Myeongsu Kang. *Introduction to PHM: Fundamentals, Machine Learning, and the Internet of Things*, pages 1–37. 08 2018.

[5] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[6] Animesh Patcha and Jung-Min Park. An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Comput. Netw.*, 51(12):3448–3470, August 2007.

[7] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407, 2019.

[8] Terran Lane and Carla E. Brodley. An application of machine learning to anomaly detection. 1997.

[9] Nico Görnitz, Marius Kloft, Konrad Rieck, and Ulf Brefeld. Toward supervised anomaly detection. 2013.

[10] Timo Schindler. Anomaly detection in log data using graph databases and machine learning to defend advanced persistent threats. *CoRR*, abs/1802.00259, 2018.

[11] Alae Chouiekh and EL Hassane Ibn EL Haj. Convnets for fraud detection analysis. *Procedia Computer Science*, 127:133–138, 01 2018.

[12] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. 2017.

[13] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. *CoRR*, abs/1611.06455, 2016.

[14] Zhiguang Wang and Tim Oates. Encoding time series as images for visual inspection and classification using tiled convolutional neural networks. 01 2015.

[15] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 3995–4001. AAAI Press, 2015.

[16] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. 03 2016.

[17] Yoni Yom Tov Ben Simhon, David Drai, and Ira Cohen. System and method for transforming observed metrics into detected and scored anomalies. `https://patentimages.storage.googleapis.com/f1/1b/40/c8f541e310b23f/US10061632.pdf`, 2018. Last checked: 21 Jan 2019.

[18] Jason Bloomberg. Real time anomaly detection and analytics for today's digital business. Intellyx LLC, Intellyx White Paper, 2016. Last checked: 26 Sep 2018.

[19] NY 10504 IBM Corporation, Armonk. Ibm watson: Next generation cognitive system. IBM Watson Software White Paper, 2012. Last checked: 26 Sept. 2018.

[20] NY 10589 IBM Corporation Software Group, Somers. Becoming a cognitive business with ibm analytics. IBM Analytics White Paper, 2016. Last checked: 15 Aug 2018.

[21] A. Chabert, A. Forster, L. Tessier, and P. Vezzosi. *SAP Predictive Analytics: The Comprehensive Guide*. SAP Press. Rheinwerk Publishing Incorporated, 2017.

[22] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1):12:1–12:34, December 2012.

[23] Joao Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC, 1st edition, 2010.

[24] Tavish Srivastava. A complete tutorial on time series modeling in r. `https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/`. Last checked: 11 Nov 2018.

[25] Zhiqiang Ge, Zhihuan Song, Steven Ding, and Biao Huang. Data mining and analytics in the process industry: the role of machine learning. *IEEE Access*, PP:1–1, 09 2017.

[26] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[27] Volcanohong. Machine learning notes. `https://volcanohong.github.io/2016/09/01/machine-learning-notes//`. Last checked: 11 Mar 2019.

[28] Fazel Famili, Wei-min Shen, Richard Weber, and Evangelos Simoudis. Data preprocessing and intelligent data analysis. *Intell. Data Anal.*, 1:3–23, 12 1997.

[29] Cannannore Nidhi Narayana Kamath, Syed Bukhari, and Andreas Dengel. Comparative study between traditional machine learning and deep learning approaches for text classification. pages 1–11, 08 2018.

[30] Sotiris Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31, 10 2007.

[31] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[32] Misha Denil, David Matheson, and Nando de Freitas. Narrowing the gap: Random forests in theory and in practice. *31st International Conference on Machine Learning, ICML 2014*, 2, 10 2013.

[33] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, Sep. 1999.

[34] Taoran Cheng, Pengcheng Wen, and Yang Li. Research status of artificial neural network and its application assumption in aviation. pages 407–410, 12 2016.

[35] Kaushal Kumar, Gour Sundar, and Gour Mitra Thakur. Advanced applications of neural networks and artificial intelligence: A review. *IJITCS*, 4:57–68, 05 2012.

[36] Yann LeCun, Y Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[37] Yann Lecun, Bernhard Boser, John Denker, Don Henderson, R E. Howard, W.E. Hubbard, and Larry Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1:541–551, 12 1989.

[38] Pranavathiyani G. Google trends on data science. `https://towardsdatascience.com/google-trends-on-data-science-160146fea72a/`. Last checked: 19 Aug 2018.

[39] François Chollet. *Deep Learning with Python*. Manning, November 2017.

[40] Yann Lecun, Leon Bottou, Y Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[41] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[44] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei Fei Li. Imagenet: a large-scale hierarchical image database. pages 248–255, 06 2009.

[45] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and S. S. Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv.*, 51(5):92:1–92:36, September 2018.

[46] Hp service health analyzer brings predictive control to real-time service delivery. An Enterprise Management Associates® (EMA™) White Paper Prepared for Hewlett-Packard, 2011. Last checked: 22 Aug 2018.

[47] Jeffrey Joyce, Greg Lomow, Konrad Slind, and Brian Unger. Monitoring distributed systems. *ACM Trans. Comput. Syst.*, 5(2):121–150, March 1987. `http://doi.acm.org/10.1145/13677.22723`.

[48] Aleksandr Tavgen. Never fail twice. `https://medium.com/@ATavgen/never-fail-twice-608147cb49b/`. Last checked: 19 Apr 2019.

[49] Vlasta Hajek, Tomas Klapka, and Ivan Kudibal. Benchmarking influxdb vs. mongodb for time series data, metrics and management. `http://get.influxdata.com/rs/972-GDU-533/images/InfluxDB%201.4%20vs.%20MongoDB.pdf/`, 2018. Last checked: 23 Jul 2018.

[50] Syeda Noor Zehra Naqvi and Sofia Yfantidou. Time series databases and influxdb. `https://cs.ulb.ac.be/public/_media/teaching/influxdb_2017.pdf/`. Last checked: 17 Apr 2019.

[51] John Shahid. Influxdb documentation release 5.2.1. `https://buildmedia.readthedocs.org/media/pdf/influxdb-python/latest/influxdb-python.pdf/`. Last checked: 26 Feb 2019.

[52] Fei-Fei Li, Andrej Karpathy, and Justin Johnson. Cs231n: Convolutional neural networks for visual recognition 2016. Last checked: 11 May 2019.

[53] Jiudong Yang and Jianping Li. Application of deep convolution neural network. In *2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 229–232, Dec 2017.

[54] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into Imaging*, 9, 06 2018.

[55] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, and Tsuhan Chen. Recent advances in convolutional neural networks. *Pattern Recogn.*, 77(C):354–377, May 2018.

[56] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, Inc., 1st edition, 2016.

[57] David Powers. Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation. *Mach. Learn. Technol.*, 2, 01 2008.

[58] Alice Zheng. Evaluating machine learning models. `https://www.oreilly.com/ideas/evaluating-machine-learning-models/page/3/evaluation-metrics`. Last checked: 14 Apr 2019.

Appendix

I. Use Case description templates

Table 9. Use case description UC1

Usecase ID	**UC1**
Name	**Fetch Data**
Description	This use case represents fetching of data from the InfluxDB.
Actors	Analyst
Pre Conditions	Network connection and connection to the DB is active. Site name, start and end dates are entered
Basic Flow	1. The start date, end date and site name are entered and the method fetch() is called. 2. The data is collected week wise from start date to end date.
Alternate Flows	There is a connection error. The system shows an error message.
Extensions	-
Post Conditions	The data is successfully fetched.

Table 10. Use case description UC2

Usecase ID	**UC2**
Name	**Transform Data**
Description	This use case represents transforming the fetched data into sliced matrices with normalized data values.
Actors	Analyst
Pre Conditions	Network connection is active. The sliced window height and window width are entered.
Basic Flow	1. The fetched data is cleaned and the missing values are filled. 2. The fetched data values are normalized. 3. The normalized values are sliced into small matrices.
Alternative Flows	There is a connection error. The system shows an error message. There is a memory error. The system shows an error message and ends execution. There is a value error. The system shows an error message.
Extensions	Includes UC1: Fetch Data
Post Conditions	The fetched data is normalized and sliced into matrices.

Table 11. Use case description UC3

Usecase ID	**UC3**
Name	**Detect Business Incidents**
Description	This use case represent the detection of anomalies and business incidents.
Actors	Analyst
Pre Conditions	Network connection is active. The sliced matrices with normalized values are available.
Basic Flow	1. The normalized data values are entered to the function. 2. The machine learning model performs computations and detects an incident.
Alternative Flows	There is a connection error. The system shows an error message. There is a memory error. The system shows an error message and ends execution.
Extensions	Includes UC2:Transform Data
Post Conditions	The machine learning model performs computations and detects an incident.

Table 12. Use case description UC4

Usecase ID	**UC4**
Name	**Categorise Business Incidents**
Description	This use case represent the categorisation of business incidents.
Actors	Analyst
Pre Conditions	Network connection is active. The business incidents are detected.
Basic Flow	1. The incident lists are entered to the function. 2. The machine learning model performs computations, detects and categorises the business incidents.
Alternative Flows	There is a connection error. The system shows an error message. There is a memory error. The system shows an error message and ends execution.
Extensions	Includes UC3: Detect Incidents and Extends UC6: Evaluate incident categories
Post Conditions	The machine learning models performs computations, detects and categorises the business incidents.

Table 13. Use case description UC5

Usecase ID	**UC5**
Name	**Predict Business Incidents**
Description	This use case represent the prediction of any future business incidents.
Actors	Analyst
Pre Conditions	Network connection is active. The business incidents are detected and categorised.
Basic Flow	1. The incident lists are entered to the function. 2. The machine learning model performs computations, detects and categorises the business incidents. 3. The predictions are made
Alternative Flows	There is a connection error. The system shows an error message. There is a memory error. The system shows an error message and ends execution.
Extensions	Includes UC4:Classify Incidents
Post Conditions	The machine learning models performs computations and predictions are made.

Table 14. Use case description UC6

Usecase ID	**UC6**
Name	**Evaluate Model Results**
Description	This use case represents the evaluation of the categorised classes of the business incidents.
Actors	Analyst
Pre Conditions	Network connection is active. The business incidents are detected and categorised.
Basic Flow	1. The incident lists are entered to the function. 2. The machine learning model performs computations, detects and categorises the business incidents. 3. The categorised incidents are evaluated using evaluation metrics like precision, recall, accuracy. 6. The models are evaluated for performance, for example time taken for fetching data and the time taken for the model to run.
Alternative Flows	There is a connection error. The system shows an error message. There is a memory error. The system shows an error message and ends execution. There is a value error. The system shows an error message.
Extensions	-
Post Conditions	The model is evaluated using evaluation metrics such as precision, accuracy, recall.

II. Traceability between Requirements

Traceability between requirements and design of the proposed solution can be maintained in multiple ways. To list a few of the requirements which are taken care of in the design of the proposed solution are as follows. Figure 24 shows the traceability model and shows that the design of the solution will satisfy the functional and performance requirements. Table 15 shows the requirements traceability matrix. This can be explained by an example: The data is fetched from the DB over a secure connection which is then pre-processed and transformed. This fulfils the functional requirements F1 and F2.

Another inference can be that the machine learning model is used for detection of anomalies which fulfils the functional requirement F3 and performance requirements P1 & P2.

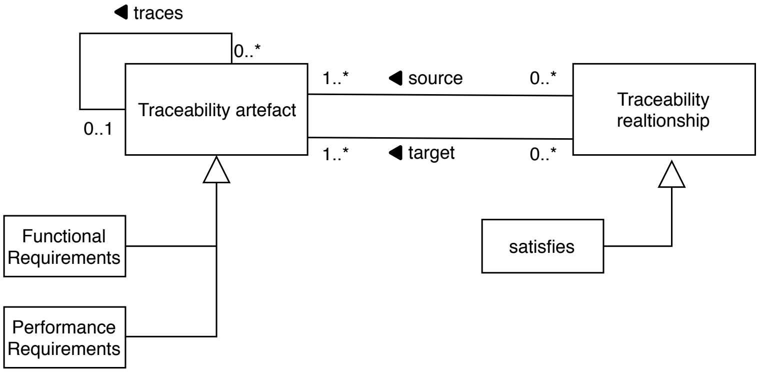


Figure 24. Traceability model

Table 15. Requirements traceability matrix

	F1	**F2**	**F3**	**F4**	**F5**	**F6**
P1			satisfies	satisfies	satisfies	satisfies
P2			satisfies	satisfies	satisfies	satisfies
P3				satisfies	satisfies	satisfies

III. Evaluation Metrics

The different evaluation metrics[26] are described below:

Precision: Precision [57] is the proportion of predicted positive cases that are correctly actual positives. In a classification task, a precision 1.0 means that every item labeled as belonging to a particular class actually belongs to that class. Precision can be considered as a measure of a classifier's exactness. A low precision means a large number of False Positives. It is given by the equation 2.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

where TP – True Positive, TN – True Negative, FP – False Positive, FN – False Negative.

Recall: Recall [57] is the proportion of actual positives which are correctly predicted positives by the model. If recall is 1.0, this means that every item from a particular class was labeled as belonging to that class. Recall can be considered as a measure of a classifier's completeness. A low recall means there are many False Negatives. It is given by the equation 3.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

Recall can also be referred as the True Positive Rate or Sensitivity. Specificity [57] is True Negative Rate and is given by.

$$\text{True negative rate} = \frac{TN}{TN + FP} \tag{4}$$

F-measure: Precision and recall does not give the best interpretation and are not used in isolation. Therefore, both the measures can be combined into a single measure such as F-measure [57]. F-measure or F1-score is the weighted harmonic mean of precision and recall. It is given by the equation 5.

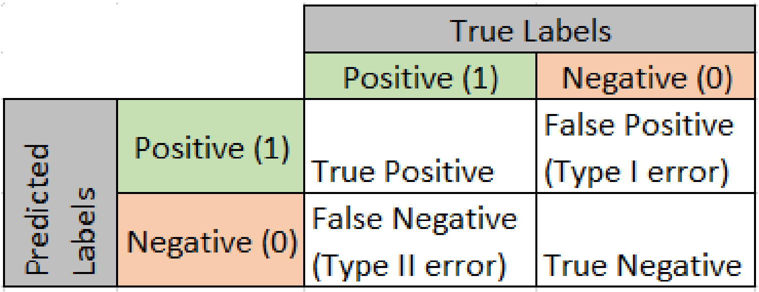
$$\text{Fmeasure} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5}$$

Accuracy: Accuracy [58] is the ratio between number of correct predictions given by the model and the total number of data points. Accuracy alone is not the best measure in case of a class-imbalanced data set, where there is a lot of disparity between the number of positive and negative labels. Therefore, its better to use precision and recall along with it. It is given by the equation 6.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

[26] https://scikit-learn.org/stable/modules/model_evaluation.html

Confusion Matrix: A confusion matrix [58] is a table which describes the performance of a classification model on a set of test data by giving the false positives and false negatives values. Figure 25 depicts an example of a confusion matrix for a 2-class classifier. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class or vice versa [57]. It helps to analyze the results and gives clues as to where the classifier is wrong.



		True Labels	
		Positive (1)	Negative (0)
Predicted Labels	Positive (1)	True Positive	False Positive (Type I error)
	Negative (0)	False Negative (Type II error)	True Negative

Figure 25. Example of a confusion matrix for 2-class classification

IV. Licence

*Non-exclusive licence to reproduce thesis

I, **Samreen Mahak Hassan**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for the purpose of preservation in the DSpace digital archives until the expiry of the term of copyright,

   Classification and Prediction of Business Incidents Using Deep Learning for Anomaly Detection,

   supervised by Aleksandr Tavgen, Martin Kiilo and Raimundas Matulevičius.

   Publication of the thesis is not allowed.

2. I am aware of the fact that the author retains the right specified in p. 1.

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

Samreen Mahak Hassan
16/05/2019