CLASSIFICATION OF ANOMALOUS TRACES IN AUDIT LOGS USING NEXT ACTIVITY PREDICTION

Master Thesis

Jennifer Alice Cutinha

in fulfillments of the requirements for the double degree of
Business Information Technology & Information Systems

Examination Committee
Prof. Dr. Maria Iacob
Faculty of Behavioural Management & Social Science
Department of Industrial Engineering & Business Information Systems
University of Twente, Enschede, The Netherlands

Prof. Dr. Dr. h.c. Jörg Becker
Chair of Information Systems & Information Management
Westfälische Wilhelms-Universität, Münster, Germany

Patrick Hafkenscheid, PhD
Senior Manager, Digital Risk Solutions
Deloitte Risk Advisory B.V.

FOR GOD’S IMMENSE GRACE AND BLESSINGS IN MY LIFE

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my mentor and committee chairperson Prof. Dr. Maria Iacob for her utmost guidance and support, not only during my thesis, but throughout the entire programme. Being the first student to navigate through this double degree programme was quite daunting and it was incredibly relieving and reassuring to know that someone had my back. Thank you Maria for being there! I would also like to thank Prof. Dr. Dr. h.c. Jörg Becker for taking on the role of being my second supervisor despite his extremely busy schedule and for all the appreciative comments and constructive feedback.

I want to extend my appreciation to my supervisor from Deloitte, Patrick Hafkenscheid, who was always available for discussions about anything I was unsure about, for constantly reviewing my work and providing in-depth feedback, and for being patient regarding the progress of my work. I would also like to thank Eric van Toor for trusting me and giving me the opportunity to work within the Digital Risk Solutions team. I am grateful for the colleagues I got to know and the time we spent together during my thesis internship.

A special word of thanks also goes to my programme coordinators Marloes van Grinsven and Martin Schmidt who did their very best to coordinate everything, who tried to resolve all the last minute problems I encountered and for their continuous support throughout the programme.

I am thankful for all my friends and classmates both at UT and WWU, as well as for my friends that I have gotten to know outside university who always checked on me and made sure I was doing well. You guys know who you are!:D

Lastly, but most importantly, this acknowledgement would be incomplete without the mention of the unending support, love and encouragement I received from my family, for being the light at the end of a very dark tunnel. Words cannot fathom how grateful I am to have you in my life.

Executive Summary

The auditing profession is an increasingly complex one. One of the main tasks of an auditor is to ensure that the financial statements produced by organisations meet legal standards. Financial statements can be considered a summary of business transactions that are encapsulated in business process, and therefore, monitoring or inspecting these processes can help validate the credibility of these statements. Given the advancements in the current technological space, the financial information of most organisations are captured by some form of information system.

Process mining seeks to leverage the data recorded in these systems in the form of event logs or audit trails. For this reason, process mining as an analytical tool for auditing has gained traction in the auditing community. However, despite the growing interest, the use cases for process mining, especially for predictive audit are fairly limited in practice. This study acknowledges this gap and demonstrates a practical use case of classifying anomalous traces in audit logs.

To guide the research process, the Design Science Research Methodology (DSRM) has been chosen as an appropriate research methodology. According to DSRM, the entry point for this research is the identification of a body of research, namely, Predictive Business Process Monitoring (PBPM) which is based on the continuous generation of predictions from the analysis of past execution traces. Following the identification of this research domain, the use case for predictive audit is formulated. As part of the design & development phase of the DSRM, a procedural method (the artefact) is developed which incorporates elements from both traditional data science pipelines as well as the generic approach adopted in PBPM literature.

The use of this procedural method is demonstrated by classifying anomalous traces in a loan application process, wherein the classification task is formulated as a next activity prediction problem which is a supervised multi-class classification problem. Accordingly, seven representative classifiers have been chosen from PBPM literature : Random Forest (RF), Extreme Gradient Boosting (XGBOOST), Multinomial Logistic Regression (MLR), Support Vector Machines (SVM), Long Short Term Memory

(LSTM) and Bidirectional Long Short Term Memory (BI-LSTM). The performances of the classifiers are evaluated using confusion matrices. Based on the confusion matrices, the precision, recall and f1-score of each classifier is also computed. Additionally, the classifiers are evaluated across four different configurations to understand the effects of hyperparameter tuning and class weighting for class imbalance on the base models.

The findings indicate that RF performs the best across all configurations achieving an f1-score of 0.69 for detecting the minority class (the anomalous class). Additionally, the experiments also indicate that some models are more sensitive to hyperparameter tuning than others such as XGBOOST, CNN, BI-LSTM and LSTM with a 6-7% increase in f1-scores from their respective base models. In general, all the models have a better performance when both hyperparameter tuning and class weighting are applied together.

This research has relevance for both predictive audit and PBPM research domains. In this research, a procedural method is developed that demonstrates how predictive audit can be enabled in practice through a practical use case. This is especially important considering that the adoption of advanced data and analytics in audit has been generally low. This study serves as a first step towards that acceptance.

Contents

List of Acronyms

PBPM	Predictive Business Process Monitoring
DT	Decision Trees
RF	Random Forest
XGBOOST	Extreme Gradient Boosting
DNN	Deep Neural Networks
RNN	Recurrent Neural Networks
CNN	Convolutional Neural Networks
LSTM	Long Short Term Memory
ReLU	Rectified Linear Unit
BI-LSTM	Bidirectional Long Short Term Memory
SVM	Support Vector Machines
MLR	Multinomial Logistic Regression
DSRM	Design Science Research Methodology
BPMN	Business Process Model Notation

List of Tables

List of Figures

Chapter 1

Introduction

This chapter introduces the research and provides a general overview for the entire thesis. In Section 1.1, the problem is identified and motivated. Section 1.2 provides some background information to the concepts introduced in Section 1.1. In Section 1.3, the research methodology that is used to carry out the research is described. Following this, the research objectives and research questions are outlined in Section 1.4. The contributions of the research are presented in Section 1.5. Finally, the entire structure of the thesis is illustrated in Section 1.6

1.1 Problem Identification

1.1.1 Problem Context

The auditing profession is a highly complex one with the role of helping organisations achieve their business objectives, manage their risk and improve their business performance. The auditor's responsibility is to ensure that financial statements produced meet legal standards. More concretely, for activities listed in the financial report, auditors identify and assess any risks which could have a significant impact on the financial position or financial performance, and also the internal controls that the organisation has put in place to mitigate those risks [6]. Considering that financial statements are essentially a summary of business transactions that are encapsulated in business processes of an organisation, it makes sense why monitoring business processes that form the very basis of financial statements can be used as a credibility check for financial statements. For this very reason, process mining, in particular, as an analytic method has sparked interest in the auditing community. Process mining was specifically developed to monitor and improve business processes. Even large external audit firms including the Big Four, have embarked on the process mining journey either through their own in-house tools or commercially available tools.

1.1.2 Problem Statement

When researchers made "a case for process mining in audit" (see Section 1.2.2), one of the main motivations stated was "process mining analyses the entire population of data and not just a sample" which was aimed at tackling the main drawback of the traditional audit methodology in the first place (see Section 1.2.1). However, the website of a large audit firm to-date states, "audits are based on selective testing only" [6]. Clearly, this drawback has not yet been fully mitigated in practice. This could mean that process mining is not fully leveraged in the audit profession. Moreover, for years, researchers have advocated for predictive audit or more forward looking approaches to audit (see Section 1.2.1). Currently, process mining in audit is conducted by analysing event logs through a process mining visualization tool. The use of event logs can go beyond this and can enable predictive audit.

Predictive Business Process Monitoring (PBPM) is a growing discipline that has the potential to address these two problems. It combines elements from both process mining and advanced analytical methods such as machine learning and deep learning. This combinative approach can enable predictive audit. To this end, this research will focus on developing a procedural method that demonstrates how machine learning and deep learning approaches employed in PBPM literature can be applied to a practical audit use case - *classifying anomalous activities in an audit log*. An intuition for this will be introduced in Section 1.2.3.

1.2 Background

1.2.1 Predictive Audit

Predictive audit is an approach for performing continuous auditing that applies prediction methodology to the audit [1]. It utilises historical and/or current data to predict potential future outcomes. Then, methods such data mining and machine learning can be leveraged to create a predictive model. Consequently, the generated predictive models can classify processes or sets of processes that may deviate from predefined control flows. An illustration of how such a model could be created is depicted in Figure 1.1. Results of such an audit could be used to identify discrepancies in transactions and those transactions may be further investigated by auditors. The biggest advantage of this approach lies in the continuity. Since, this approach enables auditing to be done more frequently or continuously rather than the traditional monthly, quarterly or annual audits, the time lag between the event and the time of associated assurance could be avoided. Moreover, this approach involves acting on the entire transaction

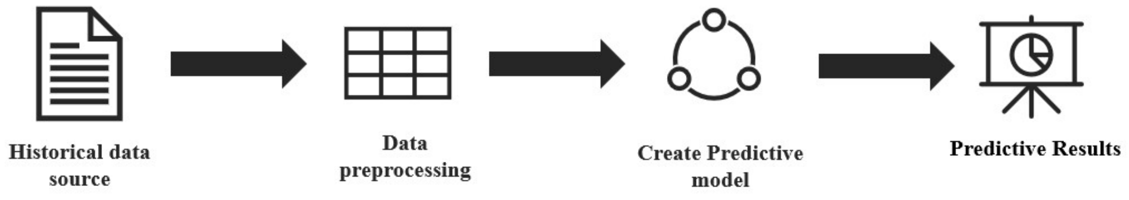


Figure 1.1: Predictive Model Formulation adapted from [1]

Table 1.1: The main components of an event log.

Components	Description
Case	It is represented by a `CASEID` which refers to a unique case or process instance. For e.g. sales order number, purchase order number or a order line identifiation number.
Activity	Every case or process instance consists of a series of occurring process steps which are referred to as activities.
Timestamp	The actual time when each process step has been executed.

population rather than a few representative samples [1].

1.2.2 Process Mining for Audit

Process mining is a technique of analysing data recorded by an organisation's information system. The data is recorded in the form of *event logs* or *audit trails*, which are defined as a "chronological record of computer system activities which are saved to a file on the system which can later be reviewed by the system administrator to identify users actions on the system or processes which occurred on the system" [7]. Table 1.1 shows the necessary components of an event log. An event log can optionally include other information such as a resource i.e., the person or party responsible for performing the activity, cost information, etc.

Process mining enables the monitoring of business process that form the basis of financial statements by ensuring the processes are under control from an efficiency and compliance point of view [8]. In order to better understand how a business process is linked to the financial statements, let's consider an example of a *purchase-to-pay* process borrowed from [8] illustrated in Figure 1.2. This process has a certain control-flow which has to be adhered to, for instance, *second approval of purchase order* follows *first approval of purchase order*. With the help of process mining, control-flows of core business processes within an organisation can be audited to determine if the financial

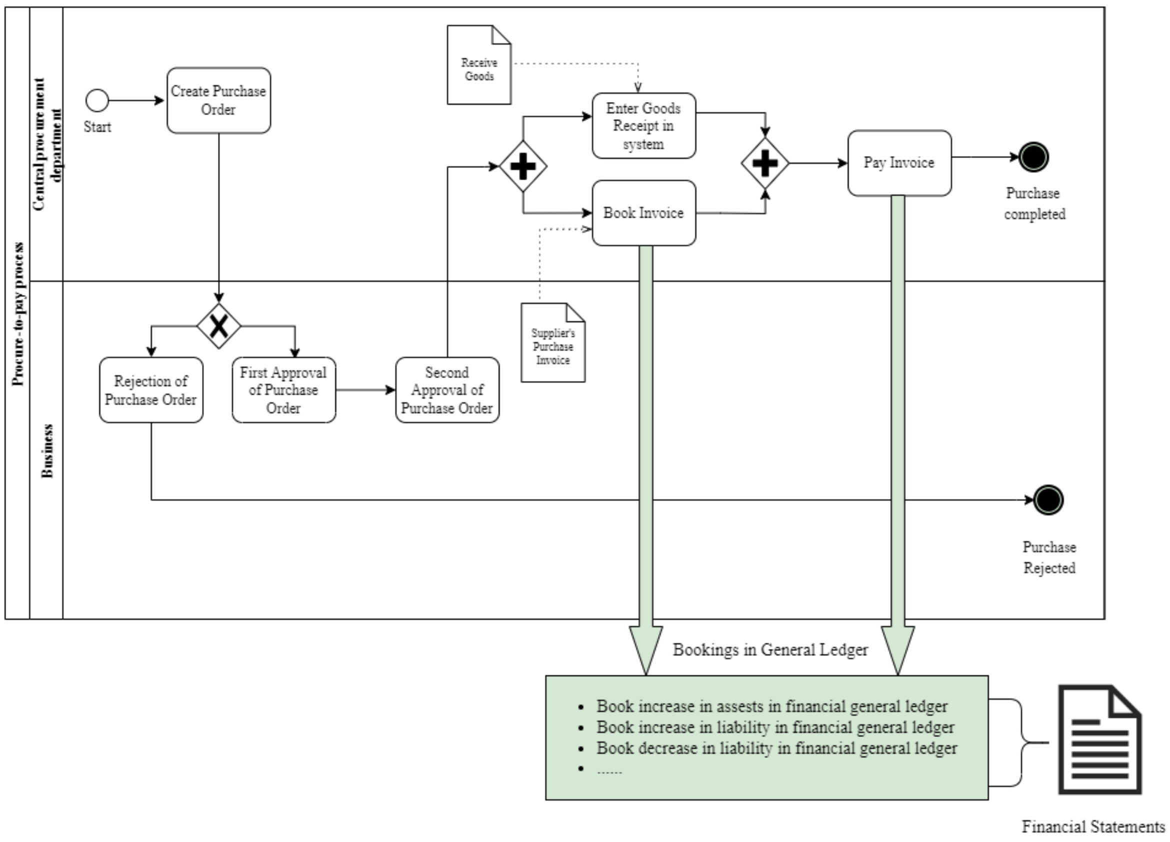

Procure-to-pay process
Central procurement department
Business
Start
Create Purchase Order
Receive Goods
Enter Goods Receipt in system
Book Invoice
Pay Invoice
Purchase completed
Supplier's Purchase Invoice
Rejection of Purchase Order
First Approval of Purchase Order
Second Approval of Purchase Order
Purchase Rejected
Bookings in General Ledger
Book increase in assests in financial general ledger
Book increase in liability in financial general ledger
Book decrease in liability in financial general ledger
......
Financial Statements

Figure 1.2: Purchase-to-pay process linked to the financial statement [1]

statements recorded are valid. Just like financial processes, operational and management processes can also be audited [8].

Another benefit of process mining is that it can leverage all the data recorded by the information system which, given the advancements in current information systems, have the potential to recreate the entire business process as it occurred in the past. This allows the auditor to make use of the entire transaction population rather than a few representative samples. However, the most important benefit that process mining has to offer for the auditing profession is that the auditor is no longer restricted to data entered by the auditee, but possesses an independent view about the circumstances under which the auditee made those entries. For a long time, the auditor had to rely on the transactions coming from entries made by the employees of the auditee, and the auditor had no way of independently verifying who made those ledger entries and when they did so [9].

In sum, the key sources of value of process mining in audit can be formulated as follows [9]:

- Process mining analyses the entire population of data and not just a sample;

- The data captured by event logs are independent of the actions of the auditee whose behaviour is the subject of the audit;
- Process mining allows the auditor to conduct analyses not possible with existing audit tools, such as discovering the ways in which business processes are actually being carried out in practice;
- Process mining allows for more effective ways of conducting the required walk-throughs of processes and conducting analytic procedures.

1.2.3 Using Predictive Business Process Monitoring for Classifying Anomalous Traces

In predictive audit, historical knowledge of processes is used to predict risks, control trends, level and flows, and other parameters of the business process [1]. This is naturally facilitated by process mining in that past activities of processes can be used to determine if a new activity in a process is being executed wrongly and can consequently be marked as erroneous in realtime. If we consider the *purchase-to-pay* example in section 1.2.2, a predictive model can learn from previous processes, for instance, that *second approval of purchase order* follows *first approval of purchase order*. If any other activity follows *first approval of purchase order*, it is seen as an irregular observation and can be marked as faulty or anomalous.

Predictive Business Process Monitoring (PBPM), first introduced by [10] as a framework, is now a growing body of research based on the continuous generation of predictions and recommendations based on the analysis of past execution traces. In particular, event logs are exploited to predict the outcome of each case given its current (incomplete) execution trace and given a set of traces of previously completed cases.The predictions often depend on the sequence of activities executed in a given case, but can also depend on the values of data attributes after each activity execution in a case, provided such information is available. Concerning the creation of the prediction model, PBPM literature distinguishes between two kinds of prediction models according to its process awareness. A predictive model is process-aware if it exploits an explicit representation of the process model to make the prediction (e.g., an annotated transition system, or a stochastic Petri net [11]. Instead, a non-process-aware predictive model does not make use of an explicit representation of a process model and is usually some form of machine learning or deep learning model. The outcome of the prediction may be the fulfillment of a compliance rule, a performance objective (e.g., maximum allowed cycle time) or business goal, or any other characteristic of a case that can be determined upon its completion [12].

1.2.3.1 How is an anomaly classified?

In this research, anomaly classification is formulated as a next activity prediction problem. In next activity prediction, a classifier is trained on an independent set of features as input where the unique activities in the previous trace executions are considered labels during training. This essentially makes next activity prediction a multi-class classification problem. Upon testing, the trained model emits a probability for every activity to follow a particular incomplete trace of events, and the activity with the highest probability is marked as the predicted value. Activities with the least probabilities of following an incomplete trace is considered irregular and marked as anomalous.

1.3 Research Methodology

This research makes use of the Design Science Research Methodology (DSRM) proposed by [2]. An adaptation of the DSRM for this research is depicted in Figure 1.3. Although the process is structured in a sequential order, the authors posit that a researcher can start from any of the entry points depicted in Figure 1.3. This research takes a design and development centered approach. According to the authors, such an approach results from the "existence of an artifact that has not yet been formally thought through as a solution for the explicit problem domain in which it will be used" [2]. The idea for the artefact could have emerged from another research domain, used to solve a different problem or may have appeared as an analogy. The application of the DSRM to this research is explained below:

Design centered solution. In this research, a procedural method is developed that is inspired by the methods employed in the PBPM research domain. So in this sense, an idea appearing from one research domain (PBPM) has been used to develop a solution for a problem identified in another (predictive audit).

Problem identification and motivation. In section 1.1.2, two problems were identified (1) process mining is not leveraged everywhere in audit practice; (2) lack of predictive audit use cases.

Objectives of a solution. Based on the problems identified, the objective of the solution is to address the two problems by developing a procedural method that demonstrates how process mining can be carried out in a predictive manner through a practical use case for audit.

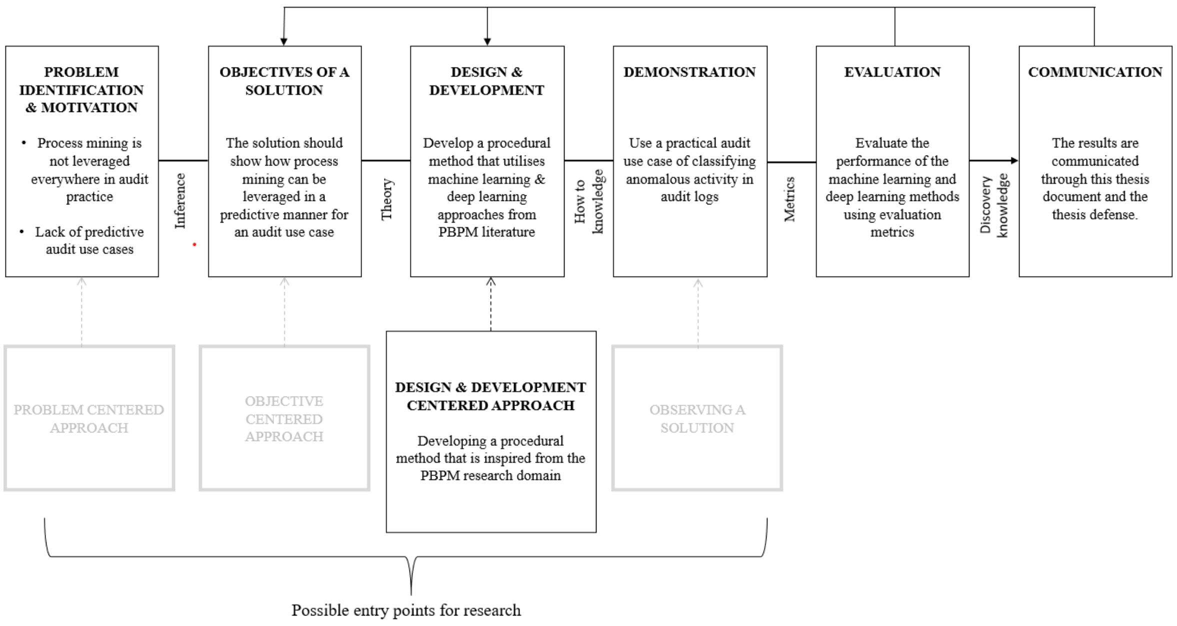

PROBLEM IDENTIFICATION & MOTIVATION
• Process mining is not leveraged everywhere in audit practice
• Lack of predictive audit use cases
Inference
OBJECTIVES OF A SOLUTION
The solution should show how process mining can be leveraged in a predictive manner for an audit use case
Theory
DESIGN & DEVELOPMENT
Develop a procedural method that utilises machine learning & deep learning approaches from PBPM literature
How to knowledge
DEMONSTRATION
Use a practical audit use case of classifying anomalous activity in audit logs
Metrics
EVALUATION
Evaluate the performance of the machine learning and deep learning methods using evaluation metrics
Discovery knowledge
COMMUNICATION
The results are communicated through this thesis document and the thesis defense.
PROBLEM CENTERED APPROACH
OBJECTIVE CENTERED APPROACH
DESIGN & DEVELOPMENT CENTERED APPROACH
Developing a procedural method that is inspired from the PBPM research domain
OBSERVING A SOLUTION
Possible entry points for research

Figure 1.3: Design Science Research Methodology for this research adapted from [2]

Design and development. In this research, a procedural method is developed which utilises machine learning and deep learning models adopted from PBPM literature. In this research, this is referred to as the *artefact.*

Demonstration. To demonstrate the use of the solution to the specified problem, a practical audit use case is defined namely, classifying anomalous activity in a loan application process.

Evaluation. To evaluate the performance of the developed models, certain evaluation measures will be used.

Communication. Finally, the results are communicated through this thesis document.

1.4 Research Goal & Research Questions

The goal for the research is formulated as follows:

RG: To develop a procedural method that utilises machine learning approaches obtained from PBPM literature to classify anomalous traces in audit logs

Since classifying anomalies is formulated as a next activity prediction problem (see section 1.2.3.1), the following research questions are formulated accordingly to achieve the defined research goal:

RQ1: What is PBPM and what machine learning and deep learning algorithms have been used for next activity prediction in PBPM literature?

RQ2: How can these methods be classified?

RQ3: What evaluation measures exist in PBPM literature for next activity prediction?

RQ4: How do the methods identified in RQ1 perform when evaluated using the measures identified in RQ3?

1.5 Research Relevance

1.5.1 Scientific Relevance

This research has relevance for both predictive audit and PBPM research domains. For the former, this research contributes by developing a procedural method that demonstrates how predictive audit can be enabled in practice. This is especially important because even though predictive audit as a concept, or any form of advanced data analytics for that matter, was introduced in audit research more than ten years ago, its acceptance has been low in the audit profession [13] [14]. Additionally, this is reflected in the recent call for papers for big data and analytics in modern audit engagements [15].

For the domain of PBPM, this research contributes by an empirical review of next activity prediction methods (see Chapter 2). So far this has only been done for remaining time prediction [16] and binary outcome prediction [17].

1.5.2 Practical Relevance

Through a practical use case, this research demonstrates how process mining mitigates the problem of selective testing and uses all the data available in information systems to make an analysis. Moreover, it goes one step further and demonstrates how process mining can be leveraged in a predictive manner. Currently, process mining is carried out retrospectively and hopefully, this research provides inspiration for unique applications of process mining for the audit profession.

1.6 Thesis Structure

The structure of the thesis is depicted in Figure 1.2

Table 1.2: Thesis Structure

Thesis Chapter	Description	DSRM Mapping	Research Questions
Chapter 1	An overview of the entire research including motivating the problem, choosing the research methodology, formulating the research objective, research questions & research relevance	Steps 1 & 2	
Chapter 2	The theoretical basis for the solution used to address the problems outlined in the problem statement	Step 3	RQs 1, 2 & 3
Chapter 3	Experiment/Artefact demonstration, evaluation and overview of results	Step 4, 5 & 6	RQ 4
Chapter 4	Discussion of results & study limitations	Step 6	All
Chapter 5	Conclusions & future work	Step 6	All

Chapter 2

Literature Review

In this chapter, the main aim is to find answers to research questions 1, 2 & 3 defined in Section 1.4. For this, a Systematic Literature Review (SLR) has been conducted according to the guidelines presented by [3]. According to the authors, all reviews can be conducted following eight steps: *(1) formulating the research problem; (2) developing and validating the review protocol; (3) searching the literature; (4) screening for inclusion; (5) assessing quality; (6) extracting data; (7) analyzing and synthesizing data; (8) reporting the findings.* The steps are illustrated in Figure 2.1 :

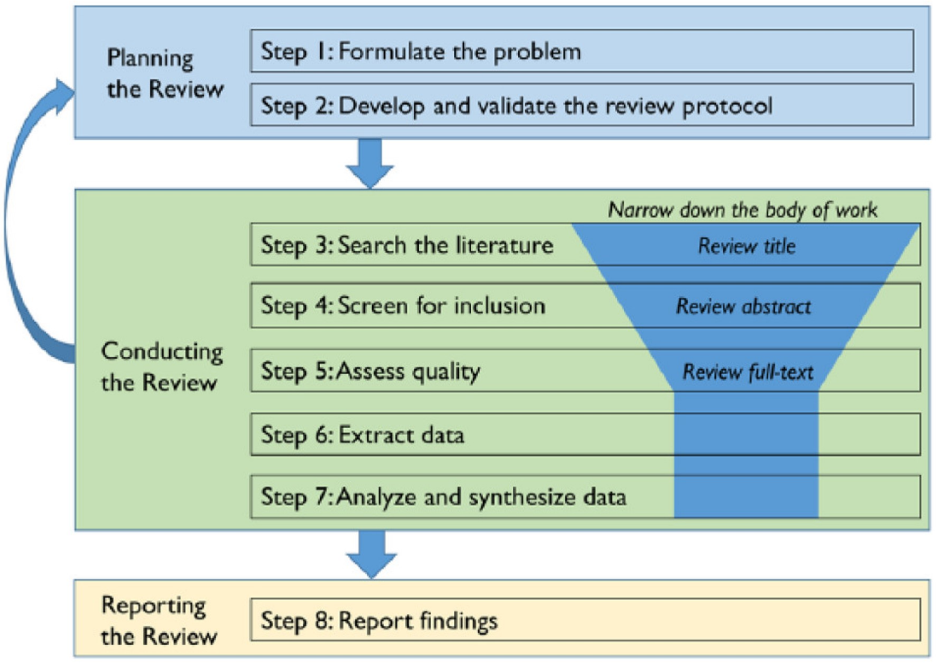

Planning the Review
Step 1: Formulate the problem
Step 2: Develop and validate the review protocol
Narrow down the body of work
Step 3: Search the literature
Review title
Step 4: Screen for inclusion
Review abstract
Conducting the Review
Step 5: Assess quality
Review full-text
Step 6: Extract data
Step 7: Analyze and synthesize data
Reporting the Review
Step 8: Report findings

Figure 2.1: Process of SLR [3]

2.1 Formulating the Problem

According to [3], this step involves defining the research questions that will guide the literature review process since literature reviews are essentially research inquiries. This literature review is aimed at answering the first three research questions defined in Section 1.4

RQ1: What is PBPM and what machine learning and deep learning algorithms have been used for next activity prediction in PBPM literature?

RQ2: How can these methods be classified?

RQ3: What evaluation measure exist in PBPM literature for next activity prediction?

2.2 Develop and Validate the Research Protocol

The review protocol should describe all the elements of the review, including the purpose of the study, research questions, inclusion criteria, search strategies, quality assessment criteria and screening procedures, strategies for data extraction, synthesis, and reporting [3]. These components will be discussed in the following sections, and so, a separate review protocol is not created.

2.3 Search the Literature

2.3.1 Channels for Literature Search

The following electronic databases are used to carry out the literature review:

- Scopus
- Web of Science
- IEEE
- ACM

2.3.2 Search Terms

The search terms used to initiate the SLR are 'business process', "monitoring" and "prediction".

2.3.3 Sampling Strategy

The purpose of this review is to first gain an initial understanding of the domain of predictive business process monitoring, before delving deeper into the literature pertaining to the defined research questions. Therefore, a broader search string is defined to reflect this:

TITLE-ABS-KEY (”business process” AND ”monitoring” AND ”prediction”)

Following this, the papers are scanned to only include literature relevant to the research questions defined. Furthermore, reports and theses, referred to as gray literature by [3], are excluded from the review. Additional criteria for inclusion and exclusion are outlined in the following sections.

2.4 Screen for Inclusion

2.4.1 Inclusion Criteria

Only papers satisfying the following inclusion criteria are included in this review:

- The articles should be fully published and written in English;
- The articles should belong to the disciplines of Computer Science, Decision Science, Engineering, Mathematics and Business Management;
- The articles should be academic in nature (e.g., articles, conference papers, book chapters and review articles);
- Only articles that are openly accessible or can be accessible through the university VPN are included;
- The title and abstract of the articles should contain the search terms defined in Section 2.3.2.

2.4.2 Exclusion Criteria

In addition to the inclusion criteria, the initial search results are subjected to the following exclusion criteria:

- The abstracts of articles that concern theoretical developments (e.g., frameworks, methodologies, taxonomies and untested prototypes) are excluded to only account for fully developed prediction algorithms that are evaluated;
- Studies that do not include business process execution logs or event logs are excluded.

2.5 Assess Quality

Articles satisfying all the inclusion and exclusion criteria are downloaded for further assessment. All the sources are managed through the reference management software Zotero. In order to refine the full-text articles and to prepare the final pool of studies for data extraction and analysis, the following criteria are defined:

- All duplicates are removed;
- Studies that are similar to each other i.e., are written by the same authors and have similar titles either indicate extensions to previous work or are a complete version of a prior incomplete version. These articles are carefully examined to avoid redundancy;
- The articles should concern a prediction problem and provide a solution for the same. Articles providing only an analysis (e.g., visualization) and not a prediction are removed;
- Studies where the developed prediction algorithms are not evaluated or validated are removed;
- Studies where the developed prediction algorithms are not properly documented i.e., the underlying concepts are not explained are removed;
- Literature reviews are archived separately. The purpose of this is to ensure that all the existing concepts of predictive business process monitoring are roughly covered, This not only builds on existing reviews, but makes this review more exhaustive.

2.6 Extracting data

At this stage, all the papers are considered relevant and are downloaded for reading. It is important to note that, even at this point, the literature is quite broad. This is deliberate to first outline the general concepts. Following this, only the papers pertaining to the research questions defined in 2.1 will be considered.

2.7 Analyzing and Synthesizing Data

The final papers are categorized according to the steps in figures 2.3 & 2.4. An overview of the classification can be seen in Table 2.1.

2.8 Report Findings

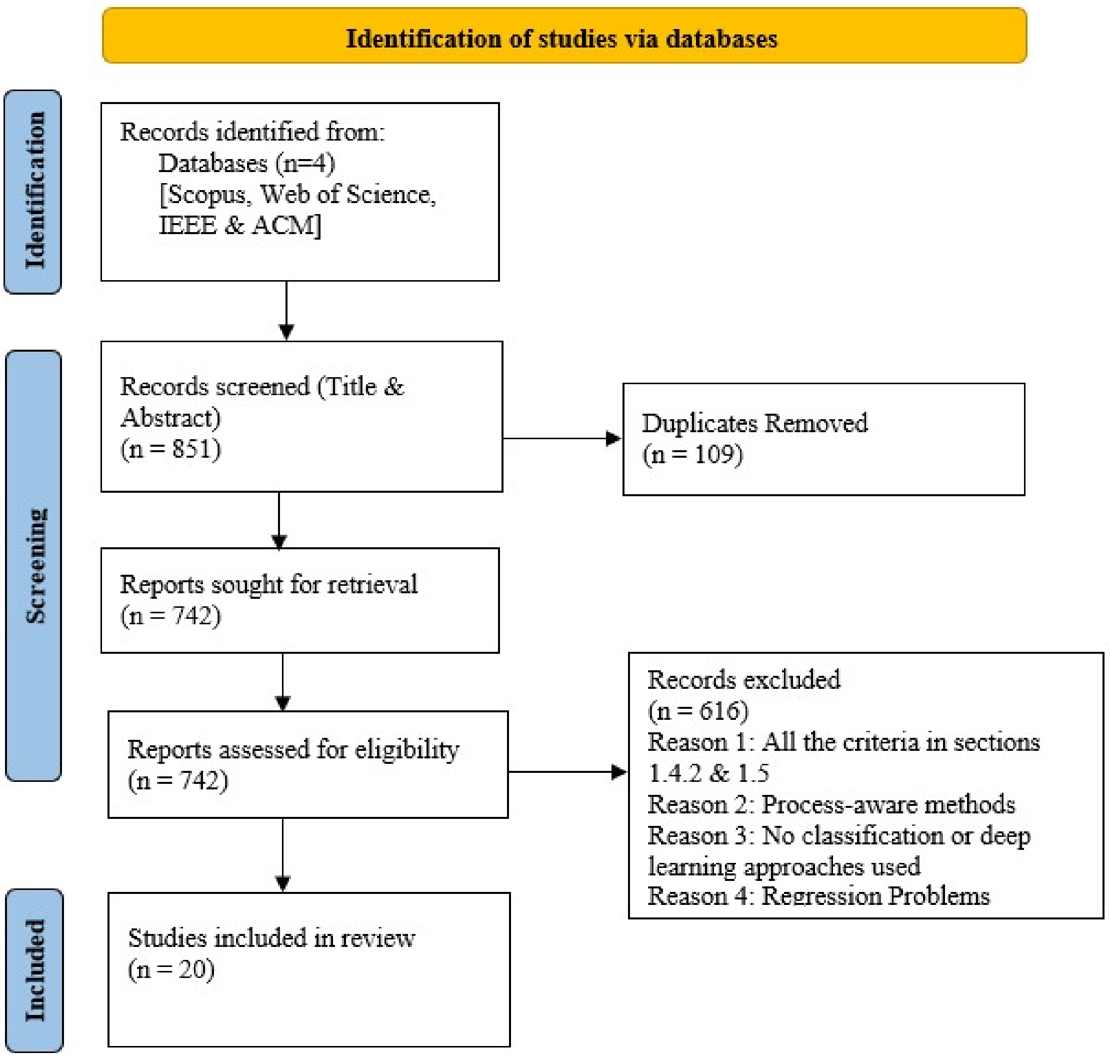

Identification of studies via databases
Identification
Screening
Included
Records identified from: Databases (n=4) [Scopus, Web of Science, IEEE & ACM]
Records screened (Title & Abstract) (n = 851)
Duplicates Removed (n = 109)
Reports sought for retrieval (n = 742)
Reports assessed for eligibility (n = 742)
Records excluded (n = 616) Reason 1: All the criteria in sections 1.4.2 & 1.5 Reason 2: Process-aware methods Reason 3: No classification or deep learning approaches used Reason 4: Regression Problems
Studies included in review (n = 20)

Figure 2.2: Prima diagram visualizing the screening process [3]

2.8.1 Predictive Business Process Monitoring (PBPM)

According to [18], predictive business process monitoring refers to a ”family of online process monitoring methods that seek to predict as early as possible the outcome of a case given its current (incomplete) execution trace and given a set of previously completed cases”. Here, the outcome can be any attribute that can be predicted given an event log. Some of the different types of outcomes studied in literature are next activity [19] [20] [21] [22], remaining time [19] [23] [24] and risk [25] [26].

Based on the final value of prediction - numerical or categorical, all predictions addressed in literature can be broadly classified either as regression or classification problems. In this thesis, the focus will only be on next activity prediction which is essentially a classification problem where the outcome to be predicted is categorical in nature. Furthermore, some studies focus on the prediction of a sequence of activities or so-called "suffixes", for instance, the study by [27] which will not be addressed in this thesis. In addition to the kind of prediction problem, another kind of classification can be deduced depending on whether or not studies make use of explicit representations of processes such as petri nets, instance graphs etc. for prediction, and can accordingly be classified as "process-aware" or "non-process aware" [28]. The focus of this thesis will be the latter since studies addressing classification problems largely fall under this category [28].

2.8.2 Preliminaries

The following definitions and notations have been borrowed from [29]

Definition 2.8.1 (Event Log, Event, Trace)**.** An event log L contains events. An event e is a tuple $e = \langle c, a, t, r, (d_1, v_1), ..., (d_m, v_m)\rangle$, where c is the case id, a is the activity to which the task recorded by this event belongs, t is the timestamp at which the event has been recorded, r is the resource that executed the task and $(d_1, v_1), ..., (d_m, v_m)$, with $m \geq 0$ are other domain specific attribute or context specific attributes and their values. The universe of all events is denoted by $\mathcal{E}$ .The sequence of events generated in a given case forms a trace $\sigma = \langle e_1, ..., e_n\rangle$, where $\forall i \in [1, n], e_i \in \mathcal{E}$ and $\forall i, j \in [1, n], e_i.c = e_j.c$, i.e., all events of a trace belong to the same case, and $\forall i \in [1, n-1], e_i.t \leq e_{i+1}.t$, i.e., events in a trace can be ordered in time using the timestamp attribute. The universe of all traces is denoted by $\mathcal{S}$.

Definition 2.8.2 (Prefix)**.** The prefix function $P_l(\sigma)$ is used to extract the prefix logs L^*. Given a trace $\sigma = \langle e_1, ..., e_n\rangle$ and a positive integer $l \leq n, P_l(\sigma) = \langle e_1, ..., e_l\rangle$. For an event log L, its prefix log L^* is the event log that contains all prefixes of L, i.e., $L^* = \{P_l(\sigma) : \sigma \in \mathcal{S}, 1 \leq l \leq |\sigma|\}$. For example, for a sequence $\sigma_1 = \langle a, b, c, d\rangle, P_3(\sigma_1) = \langle a, b, c\rangle$, and $L^* = \{\langle a\rangle, \langle a, b\rangle, \langle a, b, c\rangle, \langle a, b, c, d\rangle\}$

Definition 2.8.3 (Prefix Bucket)**.** A prefix bucketing B_N of size N is a partition of the prefixes P in N subsets, that is, $B_N(P) = \{B_i\}_{i=1...N}$ with $\bigcup_i B_i = P$ and $\bigcap B_i = \emptyset$

Definition 2.8.4 (Labeling Function)**.** A labeling function $y : \mathcal{S} \longrightarrow \mathcal{Y}$ is a function mapping a trace $\sigma \in \mathcal{S}$ (or any prefix derived from it) to its class label $y(\sigma) \in \mathcal{Y}$, with $\mathcal{Y}$ being the domain of the class labels.

Definition 2.8.5 (Sequence Encoder)**.** A sequence or trace encoder $f : \mathcal{S} \longrightarrow \mathcal{X}_1 \times ... \times \mathcal{X}_P$ is a function that takes a (partial) trace σ and transforms it into a feature vector in a $\mathcal{D}$-dimensional vector space $\mathcal{X}_1 \times ... \times \mathcal{X}_D$ with $\mathcal{X}_d \subseteq \mathbb{R}, 1 \leq d \leq D$ being the domain of the d-th feature.

Definition 2.8.6 (Classifier)**.** A process outcome classifier *poc* is defined by a label predictor function *lp* that assigns a class label to a feature vector, i.e, $lp : \mathcal{X}_1 \times ... \mathcal{X}_P \longrightarrow \mathcal{Y}$

2.8.3 General Workflow for PBPM

Figures 2.3 and 2.4 from [4] depict the general workflow of predictive business process monitoring.

The process typically consists of two phases: the offline phase and the online phase. In the offline phase, case prefixes from the event log are extracted to create a prefix log (see section 2.8.2), and this generally serves as input data. The prefix log would essentially be an historical log with incomplete traces rather than fully complete traces. Since the final prediction would be made on (incomplete) running traces, the approach of using a prefix log seems intuitive. The second step in the offline phase is prefix bucketing, where similar prefixes are grouped together in buckets. Each prefix bucket is then encoded into feature vectors and a classifier is trained for each prefix bucket. In some studies, prefix bucketing is not adopted, and a classifier is trained on the entire prefix log.

In the online phase, the model generated in the offline phase is applied on the encoded running trace. If prefix bucketing is used to train the classifier in the offline stage, then given the running trace and buckets of historical prefixes, the running trace is matched to the appropriate bucket and then encoded. Finally, the model trained in the offline phase provides a prediction for the running trace.

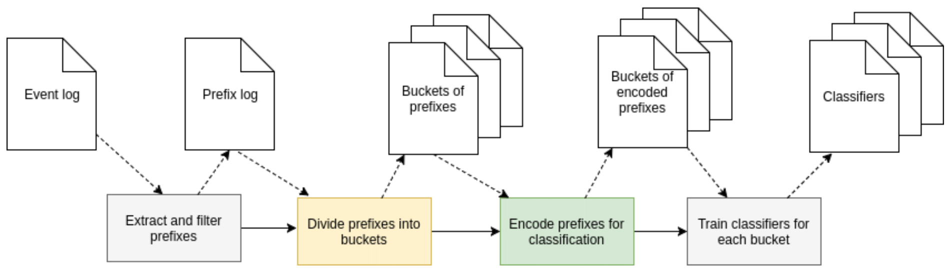

Event log
Prefix log
Buckets of prefixes
Buckets of encoded prefixes
Classifiers
Extract and filter prefixes
Divide prefixes into buckets
Encode prefixes for classification
Train classifiers for each bucket

Figure 2.3: Predictive business process monitoring workflow (offline) [4]

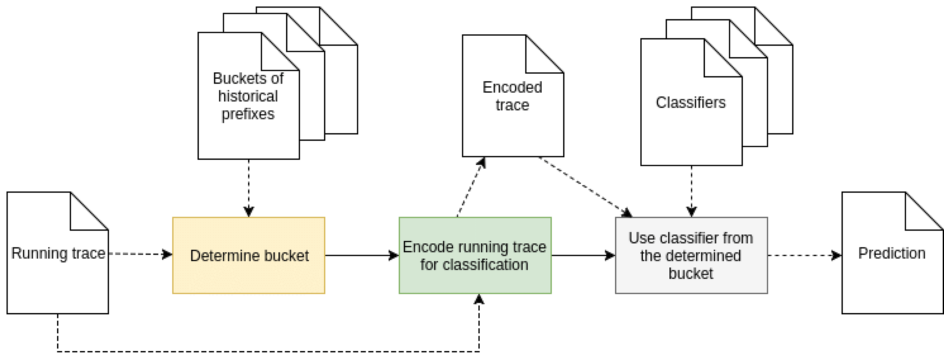

Buckets of historical prefixes
Encoded trace
Classifiers
Running trace
Determine bucket
Encode running trace for classification
Use classifier from the determined bucket
Prediction

Figure 2.4: Predictive business process monitoring workflow (online) [4]

Just like [4], the papers surveyed in this review are classified based on the approaches taken while undertaking the steps depicted in figures 2.3 & 2.4. Table 2.1 gives an overview of the classification of papers.

2.8.3.1 Prefix Extraction

Most of the studies use a prefix log (see section 2.8.2) as input data rather than the entire event log. Given that at runtime (online phase), the prediction model is supposed to generate a prediction based on an incomplete trace, it is only natural that the model be trained on similar data during the offline phase. Prefix extraction is approached differently in the papers surveyed for this review. Some studies use all possible prefixes, while others use prefixes only up to a certain length. The reason for this is that the large number of prefixes greatly slows down the training process. Another noticeable observation is that in deep learning approaches, prefixes of different sizes are converted to a fixed length because the input vector has to be of the same size. For this, a fixed length is defined, for instance, length of the largest trace [20], or average length of all the traces [30]. Any prefix trace smaller than this is padded with zeros [31] [32] [33], while larger prefixes are broken down into smaller ones with minimum length 2.

2.8.3.2 Prefix Bucketing

Two main approaches to prefix bucketing have been noticed in the surveyed literature. One approach is *clustering* where each bucket represents a cluster that is formed by applying a clustering algorithm and a classifier is trained on each bucket or cluster. At runtime, the running trace is matched to the cluster based on similarity and the corresponding classifier is used. The other approach is *prefix-length* buckets where each bucket contains prefixes of a specific length, and a classifier is trained on each bucket. In studies where no bucketing approach is applied, all prefix traces are placed in a single bucket on which the classifier is trained on.

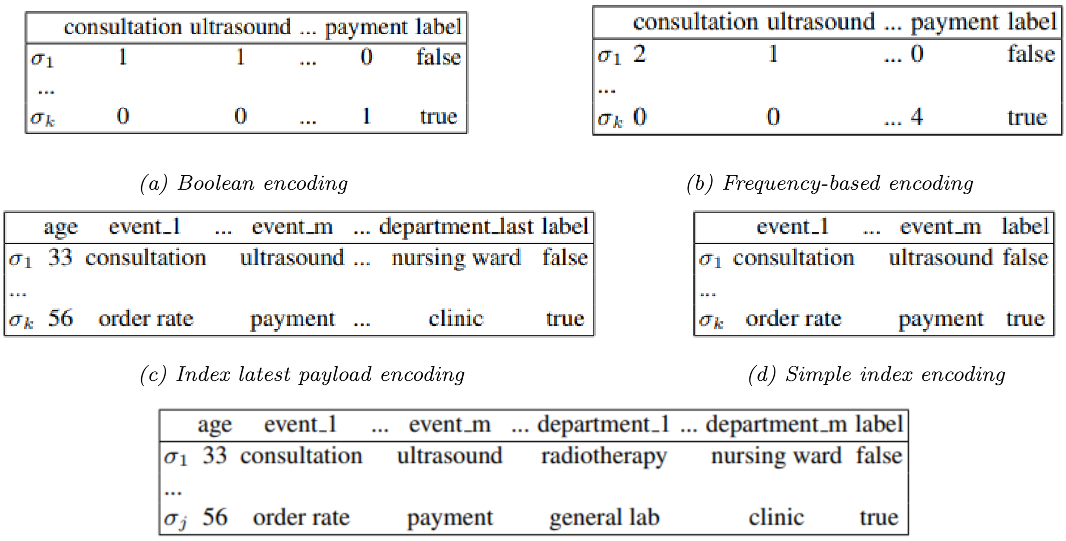
2.8.3.3 Sequence Encoding

In order to train a classifier, all the prefixes need to be encoded into a feature vector. Each trace in a log consists of a sequence of events indicating its control-flow. Moreover, each event in a trace is also supported with additional attributes such as the timestamp of the activity or the resource who carried out the activity corresponding to a particular event. In this setting, the attributes that remain the same throughout the case are referred to as static attributes, while the attributes that change throughout the case are referred to as dynamic attributes [5], Normally, features that account for the control-flow of events in a case are commonly referred to as control-flow features, while the features that correspond to the attributes of an event in a case form the "data-payload" corresponding to an event. In the most simple and straightforward encodings, only the control-flow of events in a trace are encoded as features, excluding the data payload associated with an event.

Two such encoding schemes identified by [5] that only consider control-flow features are *boolean* encoding and *frequency-based* encoding. In *boolean* encoding, a prefix $\mathcal{P}_i$ is represented through a feature vector $g_i = (g_{i1}, g_{i2,..g_{ih}})$, where if g_{ij} corresponds to the event class e, then $g_{ij} = 0$ if e is not present in σ_i and $g_{ij} = 1$ is e is present in $\mathcal{P}_i$. This technique is also commonly referred to as "one-hot encoding" in many papers. *Frequency-based* encoding, instead of boolean values, represents the control flow in a case with the frequency of each event class in the case [5]. In Table 2.1, an encoding technique is categorized as "agg" or aggregation if it uses some sort of aggregation based encoding other than boolean or frequency encoding and doesn't consider the control-flow of events. Sometimes, each unique event class is represented as a categorical variable. In Table 2.1, such an encoding technique is represented as 'categorical'. This can also be seen as a form of aggregate encoding

While the above two encoding techniques utilise information from all the events in a trace, there is still information loss since the order of events is not considered. *Simple index* encoding [5], on the other hand, takes into account the order in which the event occurs in a sequence. Each feature corresponds to a position in the sequence and the values of each feature are the event classes.

None of the aforementioned techniques consider the data payload associated with each event or the evolution of the data payload throughout the sequence (data-flow). Techniques such as *index latest payload* encoding and *index-based* encoding, the latter being developed by [5], additionally consider the corresponding data payload. In *index latest payload* encoding (or just *latest payload* when the flow is not considered), the data payload associated only with the last event in the sequence is considered, while in *index-based* encoding, the attributes associated with all the events in the sequence are considered. This allows for the feature vector to account for the evolution of attributes corresponding to an event throughout the sequence. The different forms of sequence encoding as visualized in [5] are depicted in Figure 2.5.



	consultation	ultrasound	...	payment	label
σ_1	1	1	...	0	false
...					
σ_k	0	0	...	1	true

(a) Boolean encoding

	consultation	ultrasound	...	payment	label
σ_1	2	1	...	0	false
...					
σ_k	0	0	...	4	true

(b) Frequency-based encoding

	age	event_1	...	event_m	...	department_last	label
σ_1	33	consultation		ultrasound	...	nursing ward	false
...							
σ_k	56	order rate		payment	...	clinic	true

(c) Index latest payload encoding

	event_1	...	event_m	label
σ_1	consultation		ultrasound	false
...				
σ_k	order rate		payment	true

(d) Simple index encoding

	age	event_1	...	event_m	...	department_1	...	department_m	label
σ_1	33	consultation		ultrasound		radiotherapy		nursing ward	false
...									
σ_j	56	order rate		payment		general lab		clinic	true

(e) Index-based encoding

Figure 2.5: Different forms of sequence encoding [5]

2.8.3.4 Classification Algorithms in PBPM literature

Decision Treess (DTs) and their ensembles

A DT is a tree-structured classification model, that as the name suggests, uses a tree-like structure to display predictions based on series of feature splits. It is one of the oldest and most popular technique for learning discriminatory models and dates back to the early work of [34]. DTs are learned in a top-down fashion, with an algorithm known as *Top-Down Induction of Decision Trees (TDIDT)*, *recursive partitioning*, or *divide-and-conquer* learning [35]. In the beginning, the entire training sample is considered a root node, after which the algorithm selects the best attribute to be the *root* node for the first split. For this, the *impurity* of each attribute is calculated. This is the degree to which the attribute contains examples that belong to a single class. Common impurity measures include entropy and the Gini index:

$$Entropy(S) = -\sum_{i=1}^{c} \frac{|S_i|}{|S|} . \log_2 \left(\frac{|S_i|}{|S|} \right)$$

$$Gini(S) = 1 - \sum_{i=1}^{c} \left(\frac{|S_i|}{|S|} \right)^2$$

where S is the set of training examples, and S_i is the set of training examples that belong to class c_i. For a two class problem, when the different classes are equally distributed in the training sample, then the impurity or entropy is maximum at 1 and when the sample has examples of a single class, the impurity is 0.

A good attribute divides the dataset into subsets that are as pure as possible, ideally into homogeneous subsets i.e., subsets containing training examples from the same class. So, to determine the best attribute for the root node, the attribute is selected that provides the highest decrease in average impurity, also known as the *gain* [35]:

$$Gain(S, A) = Impurity(S) - \sum_t \frac{|S_t|}{|S|}.Impurity(S_t)$$

where S_t is value of the attribute A and $|S_t|$ depicts the number of occurrences of a particular value of attribute A.

After the first split, the procedure is applied recursively to each of the resulting samples. If a set contains only examples from the same class, or if no further splitting is possible, the corresponding node in the tree is turned into a leaf node and labeled with the respective class. Otherwise, an interior node is added and associated with the best splitting attribute for the corresponding set using the steps described above. Thus the original training sample is successively partitioned into non-overlapping, smaller samples until each set only contains examples of the same class or a *pure* node [35].

Although, DTs are quite popular for classification problems, they are quite prone to overfitting in the form of overly complex trees and do not generalise well on test data. This led to the development of several ensemble models with decision trees as base learners. The ensemble approach relies on combining a large number of relatively weak simple models to obtain a stronger ensemble prediction. Bagging and boosting are two common techniques to build ensembles. Bagging, also commonly known as bootstrap aggregation uses random samples of the original training sample with replacement and several weak models are trained in parallel on these random samples independently. The majority of those predictions is determined as the final output prediction. Boosting is similar to bagging in that several weak learners are aggregated to obtain a strong learner. But unlike bagging, weak learners are trained sequentially in an adaptive way, wherein each weak learner added focuses on the examples that the previous weak learners predicted incorrectly. Two popular ensembles using decision trees found in PBPM literature are RF (bagging), proposed by [36] and XGBOOST, proposed by [37].

Random Forest (RF)

In this approach, first a bootstrapped dataset is created from the original training dataset by sampling N training examples at random with replacement, where N is the total number of training examples in the training dataset. This new bootstrapped dataset will be the new training set for growing the tree. Unlike DTs, RF does not use all the input attributes to determine the root node. Instead, if there are M input attributes, a number $m << M$ attributes are selected at random out of M and the best split on these m attributes is used to split the node [38]. In this way, bootstrapping and selection of random subset of input attributes is done multiple times inducing multiple trees. Once the forest is trained in this manner, a new example to be classified is run across all trees. Each tree give a classification for the new test example which is recorded as a vote. The votes from all the trees are combined and the class for which maximum votes are counted is declared as the class of the new example. This is also known as *majority voting* [38]

.

Extreme Gradient Boosting (XGBOOST)

The explanations in this section have been derived from the original paper [37]. XGBOOST is an adaption of gradient boosting. In normal gradient boosting, for a given training dataset with n examples and m features $\mathcal{D} = (x_i, y_i)(|\mathcal{D}| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, a tree ensemble model uses K additive functions to predict the output :

$$\hat{y_i} = \textstyle\sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

where $\hat{y_i}$ is the predicted value and $\mathcal{F} = f(x) = w_{q(x)}(q : \mathbb{R}^m \longrightarrow T, w \in \mathbb{R}^T)$ is the space of regression trees, where each f_k corresponds to an independent tree structure with q leaf nodes and leaf weights w. The main feature of gradient boosting is that the model is trained in an additive manner. Formally, let $\hat{y_i}^{(t)}$ be the prediction of the i-th instance at the t-th iteration, we add a new model such that the following loss is minimised during training:

$$\mathcal{L}^{(t)} = \textstyle\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i))$$

where $f_t(x_i)$ is the new model, or in this case, tree to be added.

For XGBOOST, the authors posit that while focusing on only minimising training loss, one can fit the model very close to the training data and achieve a high training accuracy, the model becomes complex and doesn't generalise well on new data. To counteract this, the authors add a regularisation term $\Omega(f) = \gamma T + \frac{1}{2}\lambda w^2$ that penalises the complexity of the model and helps smooth the final weights to prevent overfitting. The aim of XGBOOST is to then minimise this regularised objective:

$$\mathcal{L}^{(t)} = \textstyle\sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

Since loss functions can get quite complex and cannot be solved simply by, for instance, gradient descent, the authors use second order Taylor approximations to approximate the loss function. Based on this, a scoring function is derived to measure the quality of a tree. This measure behaves analogously to impurity measures in decision trees. For brevity, an in-depth derivation of the scoring function is not included here and the interested reader can have a look at the original paper [37]

Decision trees and their ensembles have been widely explored in PBPM literature. In particular, XGBOOST has been found to perform better than the former in many studies [39] [40]

Multinomial Logistic Regression (MLR)

MLR is a generalis ation of logistic regression to multi-class classification problems. In logistic regression, which is a binary classifier, the output is a probability that a given example belongs to a positive class:

$$P(y=1|x) = f_w(x) = \frac{1}{1+e^{(-w^T x)}} \equiv \sigma(w^T x)$$

and the probability of not belonging to that class can be inferred:

$$P(y=0|x) = 1 - P(y=1|x) = 1 - f_w(x)$$

where $f_w(x)$ is a function of the input feature x and some model parameters or weights w^T

The expression $w^T x$ comes from linear regression, where given linear function $f_w(x) = w^T x$, the goal is to generate a prediction. This is done by finding optimal values for the weights w^T such that the loss function is minimized. However, the outputs of linear regression are not probabilities i.e $\hat{y} \notin [0,1]$. This is the reason that logistic regression makes use of the sigmoid function $\sigma(z) \equiv \frac{1}{1+e^-z}$. After using the sigmoid function, the output of $w^T x$ falls in the range $[0,1]$. The goal of logistic regression is then to find w^T such that the following loss function is minimized:

$$L = -\sum_i (y^i \log(f_w(x))) + (1-y^i)\log(1-f_w(x^i)))$$

The derivation of the loss function can be found in [41]

For multi-class classification, the binary classifier is adapted to output a vector of probabilities for each class rather than a single class and can be represented as :

$$f_w(x) = \begin{bmatrix} P(y=1|x;w) \\ P(y=2|x;w) \\ . \\ . \\ . \\ P(y=K|x;w) \end{bmatrix}$$

Like with the binary classifier, the outputs need to be positive and the probabilities in the range $[0,1]$. For this purpose, the softmax function is used in place of the sigmoid function that converts a vector of K real numbers into a probability distribution of K possible outcomes. The probability of belonging to a particular class k is:

$$P(y=k|x;W) = \frac{e^{W_k^T x}}{\sum_{j=1}^{K} e^{W_j^T x}}$$

where K is the total number of classes and W for a single input is a $1 \times K$ matrix. The loss function to be minimised can be adapted from the binary classifier to the following:

$$\begin{aligned} \mathcal{L} &= -\big[\textstyle\sum_{i=1}^{m}(1-y^i)\log(1-f_w(x^i)) + y^i \log f_w(x^i)\big] \\ &= -\big[\textstyle\sum_{i=1}^{m}\sum_{k=0}^{K} \mathbb{1}(y^1=k)\log P(y^i=k|x^i;w)\big] \\ &= -\left[\textstyle\sum_{i=1}^{m}\sum_{k=0}^{K} \mathbb{1}(y^1=k)\log \frac{e^{w_k^T x^m}}{\sum_{j=1}^{K} e^{w_j^T x^m}}\right] \end{aligned}$$

Then, the model parameters are continuously adapted until the training loss cannot be further minimised and the optimal model parameter values are found.

Support Vector Machines (SVM)

Classification using SVM can be illustrated by using a simple example wherein there are two linearly separable classes in d-dimensional space. Given a set of training data in d-dimensional space, the goal is to find an optimal hyperplane that runs between both classes, with all cases of a class located to one side of the separating hyperplane. Additionally, the distance between the hyperplane and the observations closest to the hyperplane (support vectors) referred to as the *margin* needs to be maximised.

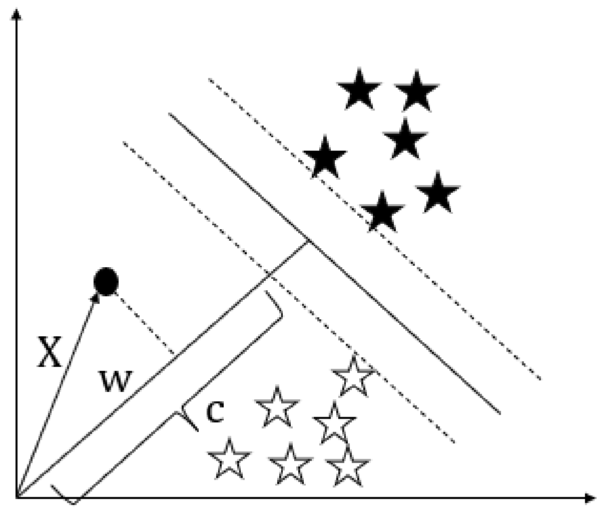


Figure 2.6: Classification using SVM

If we consider a random point X represented by a vector in Figure 2.6, and we need to determine whether the point lies on the left or right side of the plane (depicted by the black line in Figure), then

$$\vec{X} \cdot \vec{w} = c, \text{ when the point lies on the plane,}$$
$$\vec{X} \cdot \vec{w} > c \text{ for the positive samples}$$
$$\vec{X} \cdot \vec{w} < c \text{ for the negative samples}$$

where $\vec{w}$ is perpendicular to the plane and c is the distance of $\vec{w}$ from the origin to the plane. Based on this, a decision rule can be defined as follows:

$$\vec{X} \cdot \vec{w} - c \geq 0 \text{ or,}$$
$$\vec{X} \cdot \vec{w} + b \geq 0, \text{ substituting } -c \text{ as } b$$

Hence, a point is considered positive if $\vec{X} \cdot \vec{w} + b > 0$ and a negative point otherwise. The next step is to maximize the margin or the distance between the support vectors (indicated by dotted lines in Figure 2.6). The equation for the points that lie on the either sides of the dotted lines can be defined as $\vec{W} \cdot \vec{X} + b \geq +1$ for positive points and $\vec{W} \cdot \vec{X} + b \leq -1$ for negative points. The two equations can be combined as follows:

$$y_i(\vec{w} \cdot \vec{X} + b) - 1 \geq 0 \tag{2.1}$$

The support vectors of the two classes lie on the dotted lines which are parallel to the hyperplane and are defined by $\vec{w} \cdot \vec{X} + b = \pm 1$. The margin between the two dotted lines is given by $\frac{2}{\|w\|}$ and is subject to maximisation. This can also be formulated by the following constrained optimisation problem:

$$min\{\tfrac{1}{2}\|\vec{w}\|^2\}$$

subject to the inequality constraints of 2.1.

In the real world, however, the classes are almost never perfectly linearly separable, and the constraints of 2.1 cannot be satisfied. To deal with such cases using only linear separating boundaries, so-called *slack variables* $\xi_{i=1}^{n}$ are introduced that indicate the distance the classified point is from the hyperplane [42]. The constraint of 2.1 then becomes:

$$y_i(\vec{w} \cdot \vec{X} + b) > 1 - \xi_i \tag{2.2}$$

In case of outliers, the above constraint can easily be met, so a penalty term $C\sum_{i=1}^{r}\xi_i$ is added to penalise solutions for which ξ_i are very large. The constant C controls the magnitude of the penalty associated with training samples that lie on the wrong side of the decision boundary [42]. With the addition of the penalty term, the optimisation problem becomes:

$$min\left[\frac{\|\vec{w}\|^2}{2} + C\sum_{i=1}^{n}\xi_i\right]$$

subject to the constraints of 2.2. The first part of the term aims to maximize the margin,while the second part seeks to penalize the cases located on the incorrect side of the decision boundary with C controlling the relative balance of these two competing objectives.

This basic approach can also be extended to allow for non-linear decision surfaces. In this, the input data may be mapped onto a high-dimensional space through some non-linear mapping that has the effect of spreading the distribution of the data points in a way that enables the fitting of a linear hyperplane. More formally, an input data point X can be represented as $\Phi(X)$ in the high dimensional space. This is achieved through the use of *kernels* (functions that map lower dimensional inputs to a higher dimension).

While there are several approaches to extend this problem to a multi-class classification problem, the *one-versus-one* strategy is employed in this study. In this approach the multi-class classification is split into several binary classification problems. In this, a series of classifiers are applied to each pair of classes. Therefore, this method requires $\frac{C(C-1)}{2}$ classifiers to be applied to each pair of classes, where C is the number of classes. Each binary classifier may predict one class label and the input is predicted to belong to the class with the maximum number of votes.

Recurrent Neural Networks (RNN) [1]

Deep Neural Networkss (DNNs) have been used for a variety of applications and have especially revolutionised the field of natural language processing. RNNs and CNNs are two main kinds of DNN architectures that have been widely explored [43]. RNNs were particularly developed for sequence modeling tasks, e.g., predicting the next word from a sentence (a sequence of words). Therefore, its use for next activity prediction in PBPM literature is quite intuitive - a prefix can be considered as a sequence of events from which the next activity has to be predicted. Figure 2.7 depicts a simple representation of a RNN adapted from [44].

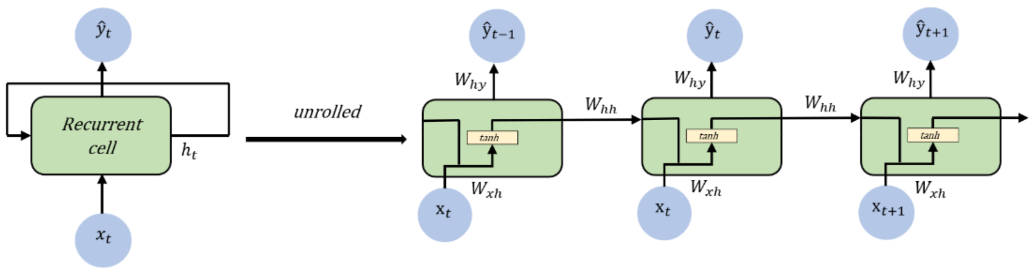


Figure 2.7: Neurons with recurrence

The way in which RNNs differentiate from regular neural networks is that the network's output predictions are not only a function of the input at a particular time step, but also the past memory of the cell state. This is depicted in Figure 2.7 through the arrow going to itself which indicates that the output of the previous timestep is fed to the next layer. In order to have a more clear representation, an 'unrolled' form of the RNN is used (Figure 2.7). At each timestep t, the hidden states and outputs are:

$$h_t = tanh(W_{hh}^T h_{t-1} + W_{xh}x_t + b_h)$$
$$\hat{y}_t = \sigma(W_{hy}^T h_t + b_y)$$

where W_{xh}^T, W_{hh}^T and W_{hy}^T are the weights or parameters of the model. For the forward propagation, each layer is propagated through one time step at a time. The difference between the output y_t and the target $\hat{y}_t$ is represented by the loss function that has to minimised across all T time steps:

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial W}$$

Learning with RNNs, has long been considered to be difficult [45]. When an input is passed to the network at time t and the loss is calculated at time T. The tying of

[1] In addition the academic sources used in this section, the information is also inspired from other sources: Colah's blog, Madu Sanjevi's post on Medium and Nir Arbel's post on Medium

weights across time steps means that the recurrent edge at a hidden node h always has the same weight. Therefore, the contribution of the input at time t to the output at time T will either explode or approach zero, exponentially fast, as $T - t$ grows large. Hence, the derivative of the loss function with respect to the input will either explode or vanish. This is known as the *vanishing* and *exploding* gradient problem. Which of the two occur depends on the weights of the recurrent edges and the activation function that is used in the hidden node (see Figure 2.8 for an overview of some popular activation functions). For instance, given a sigmoid function, one can imagine the vanishing gradient problem, while with the Rectified Linear Unit (ReLU), one can image the opposite.

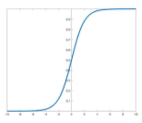
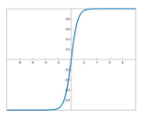
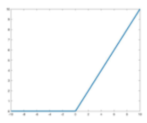
Activation function	Equation	Graph
Sigmoid	$S(x) = \frac{1}{1+e^{-x}}$	
Tanh	$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	
ReLU	$RELU(x) = \begin{cases} 0 \; if \; x<0 \\ x \; if \; x>=0 \end{cases}$	

Figure 2.8: Some popular activation functions [2]

Long Short Term Memory (LSTM)

LSTMs are a special kind of RNN that are designed to deal with the problems addressed in RNN. Each memory cell contains a node with a self-connected recurrent edge of fixed weight one, ensuring that the gradient can pass across many time steps without vanishing or exploding [45]. In standard RNNs, the repeating module has a simple structure with a single tanh activation function (Figure 2.7). LSTMs have a similar chain like structure, but the repeating module has a different structure. Figure 2.9 shows the structure of a LSTM.

[2]Image from : https://datawow.io/blogs/interns-explain-basic-neural-network

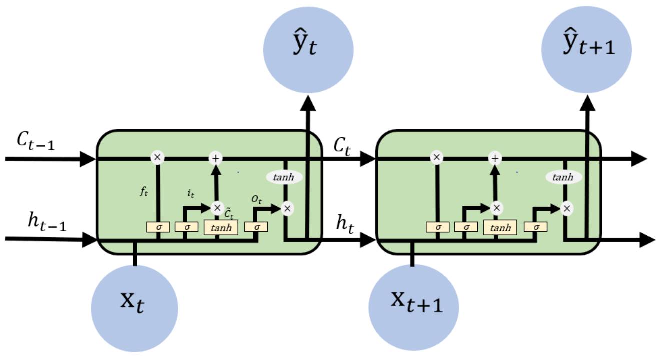


Figure 2.9: A LSTM structure

The gates in Figure 2.9 have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a distinctive feature of LSTMs. It is a gate in the sense that if its value is zero, then flow from the other node is cut off. If the value of the gate is one, all the flow is passed through. This is captured by the sigmoid funtion. Formally, the different components of the LSTM can be represented as :

$$\begin{aligned} f_t &= \sigma(W_f h_{t-1} + W_f x_t + b_f) \\ i_t &= \sigma(W_i h_{t-1} + W_i x_t + b_i) \\ \tilde{C}_t &= tanh(W_C H_{t-1} + W_C x_t + b_C) \\ C_t &= f_t C_{t-1} + i_t \tilde{C}_t \\ O_t &= \sigma(W_O h_{t-1} + W_O x_t + b_O) \\ h_t &= O_t tanh(C_t) \end{aligned}$$

The forget gate given by f_t controls what information in the cell state to forget. Similarly, the input gate i_t and the input node $\tilde{C}_t$ control what new information will be added to the cell state given the new input information. The update from the old cell state C_{t-1} to the new cell state C_t is reflected by multiplying the old by f_t, to forget the information if needed and then adding it to $i_t\tilde{C}_t$ which is the new information. The final output O_t is first run through a sigmoid layer to determine what parts of the current cell to keep and then run through the tanh activation function so that the output of each cell has the same range as the hidden units [45]. The value h_t ultimately produced by a memory cell is the value of the internal state C_t multiplied by the value of the output gate O_t.

In terms of the forward pass, the LSTM can learn when to let activation into the internal cell state. As long as the input gate takes values zero, no activation can get in. Similarly, the output gate learns when to let the value out. When both gates are closed, the activation is trapped in the memory cell, neither growing nor shrinking, nor affecting the output at intermediate time steps [45]. Because each memory cell contains a node with a self-connected recurrent edge of constant weight, the error as well as the gradient during backward pass can flow across time steps without vanishing or exploding. This edge is often called the *constant error carousel* [45]. As a result of all these mechanisms, the LSTM has shown superior ability to learn long-range dependencies as compared to simple RNNs [45].

Bidirectional Long Short Term Memory (BI-LSTM)
Another popular variation of the RNN that is used is the BI-LSTM which is illustrated in Figure 2.10. In this architecture, there are two layers of hidden nodes. Both hidden layers are connected to the input and the output. The first hidden layer has recurrent connections from the past time steps just as in the regular LSTM. In the additional second layer, the direction of recurrent connections is flipped. As a result, not only past information, but also future sequence information is passed [45]. Formally, this is represented as:

$$\begin{aligned} h_t &= \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \\ z_t &= \sigma(W_{xz} + W_{zz}z_{t+1} + b_z) \\ \hat{y}_t &= \sigma(W_{hy}h_t + W_{zy}z_t + b_y) \end{aligned}$$

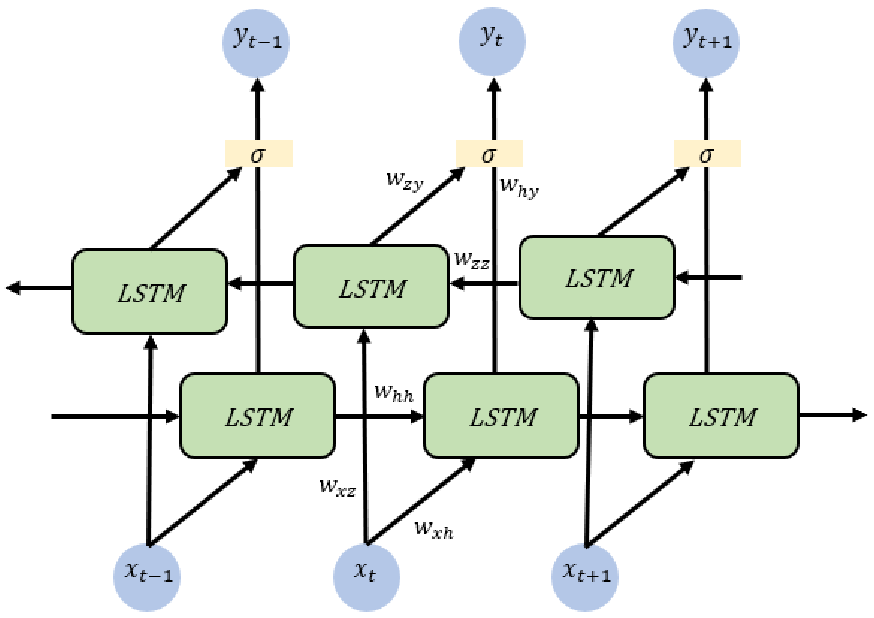
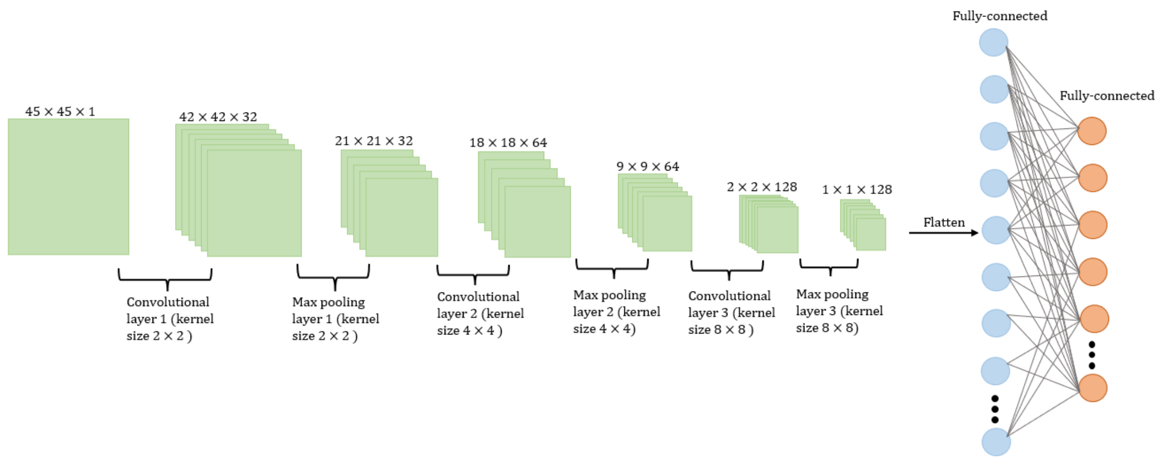


Figure 2.10: A BI-LSTM structure

The use of the LSTM variant of RNNs is quite popular in PBPM. This is often attributed to their natural ability in delivering consistently high accuracy in the natural sequence modelling of a trace. While some authors use just the "vanilla" LSTM architectures [19], others use different extensions of it. For instance, in [20], the authors use a BI-LSTM since they are forward and backward LSTMs that can exploit control-flow information from two directions of sequences. In [22], the authors use a time-aware LSTM architecture inspired by the work of [46] to account for the irregularities in the elapsed time between events. [40] combine BI-LSTMs with an attention mechanism where different attention weights are assigned for the different features of inputs. This is done to only account for the critical events and their attributes and minimise the influence of non-critical events or attributes. [47] use a cost sensitive LSTM model (CS-LSTM) to account for class imbalance in the data. In [48], the authors introduce a hierarchical attention mechanism network (HAM-Net) where they use two layers of the attention mechanism, hierarchically, on top of BI-LSTM.

Convolutional Neural Networks (CNN)

Another class of DNNs is the CNN which have shown remarkable performance in a variety of applications with its roots in image recognition. The CNN architecture comprises of three layers- *convolutional layers*, *pooling layers* and *fully-connected layers*. Figure 2.11 depicts a CNN architecture with three convolutional and pooling layers (hidden layers) and the fully-connected layers. The dimensions for each of the images are based on the experiments conducted in this study and will be discussed in Chapter 3. In this section, just the architecture and the working of the CNN is introduced. The detailed explanation of how traces are processed to form images for this study will be elaborated in Chapter 3.



45 × 45 × 1
42 × 42 × 32
21 × 21 × 32
18 × 18 × 64
9 × 9 × 64
2 × 2 × 128
1 × 1 × 128
Convolutional layer 1 (kernel size 2 × 2)
Max pooling layer 1 (kernel size 2 × 2)
Convolutional layer 2 (kernel size 4 × 4)
Max pooling layer 2 (kernel size 4 × 4)
Convolutional layer 3 (kernel size 8 × 8)
Max pooling layer 3 (kernel size 8 × 8)
Flatten
Fully-connected
Fully-connected

Figure 2.11: A CNN architecture

The first layer in the CNN is the convolutional layer and the parameters in this layer are centered around the use of learnable *kernels*. These kernels are small in spatial dimensionality, but spread along the entirety of the depth of the input [49]. This kernels are essentially filters and depending upon the type of the kernel, different features from the image can be extracted [3]. The type of kernel that will be selected is determined during learning during the training process and typically multiple filters are used within each convolutional layer, for e.g., 32, 64, 128 etc. (see Figure 2.11). As the filter glides across the spatial dimensionality of the input, the scalar product is calculated for each value in the kernel to produce a feature map and then run through an activation function to produce an activation map [49]. This is visualized in Figure 2.12[4].

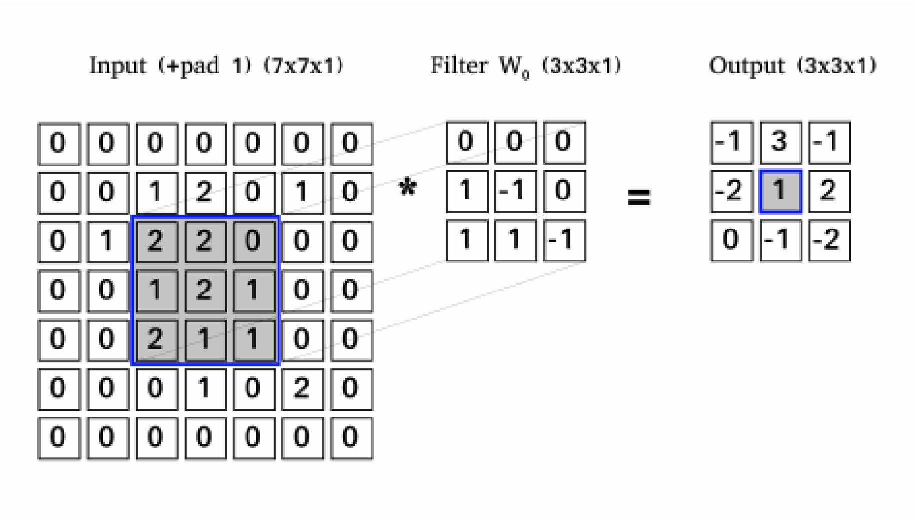


Figure 2.12: A pictorial representation of a convolutional operation with stride 2

Just like other neural networks, CNN has parameters that can be optimized through hyperparameter tuning. These parameters are the *depth* (number of filters), the *stride* (the amount of movement of each filter across the image) and *zero padding* (padding the border of the input with zeros to not affect the dimensionality of the output.)

The second layer is the pooling layer whose main aim is to gradually reduce the dimensionality of the representation. The pooling layer is applied to each activation map produced by the convolutional layer. The most common form of pooling is called *max-pooling*, wherein the kernel glides across each activation map performing the 'MAX' operation analogous to the element-wise multiplication operation that is performed in the convolutional layer. This scales the activation map down to a portion of the original size [49]

[3]One can have a look at the different kind of filters and the resulting convolved images here
[4]https://www.cosmos.esa.int/web/machine-learning-group/convolutional-neural-networks-introduction

The last layer is the fully-connected layer which is obtained by flattening the last pooling layer (see Figure 2.11). The connections here are very similar to the ones in traditional neural networks.

The use of CNNs for PBPM has been gaining momentum recently . [21] proposed the application of the CNN architecture to image-like representations. For every trace in the event log, for every prefix trace in this trace, a 2D image is generated. In this approach, the imagery rows represent the consecutive indices of the events in the prefix trace and the imagery columns represent the distinct activities of the activity domain. Every cell in this imagery is a 2-dimensional vector including the sequence of activities (encoded using frequency encoding) and the corresponding time attributes, respectively. [50] differs from [21] in that they do not adopt any complex preliminary data engineering step. Instead, they directly extract a hierarchy of representation for the process traces by computing dilated convolutions directly on event sequences.

2.8.3.5 Evaluation Measures

The most commonly used evaluation measure in the reviewed literature is accuracy, where $accuracy = \frac{TP+TN}{TP+FP+TN+FN}$, where TP, TN, FP and FN refer to true positive, true negative, false positive and false negative, respectively. Accuracy represents the ratio of correctly classified process instances to the total number of process instances. Some papers use precision and recall as evaluation measures. Next activity prediction is a multi-class classification problem, and so precision and recall can be defined for each of the classes. So, $precision = \frac{TP}{TP+FP}$ is the number of correct predictions of a class out of all the predictions for that particular class. On the other hand, $recall = \frac{TP}{TP+TN}$ is the number of correct predictions of a class out of the total number of instances that actually carry that particular class value. Some studies go one step further and calculate the f1-score of the methods where f1-score is $\frac{precision \times recall}{precision+recall}$.

Area under the curve (AUC) is another common evaluation measure adopted by different studies. In order to calculate this, a receiver operator characteristic (ROC) curve needs to be created which plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold values. Each prediction represents one point in the ROC space. The best possible prediction method would yield a point in the upper left corner resulting in a perfect classification. On the other hand, a random guess would be a point along a diagonal line. Analogous to calculations of precision and recall, this measure is also typically used for binary classification problems but can be extended to multi-class classification problems as well. In this case, the class to be predicted is considered the positive class while the rest are considered negative classes.

Another typical evaluation measure that is used in studies is the mean absolute error (MAE) which is the arithmetic mean of the prediction errors which is given by $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$, where y_i is the actual value while $\hat{y}_i$ is the predicted value. While these measures are widely used, some uncommon measures are also adopted in some studies. For instance, [51] use the *kappa coefficient* which measures the degree of agreement between the true values and the predicted values. This is given by, $\kappa = \frac{p_o - p_e}{1 - P_e}$, where p_o is the observed agreement (or observed prediction), and p_e is the expected agreement (or expected prediction) . In [52], the Brier score is used as an evaluation measure which is the mean square difference between the true classes and the predicted probability. It is given by $BS = \frac{1}{2n}\sum_{i=1}^{n}\sum_{K}^{k=1}(C_{ik} - p_{ik})^2$, where n is the number of observations, K is the number of classes, $C_{ik} = \{0, 1\}$ the indicator of class k for observation i, and p_{ik} is the predicted probability of the observation i to belong to class k [53]. [54], Matthews Correlation Coefficient (MCC) is proposed as an accuracy metric. It is given by $MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$.

A very popular evaluation measure unique to predictive business process monitoring is measuring the earliness of predictions. To do this, a classifier is applied to a subset of prefix traces with each length until the prediction of a an ongoing case with different lengths reaches an acceptable level of accuracy [5] [40]

Table 2.1: *Classification of papers based on the steps in figures 2.3 & 2.4 (RQ2)*

Study	Prefix Extraction	Trace Bucketing	Sequence Encoding		Classifier	Evaluation
			Control flow	Data Payload		
[20]	Prefix length (3-8)	Prefix length	Boolean	None	Bi-LSTM	Accuracy & F-1 Score
[21]	All prefixes	None	Frequency	Index	CNN	Precision & Recall
[19]	All prefixes	None	Index	Agg	LSTM	MAE
[22]	All prefixes	None	Index	Agg	LSTM	MAE
[51]	All prefixes	Clustering	Agg	Agg	RF	Kappa coefficient
[55]	All prefixes	Prefix length	Index	Index	RF & LR	Precision, Recall & F-Score
[56]	Prefix length	Prefix length	Boolean	None	GAN	MAE
[57]	All prefixes	Clustering	Boolean	Index latest payload	Decision tree	AUC
[58]	None	None	Index, Categorical	Index, Categorical	ANN	MAE
[40]	All prefixes	Prefix length	Boolean	None	Att-Bi-LSTM	AUC
[30]	All prefixes	None	Categorical	Categorical	LSTM	F-Score

Table 2.1 (continued)

Study	Prefix Extraction	Trace Bucketing	Sequence Encoding		Classifier	Evaluation
			Control flow	Data Payload		
[39]	All prefixes	None & Prefix length	Index, Frequency	Index latest payload	XGBoost	AUC
[48]	All prefixes	None	Boolean	None	Ham-Net	Accuracy
[47]	All prefixes	None	Boolean	None	CS-LSTM	Accuracy
[50]	All prefixes	None	Index	Agg	CNN	Accuracy
[59]	All prefixes	None	Frequency, Simple index & Index	Frequency, Simple index & Index	RF	Accuracy & AUC
[54]	None	Defined checkpoints	Boolean	None	RNN	Matthews Corr-elation Coefficient
[5]	Prefix length (2-20)	Prefix length	Index	Index	RF	AUC Score
[27]	Prefix lengths of gaps (3,5 & 10)	Clustering	Frequency	Latest payload	DT & RF	Accuracy
[52]	All prefixes	None	Categorical	Index	CNN	Brier Score & Accuracy

Chapter 3

Classifying Anomalous Traces

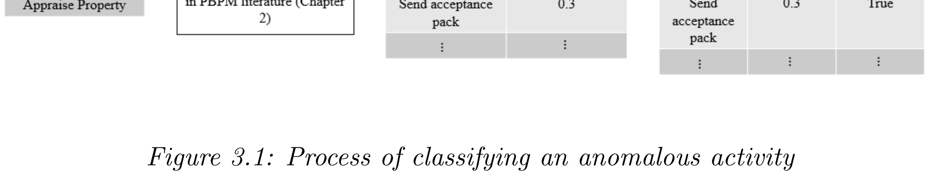
In this chapter, the procedural method (the artefact) that is developed for the experiments is introduced (Section 3.4). Its use is then demonstrated step-by-step for the chosen use case of anomaly classification.

3.1 Machine Learning Objective

In this work, the task of classifying anomalous activities is formulated as a supervised multi-class classification problem (as introduced in section 1.2.3.1). For every activity in an event log, the aim of the trained classifier is to emit a probability for each of the unique activities observed in the event log to happen next. If the actual activity or the ground truth has a very low probability of occurring according to the prediction, then it is classified as an anomaly. This is illustrated in Figure 3.1

3.2 Choice of Classification Algorithms

This study makes use of classification algorithms identified in literature to determine anomalous activity. Therefore, before anything, it is important to choose the classifiers that will be used for the experiments since most of the data preparation is heavily reliant on this. Based on the systematic review conducted (see chapter 2), RF, XGBOOST,



Activity
Appraise Property

Next Activity Prediction using classifiers identified in PBPM literature (Chapter 2)

Next Activity	Probability
Check credit history	0.8
Send acceptance pack	0.3
⋮	⋮

Next Activity	Probability	Anomalous
Check credit history	0.8	False
Send acceptance pack	0.3	True
⋮	⋮	⋮

Figure 3.1: Process of classifying an anomalous activity

Python Libraries/ Frameworks	Description
`NumPy`	Popular python library for large multi-dimensional array and matrix processing
`Pandas`	Popular python library that is used for data analysis. In this study, it is primarily used for building and manipulating dataframes needed for creating classical machine learning models
`Scikit-learn`	Popular machine learning library for classical machine learning algorithms. In this study, it is used to implement RF, XGBOOST, MLR and SVM and the tuning of their parameters. It is also used for data analysis and for visualising the evaluation metrics
`TensorFlow`	It is a popular open source library especially for developing deep learning models. It is specially used for computations involving tensors
`Keras`	It is a high-level neural networks API that runs on top of TensorFlow. It is used in this study to build the deep learning models as well as to tune the models through `KerasTuner`

Table 3.1: Python libraries and frameworks used in this study

SVM, MLR, LSTM, BI-LSTM and CNN are selected for the experiments because of their wide adoption in PBPM literature for a variety of use cases.

3.3 Technical Details

All the experiments have been carried out in python using Google Colab with GPU enabled. The different frameworks and libraries used for the experiments are summarised in Figure 3.1

3.4 Procedural Method

In this section, the procedural method is described. Every large machine learning project follows a generic pipeline that includes *data acquisition*, *data preparation*, *training* and evaluation [60]. This project is not any different and adheres to a similar structure, but differs in the way each step is implemented. At a more granular level, the experimental setup closely resembles the general workflow adopted in PBPM literature (see section 2.8.3). The setup is visualised in Figure 3.2 .

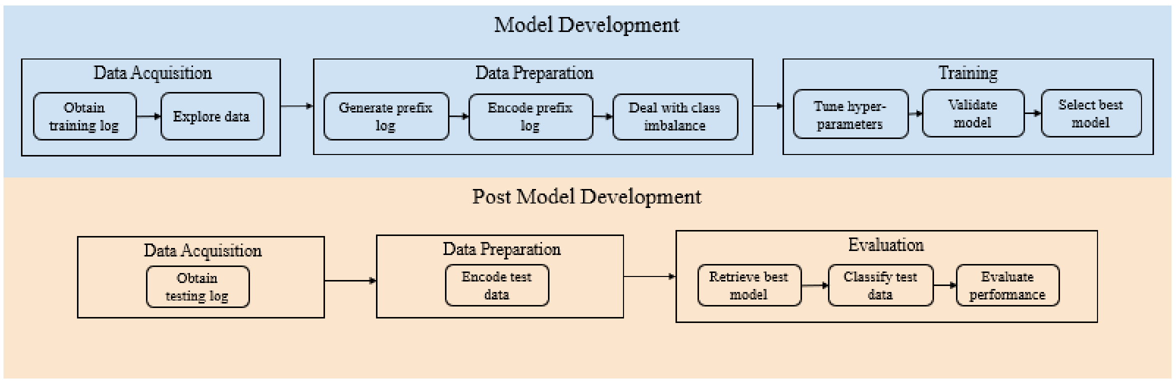


Figure 3.2: Experimental approach for this study

3.4.1 Model Development

In this section all the steps leading onto the development of the classification model from acquiring the data to training the model is elaborated.

3.4.1.1 Data Acquisition

Acquiring data is the first step and foundation for any machine learning project. In the next two subsections, the event log used for training is described along with some visualisation to provided the reader with some insight into the underlying data.

3.4.1.1.1 Obtain Training Log

The event log used for this research pertains to a loan application process and has been taken from [61]. This particular event log has been chosen since it represents a financial process and can realistically be part of an audit.

3.4.1.1.2 Explore Data

The best way to visualise an event log is through a process map that captures the underlying process that the event log represents. Figure 3.3 depicts the loan application process which is generated using the process mining visualisation tool Disco. Additionally, a basic description of the training log is provided in Table 3.2. Unlike traditional machine learning projects, the data does not consist of external features or variables. The features are the activities themselves, wherein previous activities are used as input features to generate a prediction for the next activity. Of course, external information can also be used as input, however, the data used here does not contain any external information.

For any classification problem, it is necessary to visualise the distribution of classes in

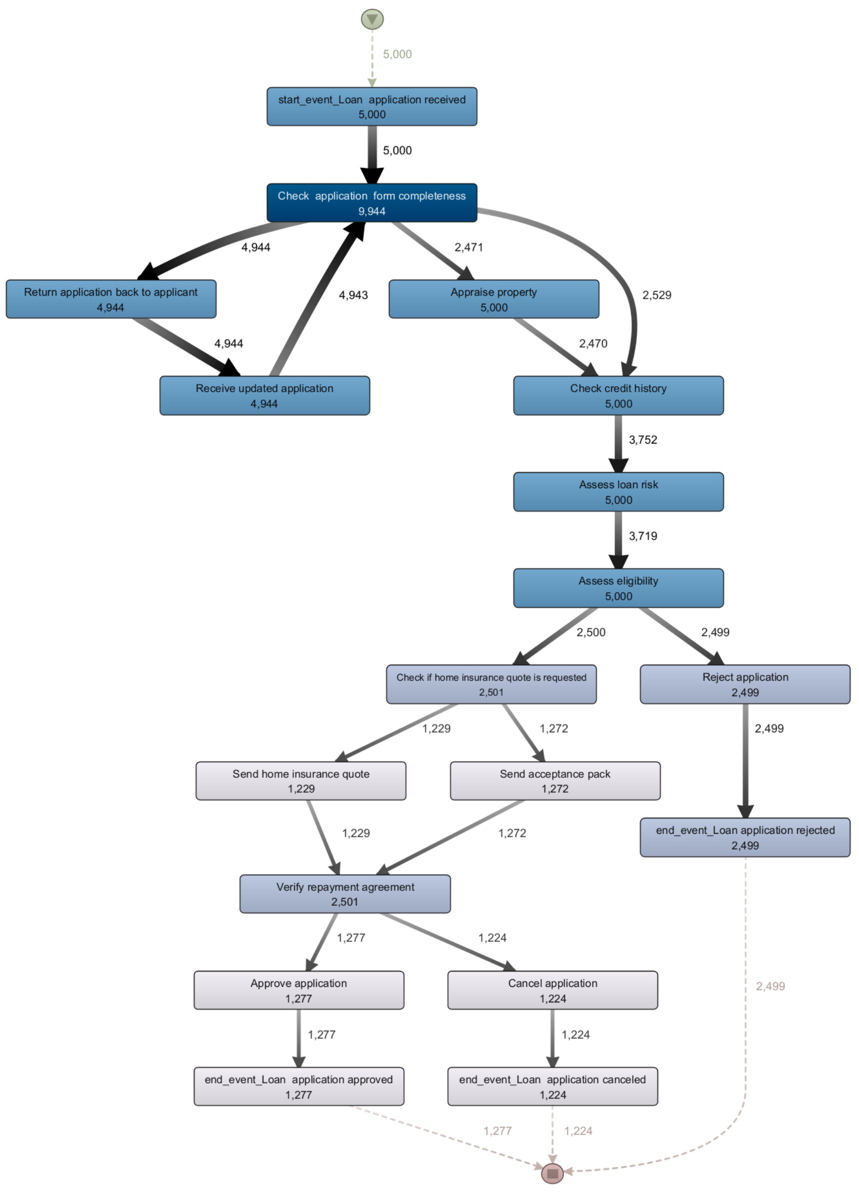


Figure 3.3: Loan application process visualized using Disco

Description of training log	
# Unique Cases	5000
# Events	62324
# Unique Activities	18

Table 3.2: Description of training log

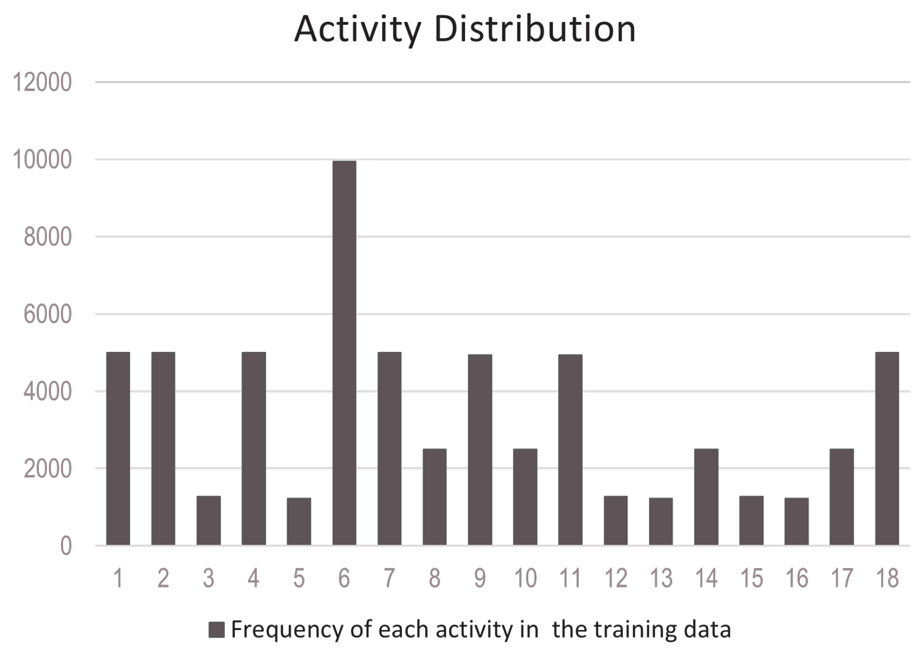


Figure 3.4: Class distribution

the data. Most machine learning algorithms assume that data is equally distributed. So in case of a class imbalance, the machine learning classifier tends to be more biased towards the majority class, causing bad classification of the minority class. The problem addressed in this study is a multi-class classification problem, where the classes are the unique activities observed in the event log. The distribution of the classes can be seen in Figure 3.4. From the figure, it is evident that all the classes are not equally distributed. The mechanisms for dealing with class imbalance will discussed in Section 3.4.1.2.3

3.4.1.2 Data Preparation

In the related subsections, all the steps undertaken for preparing the data for training are described in detail.

3.4.1.2.1 Generate prefix log

The first step in the data preparation phase is converting the training event log to a prefix log. The event log is converted to a prefix log because during testing, the classifier is supposed to generate an prediction based on an incomplete trace (see Section 2.8.3.1). An example of how a case in an event log is converted to a prefix is illustrated in Table 3.3.
After creating a prefix log, many studies use clustering approaches to bucket similar

Case ID	Activity
1	Activity 1
1	Activity 2
1	Activity 3

(a) A single case with three events

⇒

Case ID	Prefix No	Activity
1	1	Activity 1
1	2	Activity 1
1	2	Activity 2
1	3	Activity 1
1	3	Activity 2
1	3	Activity 3

(b) Prefixes of the case in 3.3a

Table 3.3: An example of converting a single case in an event log to prefixes

prefixes for training (see Table 2.1). However, this is not done for two reasons that were observed during experimentation:

- A bucket may have too few traces to train a meaningful classifier. This is usually tackled by creating a minimum bucket size threshold [62]. If the number of trace prefixes in a bucket is less than the threshold, the classifier is not trained on that bucket, rather, the prefix is mapped to the label that is predominant in that bucket. However, in this study, upon experimentation, it is observed that the classifier does not train rather 'memorises' the training data, even in buckets with sufficient traces. This is probably due to the fact that the traces in a bucket are quite similar to each other.
- Because a bucket may have similar traces, there is a very high possibility of contradictions (cases where matching prefixes have different labels). The performance of classifiers can be negatively impacted by the presence of such contradictions. The PhD work of [63] addresses this in detail.

Therefore, in this study, a single classifier is trained on all the prefixes of the training data.

3.4.1.2.2 Encode Prefix Log

The next step in the data preparation phase is encoding the prefix log. Machine learning algorithms do not support textual attributes, and therefore, such attributes need to be converted to a numerical format before they can be fed to a machine learning model. Different methods exist for this in literature (see Section 2.8.3.3). However, the type of encoding to be used depends on the chosen algorithm. For example, tensor encoding is popular for deep learning models because it supports vector and matrix operations that can be efficiently computed given modern GPU architectures. But, such sparse data representations deter the performance of, for instance, a RF model. Taking this into

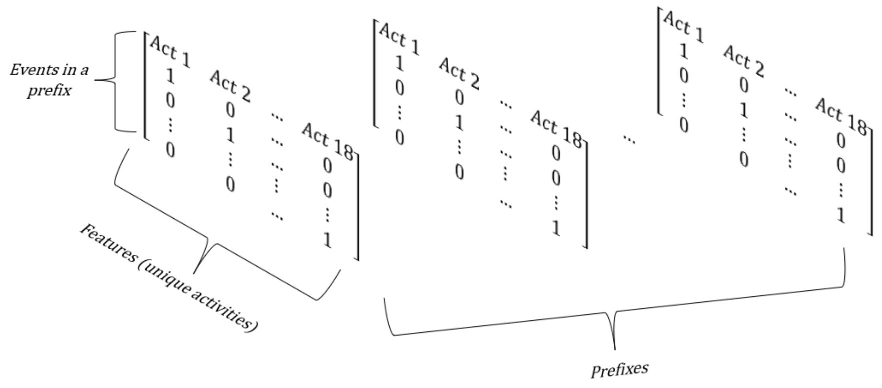


Figure 3.5: Tensor encoding of a prefix log

consideration, depending on the type of classification model, the encoding technique is adopted accordingly. For all deep learning models (LSTM, BI-LSTM and CNN) used in this study, tensor encoding is used and for all other models (RF, XGBOOST, SVM, and MLR), frequency encoding is used.

Tensor Encoding
The features of the prefix log are essentially all the unique activities that are observed in the event log. Since, past activities are used as input to predict a next activity, it can be beneficial for the sequence of all the activities within a case to be maintained. For deep learning models (RNNs and CNNs), this can be achieved with the help of *tensors* which can be simplistically thought of as generalised matrices (in this case, generalised to three dimensions). Figure 3.5 illustrates the tensor encoding of a prefix log.

Each prefix in the prefix log can be thought of as a two dimensional $m \times n$ matrix where m is the number of features or unique activities that are observed in the training log and n is the length of the largest trace observed in the training log (45 in this study). For prefixes that have lengths less than the largest trace, remaining entries are padded with zeroes. Thus, each prefix matrix has the same length. The third dimension, then, just depicts the collection of prefixes in the prefix log. So each input to a DNN model is a prefix matrix. This is illustrated in Figure 3.5

For CNNs in particular, a special kind of preprocessing is required. CNNs are usually used for image classification where the inputs to the model are images. So, here, the prefixes need to be converted to images. An image can be considered as a $w \times h \times 3$ matrix, where w and h are the width and height of the image and the third dimension depicts the three colour channels red, green and blue. Each value in the $w \times h$ grid

Case ID	Activity
1	Activity 1
1	Activity 2
1	Activity 3

(a) A single case with three events

$\Rightarrow$

Case ID	Prefix No	Act	Act	Act
1	1	1	0	0
1	2	1	1	0
1	3	1	1	1

(b) Encoding 3.4a using frequency encoding

Table 3.4: An example of encoding a singe case using frequency encoding

depicts the pixel intensity. This can be generalised to this study as follows: Each prefix is converted to a $(m \times m \times 1)$ where m is the largest trace observed in the event log, and because this particular use case doesn't require different channels, the third dimension is set to 1. The encoding here is similar to that shown in Figure 3.5. The entire CNN architecture for the base CNN architecture can be visualised in Figure 2.11.

Frequency Encoding

For the models used in these experiments that are not DNNs i.e., RF, XGBOOST, MLR and SVM, tensor encodings are not supported by the python libraries that implement them. This can be counteracted by representing each prefix as a 1×45 vector (45 is the length of the largest prefix). However, these sparse representations can lead to poor prediction accuracies. For this reason, frequency encoding (see Section 2.8.3.3) is used. Table 3.4 illustrates frequency encoding for a single sample case.

3.4.1.2.3 Dealing with Class Imbalance

From Figure 3.4, it can be seen that all the activities are not evenly distributed. A consequence of this while testing is that a class occurring more frequently during training is predicted because the model is not able to accurately learn the characteristics of the training examples of a less frequent class. There are two main approaches to deal with class imbalances (1) resampling data; (2) creating penalised models. Resampling is carried out by either *oversampling* by adding copies of instances from the under-represented class or *undersampling* by deleting instances from the over-represented class. The problem with this method is that it changes the original underlying information. For this reason, in this study, the technique of penalised classification is used. In this method, an additional cost on the model is imposed for making classification mistakes on the minority class during training. For the classification models implemented in `scikit-learn`, this method is implemented through the `class_weight` parameter by setting it to `balanced`. The balanced class weighting was introduced by [64] and it works by assigning a smaller error to smaller weight values and larger error values to larger weights. The class weights are determined using the following formula:

$$c_w = \frac{n}{k \times n_c}$$

where the class weight for a particular class is c_W, n is the total number of training examples, k is the total number of classes and n_c is the number of training examples of the class whose weight needs to be determined.

For the deep learning models, the same formula is implemented using a customised loss function that is used during training

3.4.1.3 Training

3.4.1.3.1 Tune Hyperparameters

When building a machine learning model, one is presented with the dilemma of how to define a model architecture to fit a specific use case. A classification algorithm on its own is generic and can not always be used as-is for all use cases. The process of hyperparameter tuning can be thought of as instantiating a generic model for a specific use case. We often do not know beforehand what model parameters (e.g., trees in RFs or number of neurons in LSTMs) will produce optimal results for our data. We would like our model to learn and explore different possibilities. When training, our model can plug in different values and the values that optimises the loss function best is picked. There are several approaches for hyperparameter tuning. For this study, the two most common methods, namely grid search and random search are considered.

Grid Search

Grid search is the most intuitive and basic way to carry out hyperparameter tuning. In this approach, during training, a model is built for each possible combination of all the hyperparameter values provided. Each model is evaluated and the configuration that produces the best results is chosen. Clearly, the major drawback with this approach is its exhaustiveness which is extremely inefficient. For most data sets, only a few hyperparameters play a major role in optimising the model score. Grid search has the tendency of exploring even unimportant parameters.

Random Search

Random search, as the name suggests, randomly samples from a set of hyperparameter values. The strategy in random search is an exploratory one as opposed to grid search's greedy one. However, this approach is far from perfect. There is a possibility that an optimal configuration may never have the chance to get explored.

Combining Random Search & Grid Search

In this study, an attempt is made to utilise the strengths of both methods by combining the exploratory and greedy nature of random search and grid search respectively. First random search is carried out to narrow the set of hyperparameters through random sampling. Then, grid search is performed on this narrowed down set of parameters.The hyperparameter search space considered for the experiments is summarised in Table 3.5.

Classification Algorithm	Configurable parameters for tuning	Set of values considered during experimentation
RF	`max_depth`: the maximum depth of the tree. A greater value can lead to overfitting	[10, 20, 30, 40, 50, 60,70. 80, 90, 100, None]
	`min_samples_split`: the minimum number of observations in a given node required to split it	[1, 2, 5, 10, 15, 25, 50, 75, 100, 200]
	`max_features`: maximum features provided to each tree	[sqrt(n_features), log2(n_features)]
	`bootstrap`: method for sampling (with or without sampling)	[True, False]
	`min_samples_leaf`: minimum number of samples that should be present in the leaf node after splitting a node	[1, 2, 5, 10, 15, 25, 50, 75, 100, 200]
	`n_estimators`: number of trees in our ensemble	[100, 200, 300, 400, 500]
XGBOOST	`booster`: the type of booster to use	gbtree
	`eta`: the learning rate that determines the step size at each iteration as the model optimizes toward the objective	[0.01, 0.03, 0.05, 0.1, 0.15, 0.2, 0.3]
	`max_depth`: the maximum depth of the tree. A greater value can lead to overfitting	[3, 5, 7, 10]
	`n_estimators`: number of tree in our ensemble	[100, 200, 300, 400, 500]
	`colsample_bytree`: represents the fraction of columns to be randomly sampled for each tree	[0.2, 0.4, 0,6, 0.8, 1]
	`subsample`: represents the fraction of observations to be samples for each tree	[0.2, 0.4, 0,6, 0.8, 1]
	`alpha`: L1 regularization on the weights	[0, 0.5, 1, 1.5, 2, 3, 4, 4.5]
	`lambda`: L2 regularization on the weights	[0, 0.5, 1, 1.5, 2, 3, 4, 4.5]

Table 3.5 continued from previous page

	`gamma`: the minimum loss reduction required to make a split. It is another regularization parameter	[0, 0.1, 0.3, 0.4, 0.5, 0.6]
SVM	`c`: decides the penalty for each misclassified point. A smaller value of c allows for a bigger margin for points to be misclassified	[0.1, 1, 10, 100]
	`kernel`: a method to spread the data points across N\-dimenional space	[linear, polynomial, RBF, sigmoid]
	`gamma`: influences how far the data points will be from the hyperplane	[1, 0.1, 0.01, 0.001]
MLR	`solver`: algorithm to use in the optimization problem	[newton-cg, lbfgs, sag, saga]
	`penalty` : the type of regularization to use	[None, L1, L2, elasticnet (both L1&L2)]
	`c`: controls penalty strength	[100, 10, 1, 0.1, 0.01
LSTM	`layers`: number of hidden layers	[1, 2, 3]
	`units`: number of neurons in a layer	[32 - 512 with step size of 32]
	`learning_rate`: determines how quickly parameter in the network are updated	[0.01, 0.001, 0.0001]
	`activation`: a method to introduce non-linearity to models	[tanh, sigmoid, relu]
	`dropout`: a layer to prevent overfitting by bypassing randomly selected neurons.	[0-0.5 with step size of 0.05]
	`kernel_initializer`: method for random initialization of weights	[uniform, lecun_uniform, glorot_normal, glorot_uniform, he_normal, he_uniform]
BILSTM	`layers`: number of hidden layers	[1, 1,2]
	`units`: number of neurons in a layer	[32 - 512 with step size of 32]

Table 3.5 continued from previous page

	learning_rate: determines how quickly parameter in the network are updated	[0.01, 0.001, 0.0001]
	activation: a method to introduce non-linearity to models	[tanh, sigmoid, relu]
	dropout: a layer to prevent overfitting by bypassing randomly selected neurons.	[0-0.5 with step size of 0.05]
	kernel_initializer: method for random initialization of weights	[uniform, lecun_uniform, glorot_normal, glorot_uniform, he_normal, he_uniform]
CNN	layers: number of convolutional layers	[2, 3]
	filters:number of filters in each convolutional layer	[32, 64, 128]
	learning_rate: determines how quickly parameter in the network are updated	[0.01, 0.001, 0.0001]
	activation: a method to introduce non-linearity to models	[tanh, sigmoid, relu]
	kernel_initializer: method for random initialization of weights	[uniform, lecun_uniform, glorot_normal, glorot_uniform, he_normal, he_uniform]

Table 3.5: Hyperparameter search space used for the experiments

3.4.1.3.2 Model Validation

A well-known fundamental problem in machine learning is overfitting, a process where a model tries to fit the training data perfectly, memorising even the tiny fluctuations and noise in the data. As a result, it is not able to generalise on new, unseen data. It is impossible to know when exactly a model is overfitting without testing it out on data that is not used for training. Cross-validation is commonly used for this purpose. It is a technique for evaluating machine learning models by training several models on subsets of the available input data and evaluating them on the complementary subset of the data.

In this study, during the tuning process, cross-validation is already carried out through the `GridSearchCV` and `RandomSearchCV` implementations from the `scikitlearn` library. In these implementations, *stratified k-fold* is used which is a variation of k-fold where each subset contains approximately the same percentage of samples of each target class as the complete set. Empirically, the value of k used is found to be between 5 and 10. In this study, 10 fold validation i.e., $k = 10$ is used.

3.4.1.3.3 Select best model

After the training process is completed, for each classification algorithm, we would like to save the model which performed the best during training so that we that we can later retrieve it and apply it to generate predictions on the test data. For deep learning models based on neural networks that are implemented through the `tensorflow` library, this is possible through the `ModelCheckpoint` function that allows one to continually save the model both during and at the end of training. The model is stored in a `HDF5` format, an open source file format that supports large, complex, heterogeneous data. For models that are implemented through the `scikit-learn` library, `pickle` is used. The `pickle` module in python implements the algorithm for serialising and de-serialising data structures in python.

3.4.2 Post Model Development

In the following subsections, the steps to evaluate the models developed in the previous phase are described.

3.4.2.1 Data Acquisition

In the following subsection, the process of acquiring the test data to evaluate the models developed in the previous phase is described.

Description of testing log	
# Unique Cases	5000
# Events	58785
# Unique Activities	18
# Anomalous events	1140

Table 3.6: Description of testing log

3.4.2.1.1 Obtain Testing Log

The event log used for testing has also taken from [61], and is similar to the training data, but infrequent events have been injected randomly to convert them to *anomalous logs* for evaluation. A description of the testing log can be visualised in Table 3.6. For testing, the testing log is converted to a prefix log and each prefix or incomplete trace is sent to the classifier one at a time to generate a prediction.

3.4.2.2 Data Preparation

Just like the training step in the model development phase, the test data also needs to be transformed before it is fed into a classifier. This is explained in the next subsection.

3.4.2.2.1 Encode Test Data

The process for encoding the test data is similar to one adopted to prepare the data for training. Each incoming prefix or incomplete trace generated from the test prefix log is encoded and then sent to the classifier.

3.4.2.3 Evaluation

3.4.2.3.1 Retrieve Best Model

In order to evaluate the performance of the classifiers on the test data, the best model for each classifier is retrieved. This is the model that performed the best during the training process.

3.4.2.3.2 Classify Test Data

For a given activity in the test data, each classifier produces a probability for every activity that follows next and the activity with the maximum probability is predicted as the next activity.

3.4.2.3.2.1 Threshold for determining anomalies

There is no fixed threshold value that one can apply to determine anomalies. This value is determined upon experimentation. It could also be determined from domain knowledge - knowing how frequently or infrequently an activity occurs in reality. In this study, the threshold is derived experimentally. For each activity in the test set, a set of top three most probable next activities are determined. If the ground truth or the observed next activity are not in this set, it is determined an anomaly.

3.4.2.3.3 Evaluate Performance

In this section, the steps for measuring the performance of the classifiers is described. To understand the effects of hyperparameter tuning and class weighting, four different configurations of each classifier are evaluated: (1) base model; (2) base model with hyperparameter tuning only; (3) base model with class weighting only; (4) base model with both hyperparameter tuning and class weighting. For RF, XGBOOST, MLR and SVM, the base model is the model which is not tuned or class-weighted and uses the default hyperparameter configuration from python's `scikit-learn` library. For LSTM, BI-LSTM and CNN, the base model refers to the model configurations used by [19], [20] and [21], respectively. The tuned models refer to best models for each classifier with the optimal hyperparameter configuration that were determined during training. The class-weighted models are those that account for class imbalance (see section 3.4.1.2.3). Table 3.7 shows the default parameters as well as the optimal parameters obtained after tuning.

Choice of Evaluation Metrics

Even though accuracy and f1-score are mostly adopted as evaluation measures in PBPM literature (see Table 2.1), the confusion matrix is chosen for evaluation as it gives a better depiction of the performance of the classifier, especially considering that the proportion of anomalies in the data with respect to the non-anomalous events is quite low. Based on the confusion matrix, the precision, recall and f1-score for each class is also computed.

Classification Algorithm	Parameters	Default Parameters	Optimal Parameter for tuning without class weights	Optimal Parameters for tuning with class weights
RF	`max_depth`:	None	60	60
	`min_samples_split`	2	2	2
	`max_features`	sqrt(n_features)	sqrt(n_features)	sqrt(n_features)
	`bootstrap`	TRUE	TRUE	TRUE
	`min_samples_leaf`	1	5	5
	`n_estimators`	100	500	500
XGBOOST	`eta`	0.3	0.3	0.3
	`max_depth`	6	4	6
	`n_estimators`	100	100	200
	`colsample_bytree`	1	0.6	0.5
	`subsample`	1	0.8	0.5
	`alpha`	0	0	0
	`lambda`	1	1	1
	`gamma`	0	1	0
SVM	`c`	1	0.001	1
	`kernel` function	Linear	Linear	Linear
	`penalty`	L2	L2	L2
MLR	`solver`	lbfgs	newton-cg	newton-cg
	`penalty`	L2	L2	L2
		1	0.1	0.1
LSTM	layers	2	2	2
	`units layer 1`	100	160	196
	`units layer 2`	100	160	196

Table 3.7 continued from previous page

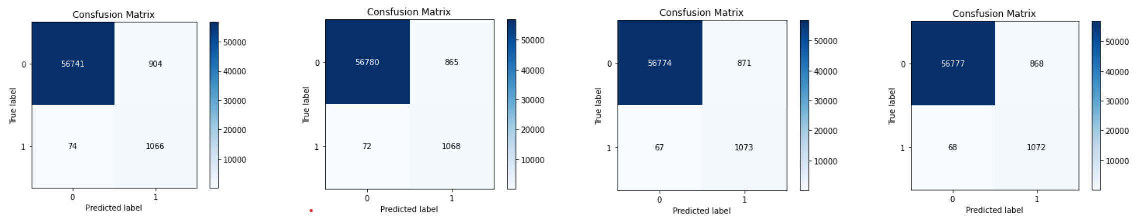
	`learning_rate`	0.002	0.0001	0.0001
	`activation`	tanh	tanh	tanh
	`dropout`	0.2	0.05	0.05
	`kernel_initializer`	glorot_uniform	he_normal	he_uniform
BI-LSTM	layers	1	2	2
	`units layer 1`	100	288	237
	`learning_rate`	0.002	0.002	0.002
	`activation`	tanh	relu	relu
	`dropout`	0.2	0.2	0.2
	`kernel_initializer`	glorot_uniform	he_normal	he_normal
CNN	`layers`	3	2	2
	`filters layer 1`	32	48	80
	`filters layer 2`	64	48	48
	`filters layer 3`	128	-	-
	`kernel size layer 1`	2	2	2
	`kernel size layer 2`	4	8	8
	`kernel size layer 3`	8	-	-
	`learning_rate`	0.002	0.01	0.001
	`activation`	relu	relu	relu
	`kernel_initializer`	glorot_uniform	he_normal	he_uniform

Table 3.7: Optimal hyperparmeters

3.5 Results

In this section, only an overview of the results is provided. The results are discussed in Chapter 4

3.5.1 RF



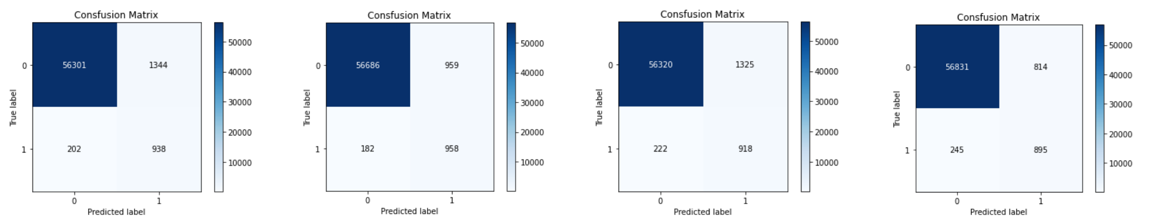
(a) Base model *(b) With hyperparameter tuning* *(c) With class weighting* *(d) With hyperparameter tuning & class weighting*

Figure 3.6: Confusion Matrices for RF

Base Model			
	Precision	Recall	f1-score
Not Anomalous	0.98	1.00	1.00
Anomalous	0.94	0.54	0.69
Base Model with Hyperparameter Tuning			
	Precision	Recall	f1-score
Not Anomalous	0.98	1.00	1.00
Anomalous	0.94	0.55	0.69
Base Model with Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	0.98	1.00	1.00
Anomalous	0.94	0.55	0.69
Base Model with Hyperparameter Tuning & Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	0.98	1.00	1.00
Anomlaous	0.94	0.55	0.69

Table 3.8: Precision, recall & f1-score for each class for RF

3.5.2 XGBOOST



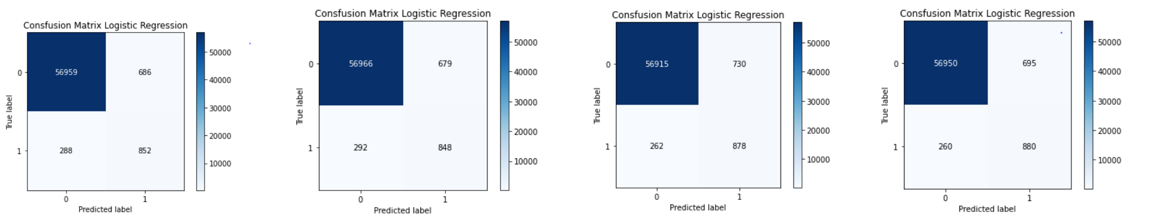
(a) *Base model* (b) *With hyperparameter tuning* (c) *With class weighting* (d) *With hyperparameter tuning & class weighting*

Figure 3.7: Confusion Matrices for XGBOOST

	Precision	Recall	f1-score
Base Model			
Not Anomalous	0.98	1.00	1.00
Anomalous	0.82	0.41	0.55
Base Model with Hyperparameter Tuning			
Not Anomalous	0.98	1.00	1.00
Anomalous	0.84	0.50	0.60
Base Model with Class Weighting			
Not Anomalous	0.98	1.00	1.00
Anomalous	0.81	0.41	0.54
Base Model with Hyperparameter Tuning & Class Weighting			
Not Anomalous	0.98	1.00	1.00
Anomlaous	0.78	0.52	0.62

Table 3.9: Precision, recall & f1-score for each class for XGBOOST

3.5.3 MLR



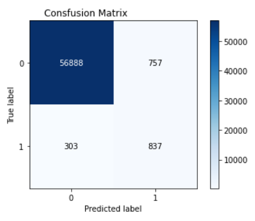
(a) Base model *(b) With hyperparameter tuning* *(c) With class weighting* *(d) With hyperparameter tuning & class weighting*

Figure 3.8: Confusion Matrices for MLR

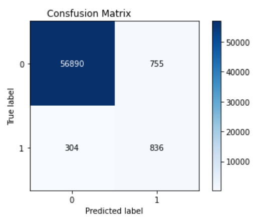
Base Model			
	Precision	Recall	f1-score
Not Anomalous	1.00	0.99	1.00
Anomalous	0.74	0.55	0.63
Base Model with Hyperparameter Tuning			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomalous	0.74	0.56	0.64
Base Model with Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomalous	0.77	0.55	0.64
Base Model with Hyperparameter Tuning & Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomlaous	0.77	0.56	0.65

Table 3.10: Precsion, recall & f1-score for each class for MLR

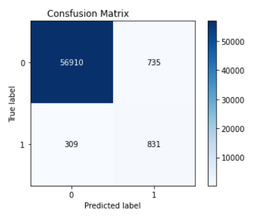
3.5.4 SVM



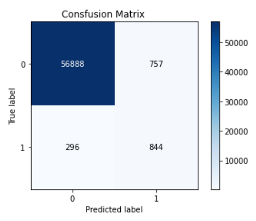
(a) Base model



(b) With hyperparameter tuning



(c) With class weighting



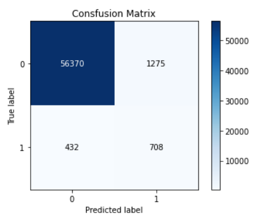
(d) With hyperparameter tuning & class weighting

Figure 3.9: Confusion Matrices for SVM

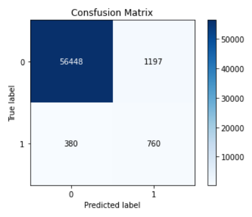
Base Model			
	Precision	Recall	f1-score
Not Anomalous	1.00	0.99	1.00
Anomalous	0.73	0.52	0.61
Base Model with Hyperparameter Tuning			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomalous	0.73	0.52	0.61
Base Model with Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomalous	0.73	0.53	0.61
Base Model with Hyperparameter Tuning & Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomlaous	0.80	0.51	0.62

Table 3.11: Precison, recall & f1-score for each class for SVM

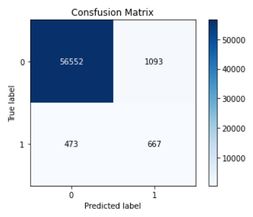
3.5.5 LSTM



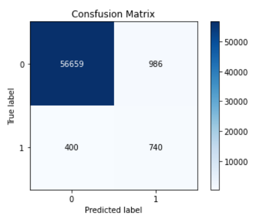
(a) Base model



(b) With hyperparameter tuning



(c) With class weighting



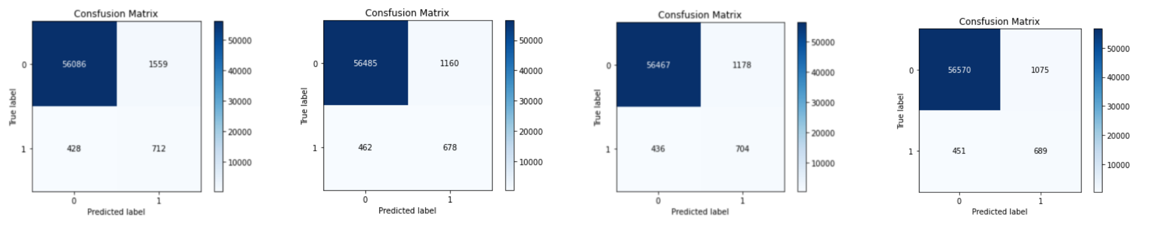
(d) With hyperparameter tuning & class weighting

Figure 3.10: Confusion Matrices for LSTM Machines

Base Model			
	Precision	Recall	f1-score
Not Anomalous	1.00	0.99	1.00
Anomalous	0.62	0.36	0.46
Base Model with Hyperparameter Tuning			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomalous	0.67	0.40	0.50
Base Model with Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomalous	0.58	0.39	0.47
Base Model with Hyperparameter Tuning & Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	1.00	1.00	1.00
Anomlaous	0.65	0.43	0.52

Table 3.12: Precison, recall & f1-score for each class for LSTM

3.5.6 BI-LSTM



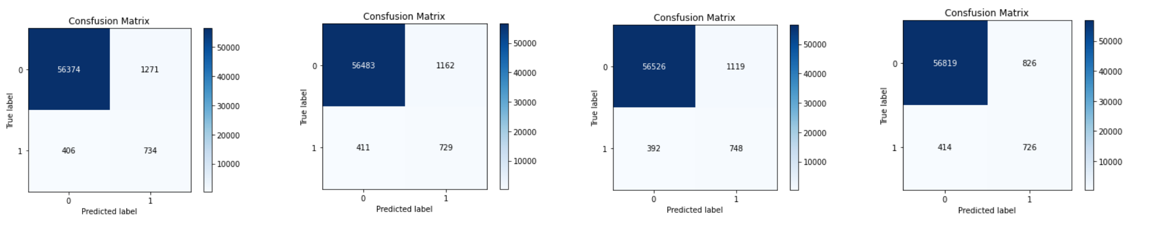
(a) Base model *(b) With hyperparameter tuning* *(c) With class weighting* *(d) With hyperparameter tuning & class weighting*

Figure 3.11: Confusion Matrices for BI-LSTM

Base Model			
	Precision	Recall	f1-score
Not Anomalous	0.97	0.99	0.98
Anomalous	0.62	0.31	0.41
Base Model with Hyperparameter Tuning			
	Precision	Recall	f1-score
Not Anomalous	0.98	0.99	0.98
Anomalous	0.59	0.37	0.44
Base Model with Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	0.98	0.99	0.98
Anomalous	0.62	0.37	0.46
Base Model with Hyperparameter Tuning & Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	0.98	0.99	0.98
Anomlaous	0.60	0.40	0.48

Table 3.13: Precison, recall & f1-score for each class for BI-LSTM

3.5.7 CNN



(a) Base model *(b) With hyperparameter tuning* *(c) With class weighting* *(d) With hyperparameter tuning & class weighting*

Figure 3.12: Confusion Matrices for CNN

	Precision	Recall	f1-score
Base Model			
	Precision	Recall	f1-score
Not Anomalous	0.97	0.99	0.98
Anomalous	0.64	0.37	0.47
Base Model with Hyperparameter Tuning			
	Precision	Recall	f1-score
Not Anomalous	0.98	0.99	0.98
Anomalous	0.64	0.38	0.48
Base Model with Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	0.98	0.99	0.98
Anomalous	0.66	0.37	0.40
Base Model with Hyperparameter Tuning & Class Weighting			
	Precision	Recall	f1-score
Not Anomalous	0.98	0.99	0.98
Anomlaous	0.64	0.47	0.54

Table 3.14: Precison, recall & f1-score for each class for CNN

Chapter 4

Discussion & Study Limitations

4.1 Understanding Model Performance

With respect to the majority class i.e. non-anomalous activities, all the classifiers produce similar results. However, for the problem of anomaly detection, it is more interesting to compare the classifiers based on the minority class. It can be seen from the results that the performance of the classifiers on the minority class is not nearly as good as it is for the majority class. This is precisely why a metric such as accuracy wasn't chosen for evaluation as it skews the results towards the majority class. Moreover, it may be tempting to mainly consider precision since it is generally more valuable to minimize the number of false negatives in a problem of anomaly classification where one would like the maximum number of anomalies to be detected. However, having a large number of false positives also isn't desirable. Therefore, some sort of trade-off between precision and recall is preferred. The f1-score, being the harmonic mean of the two, gives us a better picture of how the model performs with respect to both these aspects. Based on the f1-score, RF performs the best across all four configurations (see Figure 4.1). The performance of classifiers for the experiments in this study can be ranked according the performance of the optimal configuration (with tuning and class weighting) as follows: RF > MLR > SVM > XGBOOST > CNN > LSTM > BI-LSTM. In general, the models perform poorly in terms of recall and have a better precision score (see Figures 4.2 & 4.3). The effects of tuning and class weighting, the two main techniques applied to the base models in this study, will be discussed in the next two subsections.

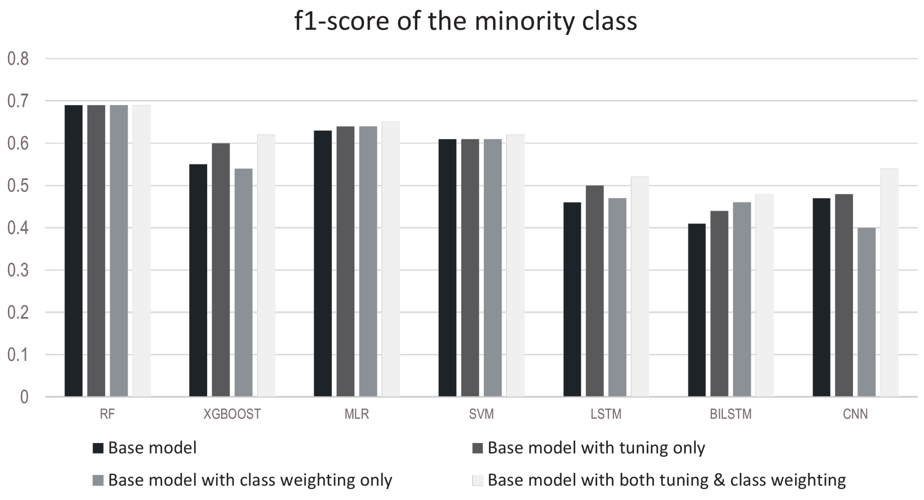


Figure 4.1: f1-score of the classifiers across the four different configurations

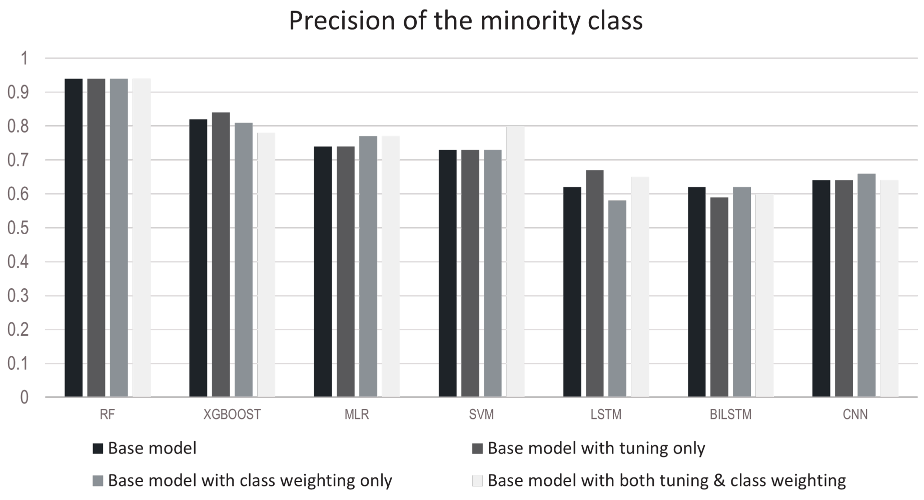


Figure 4.2: Precision of the classifiers across the four different configurations

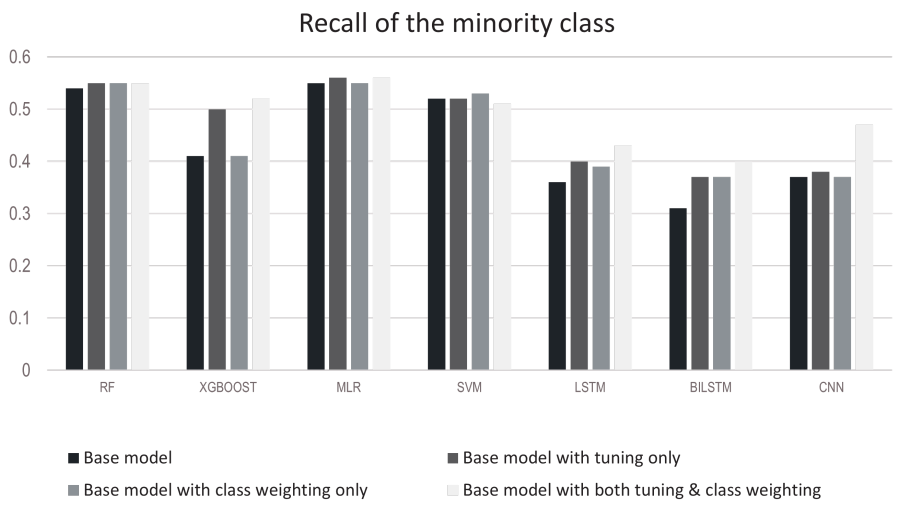


Figure 4.3: Recall of the classifiers across the four different configurations

4.1.1 Effect of Hyperparameter Tuning

From the results, it can be observed that some models are more sensitive to tuning than others, for instance, XGBOOST, LSTM, BI-LSTM and CNN. Along with class weighting, tuning results in an increase of 7%, 6%, 7% and 7% in f1-score, respectively (see figures 3.9, 3.12, 3.13 & 3.14, respectively). The performance of other models do not increase significantly with tuning. For RF, the performance is unaffected with tuning. This means that the default configurations in `scikit-learn` are quite robust and work well as-is for this study. The same can be said about the default configurations of MLR. The SVM classifier is very sensitive to the kernel choice and the value of c. In this study, only the linear and polynomial kernels were considered suitable. Another observation is that, in this study, tuning tries to improve the balance between precision and recall. In other words, it is trying to improve the f1-score. Of course, if specified, tuning can be configured to optimise only precision or only recall. However, as mentioned earlier, a balance between the two is generally preferred. Another main thing to consider when tuning is time and availability of computational resources (good GPU). Tuning for classical machine learning models such as RF, XGBOOST, MLR and SVM can take several hours, while tuning DNNs can take days. Therefore, it is certainly not recommended to exhaustively tune models. This also has implications for model choice. For this particular study, RF performs relatively well under default parameter configurations and training a RF takes only a few minutes.

4.1.2 Effect of Class Weighting

The goal with class weighting is to penalise misclassifications of the minority class, so that false negatives can be minimised, in other words improving the precision score of the minority class. However, from the results of this study, it is observed that class weighting improves precision of some models and recall in others. But in general, class weighting improves the f1-score of all the models. Along with tuning, the benefits of class weighting is more evident with improvements in both precision, recall and the overall f1-score.

4.2 Influence of Process Structure

Processes that are subjected to audit are rather structured, for instance, purchase-to-pay, in that they follow a set of agreed-upon requirements. Typically, the set of requirements can also be modelled using some form of process modelling notation, for instance, Business Process Model Notation (BPMN). A process audit, then, evaluates the series of steps that have taken place in reality against the set of agreed upon requirements. The experiments in this study are also centered around such a structured process. For such kind of processes, a machine learning model could be trained on similar past process executions to perform automated process audits by learning normal and abnormal behaviour. The same cannot be said for unstructured processes with several different process variations. In such cases, a machine learning model may interpret this as noise and not be able to learn the underlying behaviour, consequently leading to a deteriorating model performance.

4.3 Study Limitations

4.3.1 Incorporating Additional Attributes

In this research, only previous activities were considered as input features for the classifier in order to determine whether or not the following activity is anomalous. Normally, an event log contains other information such as the time when the activity was conducted or the resource that conducted the activity. Including these additional attributes could influence the performance of the classifier. One could argue that resource information could be particularly interesting for anomaly detection. However, the event log used in this study did not include this information. Although these attributes could serve as valuable input for a machine learning model, it comes with one caveat - *the curse of dimensionality.* Nevertheless, in such scenarios, feature selection methods could be used to determine useful attributes.

4.3.2 Model Deployment for Audit in the Real World

Model deployment for a real-world audit scenario i.e. integrating the models developed into an existing production environment is out of the scope of this study. Deploying models is one of the most difficult processes of gaining business insights from machine learning. Before a model can be productionised, a lot of coordination is required between data scientists, software engineers and business professionals. Even after a model is successfully launched into a production environment, several other aspects need to considered such as model serving (how the model will be used, for instance, through front-end applications), model performance monitoring and model re-training. This could mean different things for internal and external audit. For instance, constantly monitoring the performance of models on live data and retraining would be more relevant for internal audit than for external audit since external auditors audit past transactions. On the other hand, for external audit, deployment of machine learning models could facilitate automated process audits. In addition to model deployment challenges, the audit profession is also faced with other challenges such as the training and expertise of auditors, as well as challenges associated with data (data availability, reliability and integrity). This study is merely a first step towards practically realising predictive audit scenarios.

4.3.3 Embedding Predictive Process Mining in Audit

This study provides an example of how predictive process mining can be leveraged for both internal and external audit. All recorded business transactions can be utilised to automatically detect deviations. In this manner processes and their internal related controls can be monitored to determine control deficiencies in financial controls. Even though these benefits are significant for the audit profession, more work needs to be done in terms of conceptually situating predictive process mining in financial statement audits such that they also meet the requirements of contemporary audit standards. Another limitation to consider is that the benefits of process mining are bound to the business transactions that have been recorded in some sort of information system. Unrecorded transactions outside the system cannot be inspected through process mining techniques. Additionally, considering how strict audit requirements are, the use of algorithms need to meet certain quality criteria. It is, therefore, not surprising that algorithms need to produce extremely low misclassification rates. For audit, it is preferable to have zero false negatives, and maybe a few false positives so that violations or fradulent transactions may not go undetected. However, in this study, a state of 'zero false negatives' is not achieved. Additionally, the algorithms in this study have high false positive rates. This is also generally the case in practice, where in an attempt to achieve zero false negatives, algorithms produce high false positives [65].

Based on this, it is uncertain if process mining will be leveraged in a predictive manner on an operational level in audit.

Chapter 5

Conclusions & Future Work

In the audit profession, one of the main tasks of an auditor is ensuring that the financial reports produced are accurate and reliable. Since financial reports are essentially outcomes of underlying business processes, monitoring these processes becomes crucial. Due to this, the audit community both in research and practice have taken taken interest in process mining and its use cases for audit.

Before conducting this research, the DSRM was chosen as the research methodology to guide the entire research process. The starting point for this research according to the DSRM was the discovery of a research domain, namely PBPM for which, initially, an explicit problem statement could not be determined. Following the identification of this research domain, the use case for predictive audit was formulated, namely *classifying anomalous traces in audit logs.* This use case would address two problems in audit research and practice that were identified during the first step of the DSRM: (1) process mining is not leveraged fully in audit; and (2) lack of predictive audit use cases. The solution (artefact) would fulfill the objectives set out in the second step of the DSRM by demonstrating how process mining can be leveraged in a predictive manner for a practical audit use case.

As part of the third phase of the DSRM, a procedural method (the artefact) was to be developed which would incorporate elements from both traditional data science pipelines as well as the generic approach adopted in PBPM literature. The use of this procedural method would then be demonstrated by classifying anomalous traces in a loan application process in phase four. This resulted in the formulation of the research goal for this study:

> **RG**: To develop a procedural method that utilises machine learning approaches obtained from PBPM literature to classify anomalous traces in audit logs

In this study, the task of classifying anomalies was formulated as a next activity prediction problem (a supervised multi-class classification problem), and so the research questions to achieve the defined research goal were defined accordingly:

> **RQ**: What is PBPM and what machine learning and deep learning algorithms have been used for next activity prediction in PBPM literature?

To answer this question, a systematic literature review was conducted to first get a broad understanding of the domain of PBPM and then delve deeper into the machine learning and deep learning algorithms used for next activity prediction. During the literature analysis, a noticeable observation was that all studies in this domain adopted a structured process which broadly consisted of two phases - an offline and online phase, with each phase consisting of several sequential processing steps in between. The studies differed not only in terms of algorithms used, but also in the way these steps were conducted. Therefore, there was a need to classify the studies in this domain before a specific implementation could be chosen for this study. Accordingly, the next research question was formulated:

> **SRQ2**: How can these methods be classified?

As mentioned earlier, the experiments conducted in PBPM literature consisted of an online and offline phase . These phases included three main steps, namely, prefix extraction, prefix bucketing and sequence encoding and the studies differed in the way these steps were carried out. Therefore, in addition to classifying the papers based on only the algorithms used, these three steps are also used as the basis for classification. Usually after processing and training an algorithm, the algorithm needs to be evaluated on the basis of some performance metric. Therefore, the third research question was formulated as follows:

> **SRQ3**: What evaluation measures exist in PBPM literature for next activity prediction?

This research question is linked to the previous one. In addition to classifying papers based on processing mechanisms and algorithms used, the papers were also classified on evaluation measures adopted. The classified papers are summarized in Table 2.1.

Finally, the main contributions of this work were addressed by RQ4 and aimed to apply the theoretical foundations obtained from the answers to RQ1, RQ2 and RQ3 to the use case defined for this study as part of the fourth and fifth step of the DSRM

> **SRQ4**: How do the methods identified in RQ1 perform when evaluated using the measures identified in RQ3?

For this task, seven representative algorithms from PBPM literature were chosen, namely RF, XGBOOST, MLR, SVM, LSTM, BI-LSTM and CNN. Depending on the nature of the algorithms, data processing steps were performed accordingly. Additionally, class weighting and tuning were adopted to deal with class imbalance and training performance respectively. Finally, the trained classifiers were evaluated across four configurations (base, with tuning, with class weighting, and with both tuning and class weighting) using confusion matrices. The confusion matrices were used to compute the precision, recall and f1-scores of each class. The RF classifier performed the best across all configurations with the general performance ranking of all classifiers as follows: RF > MLR > SVM > XGBOOST > CNN > LSTM > BI-LSTM. In general it was observed that all classifiers performed the best with tuning and class weighting together. It was also observed that the classifiers had a better precision score that recall across both classes in all configurations.

Recommendations for future work

Based on this study, several directions for future work can be explored:

- It would be interesting to explore the effects of other features in addition to activities on the performance of the classifier;
- It can be argued that the RF classifier performs the best because it is an ensemble model and so for future work, ensembles of different models could be investigated;
- Another direction for future work could be researching how predictive analytical methods such as the one proposed in this study could be embedded in contemporary audit procedures.

Bibliography

[1] M. Vasarhelyi, "The predictive audit framework," *The International Journal of Digital Accounting Research*, vol. 13, pp. 37–71, 01 2012.

[2] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[3] Y. Xiao and M. Watson, "Guidance on conducting a systematic literature review," *Journal of Planning Education and Research*, vol. 39, no. 1, pp. 93–112, 2019. [Online]. Available: https://doi.org/10.1177/0739456X17723971

[4] I. Teinemaa, M. Dumas, M. La Rosa, and F. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, 07 2017.

[5] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F. M. Maggi, "Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes," in *Business Process Management, Bpm 2015*, H. R. MotahariNezhad, J. Recker, and M. Weidlich, Eds., vol. 9253. Berlin: Springer-Verlag Berlin, 2015, pp. 297–313, iSSN: 0302-9743 WOS:000364712100021. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/2

[6] "What is an audit?" https://www.pwc.com/m1/en/services/assurance/what-is-an-audit.html, accessed: 2022-08-26.

[7] M. Jans, M. Alles, and M. Vasarhelyi, "Process mining of event logs in auditing: Opportunities and challenges," *SSRN Electronic Journal*, 02 2010.

[8] M. Jans and M. Eulerich, *Process Mining for Financial Auditing.* Springer International Publishing, 2022, pp. 445–467. [Online]. Available: https://doi.org/10.1007/978-3-031-08848-3_15

[9] M. Jans, M. Alles, and M. Vasarhelyi, "The case for process mining in auditing: Sources of value added and areas of application," *International Journal of*

Accounting Information Systems, vol. 14, no. 1, pp. 1–20, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1467089512000462

[10] F. M. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini, "Predictive monitoring of business processes," in *Advanced Information Systems Engineering*, M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, and J. Horkoff, Eds. Cham: Springer International Publishing, 2014, pp. 457–472.

[11] A. Marquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, "Predictive monitoring of business processes: A survey," *IEEE Transactions on Services Computing*, vol. PP, pp. 1–1, 11 2017.

[12] I. Teinemaa, M. Dumas, F. M. Maggi, and C. D. Francescomarino, "Predictive business process monitoring with structured and unstructured data," in *BPM*, 2016.

[13] H.-J. Kim, M. Mannino, and R. J. Nieschwietz, "Information technology acceptance in the internal audit profession: Impact of technology features and complexity," *International Journal of Accounting Information Systems*, vol. 10, no. 4, pp. 214–228, 2009.

[14] H. Chen, R. H. Chiang, and V. C. Storey, "Business intelligence and analytics: From big data to big impact," *MIS quarterly*, pp. 1165–1188, 2012.

[15] D. Appelbaum, A. Kogan, and M. A. Vasarhelyi, "Big data and analytics in the modern audit engagement: Research needs," *Auditing: A Journal of Practice & Theory*, vol. 36, no. 4, pp. 1–27, 2017.

[16] I. Verenich, M. Dumas, M. L. Rosa, F. M. Maggi, and I. Teinemaa, "Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring," *CoRR*, vol. abs/1805.02896, 2018. [Online]. Available: http://arxiv.org/abs/1805.02896

[17] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *CoRR*, vol. abs/1707.06766, 2017. [Online]. Available: http://arxiv.org/abs/1707.06766

[18] I. Teinemaa, M. Dumas, F. M. Maggi, and C. D. Francescomarino, "Predictive business process monitoring with structured and unstructured data," in *BPM*, 2016.

[19] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, "Predictive Business Process Monitoring with LSTM Neural Networks," in *Advanced Information Systems Engineering (caise 2017)*, E. Dubois and K. Pohl, Eds., vol. 10253. Cham:

Springer International Publishing Ag, 2017, pp. 477–492, iSSN: 0302-9743 WOS:000428781000030. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/1

[20] S. Weinzierl, S. Zilker, J. Brunk, K. Revoredo, M. Matzner, and J. Becker, "XNAP: Making LSTM-Based Next Activity Predictions Explainable by Using LRP," *Lecture Notes in Business Information Processing*, vol. 397, pp. 129–141, 2020, iSBN: 9783030664978.

[21] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Using Convolutional Neural Networks for Predictive Process Analytics," in *2019 International Conference on Process Mining (icpm 2019).* New York: Ieee, 2019, pp. 129–136, wOS:000494803300017. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/2

[22] A. Nguyen, S. Chatterjee, S. Weinzierl, L. Schwinn, M. Matzner, and B. Eskofier, "Time Matters: Time-Aware LSTMs for Predictive Business Process Monitoring," *Lecture Notes in Business Information Processing*, vol. 406 LNBIP, pp. 112–123, 2021, iSBN: 9783030726928.

[23] F. Folino, M. Guarascio, and L. Pontieri, "Discovering context-aware models for predicting business process performances," in *On the Move to Meaningful Internet Systems: OTM 2012.* Berlin, Heidelberg: springer Berlin Heidelberg, 2012, pp. 287–304.

[24] A. Bevacqua, M. Carnuccio, F. Folino, M. Guarascio, and L. Pontieri, "A data-adaptive trace abstraction approach to the prediction of business process performances," in *ICEIS*, 01 2013, pp. 56–65.

[25] C. Cabanillas, C. Di Ciccio, J. Mendling, and A. Baumgrass, "Predictive Task Monitoring for Business Processes," in *Business Process Management, Bpm 2014*, S. Sadiq, P. Soffer, and H. Volzer, Eds., vol. 8659. Berlin: Springer-Verlag Berlin, 2014, pp. 424–432, iSSN: 0302-9743 WOS:000345316000031. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/1

[26] R. Conforti, M. de Leoni, M. La Rosa, W. M. van der Aalst, and A. H. ter Hofstede, "A recommendation system for predicting risks across multiple business process instances," *Decision Support Systems*, vol. 69, pp. 1–19, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923614002516

[27] C. Di Francescomarino, C. Ghidini, F. Maggi, G. Petrucci, and A. Yeshchenko, "An eye into the future: Leveraging a-priori knowledge in predictive business pro-

cess monitoring," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10445 LNCS, pp. 252–268, 2017, iSBN: 9783319649993.

[28] A. E. Márquez-Chamorro, M. Resinas, and A. Ruiz-Cortés, "Predictive monitoring of business processes: A survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 962–977, 2018.

[29] J. Kim and M. Comuzzi, "Stability metrics for enhancing the evaluation of outcome-based business process predictive monitoring," *IEEE Access*, vol. 9, pp. 133 461–133 471, 2021.

[30] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "A multi-view deep learning approach for predictive business process monitoring," *IEEE Transactions on Services Computing*, 2021.

[31] A. Jalayer, M. Kahani, A. Pourmasoumi, and A. Beheshti, "HAM-Net: Predictive Business Process Monitoring with a hierarchical attention mechanism," *Knowledge-Based Systems*, vol. 236, 2022.

[32] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, "Leveraging multi-view deep learning for next activity prediction," vol. 2952, 2021, pp. 1–6, iSSN: 1613-0073.

[33] S. Weinzierl, "Exploring Gated Graph Sequence Neural Networks for Predicting Next Process Activities," in *Business Process Management Workshops, Bpm 2021*, A. Marrella and B. Weber, Eds., vol. 436. Cham: Springer International Publishing Ag, 2022, pp. 30–42, iSSN: 1865-1348 WOS:000754564600003. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/2

[34] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees.* Routledge, 2017.

[35] J. Fürnkranz, *Decision Tree.* Boston, MA: Springer US, 2010, pp. 263–267. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8_204

[36] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https://doi.org/10.1023/a:1010933404324

[37] T. Chen and C. Guestrin, "XGBoost," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, aug 2016. [Online]. Available: https://doi.org/10.1145%2F2939672.2939785

[38] V. Y. Kulkarni and P. Sinha, “Random forest classifiers: A survey and future research directions,” *International Journal of Advanced Computing*, vol. 36, no. 1, pp. 1144 – 1153, 2013.

[39] R. Sindhgatta, C. Ouyang, and C. Moreira, “Exploring Interpretability for Predictive Process Analytics,” in *Service-Oriented Computing (icsoc 2020)*, E. Kafeza, B. Benatallah, F. Martinelli, H. Hacid, A. Bouguettaya, and H. Motahari, Eds., vol. 12571. Cham: Springer International Publishing Ag, 2020, pp. 439–447, iSSN: 0302-9743 WOS:000716943300031. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/2

[40] J. Wang, D. Yu, C. Liu, and X. Sun, “Outcome-Oriented Predictive Process Monitoring with Attention-based Bidirectional LSTM Neural Networks,” in *2019 Ieee International Conference on Web Services (ieee Icws 2019)*, E. Bertino, C. K. Chang, P. Chen, E. Damiani, M. Goul, and K. Oyama, Eds. New York: Ieee, 2019, pp. 360–367, wOS:000517091800052. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/1

[41] A. Ng, “Cs229 lecture notes,” *CS229 Lecture notes*, vol. 1, no. 1, pp. 1–3, 2000.

[42] G. Foody and A. Mathur, “A relative evaluation of multiclass image classification by support vector machines,” *Geoscience and Remote Sensing, IEEE Transactions on*, vol. 42, pp. 1335 – 1343, 07 2004.

[43] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of cnn and rnn for natural language processing,” *arXiv preprint arXiv:1702.01923*, 2017.

[44] “Mit 6.s191: Recurrent neural networks,” https://authurwhywait.github.io/blog/2021/12/02/introduction_to_dl02/, accessed: 2010-09-30.

[45] Z. C. Lipton, J. Berkowitz, and C. Elkan, “A critical review of recurrent neural networks for sequence learning,” 2015. [Online]. Available: https://arxiv.org/abs/1506.00019

[46] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou, “Patient subtyping via time-aware lstm networks,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 65–74. [Online]. Available: https://doi.org/10.1145/3097983.3097997

[47] J. Brunk, J. Stottmeister, S. Weinzierl, M. Matzner, and J. Becker, "Exploring the effect of context information on deep learning business process predictions," *Journal of Decision Systems*, vol. 29, pp. 328–343, Aug. 2020, place: Abingdon Publisher: Taylor & Francis Ltd WOS:000547893800001. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/1

[48] A. Jalayer, M. Kahani, A. Beheshti, A. Pourmasoumi, and H. R. Motahari-Nezhad, "Attention Mechanism in Predictive Business Process Monitoring," in *2020 Ieee 24th International Enterprise Distributed Object Computing Conference (edoc 2020)*. Los Alamitos: Ieee Computer Soc, 2020, pp. 181–186, iSSN: 2325-6354 WOS:000630246800020. [Online]. Available: http://gateway.webofknowledge.com/gateway/Gateway.cgi?GWVersion=2&SrcAuth=DynamicDOIConfProc&SrcApp=WOS&KeyAID=10.1109%2FEDOC49727.2020.00030&DestApp=DOI&SrcAppSID=EUW1ED0E7EkS8dVeqT9Hd9grMFLes&SrcJTitle=2020+IEEE+24TH+INTERNATIONAL+ENTERPRISE+DISTRIBUTED+OBJECT+COMPUTING+CONFERENCE+%28EDOC+2020%29&DestDOIRegistrantName=Institute+of+Electrical+and+Electronics+Engineers

[49] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.08458

[50] F. Folino, M. Guarascio, A. Liguori, G. Manco, L. Pontieri, and E. Ritacco, "Exploiting Temporal Convolution for Activity Prediction in Process Analytics," *Communications in Computer and Information Science*, vol. 1323, pp. 263–275, 2020, iSBN: 9783030659646.

[51] M. Jin, J. Ye, J. Luo, and Y. Lin, "Predictive Monitoring Algorithm Based on Global Feature Encoding," in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, Oct. 2020, pp. 1–6.

[52] N. Di Mauro, A. Appice, and T. Basile, "Activity Prediction of Business Process Instances with Inception CNN Models," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11946 LNAI, pp. 348–361, 2019, iSBN: 9783030351656.

[53] S. L. Fraley Chris, Raftery Adrian. (1999) Brier score to assess the accuracy of probabilistic predictions. [Online]. Available: https://mclust-org.github.io/mclust/reference/BrierScore.html

[54] A. Metzger and A. Neubauer, "Considering Non-sequential Control Flows for Process Prediction with Recurrent Neural Networks," in *44th Euromicro*

Conference on Software Engineering and Advanced Applications (seaa 2018), T. Bures and L. Angelis, Eds. New York: Ieee, 2018, pp. 268–272, iSSN: 1089-6503 WOS:000450238900042. [Online]. Available: http://www.webofscience.com/wos/woscc/summary/c1a437b1-7cce-4f7a-bc26-4725ed055636-3784b3a8/relevance/2

[55] I. Teinemaa, M. Dumas, F. Maria Maggi, and C. Di Francescomarino, "Predictive Business Process Monitoring with Structured and Unstructured Data," in *Business Process Management, Bpm 2016*, M. LaRosa, P. Loos, and O. Pastor, Eds., vol. 9850. Cham: Springer International Publishing Ag, 2016, pp. 401–417, iSSN: 0302-9743 WOS:000388721400023. [Online]. Available: http://gateway.webofknowledge.com/gateway/Gateway.cgi?GWVersion=2&SrcAuth=DynamicDOIConfProc&SrcApp=WOS&KeyAID=10.1007%2F978-3-319-45348-4_23&DestApp=DOI&SrcAppSID=EUW1ED0E7EkS8dVeqT9Hd9grMFLes&SrcJTitle=BUSINESS+PROCESS+MANAGEMENT%2C+BPM+2016&DestDOIRegistrantName=Springer-Verlag

[56] F. Taymouri, M. Rosa, S. Erfani, Z. Bozorgi, and I. Verenich, "Predictive business process monitoring via generative adversarial nets: The case of next event prediction," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 12168 LNCS, pp. 237–256, 2020, iSBN: 9783030586652.

[57] Q. Liu and B. Wu, "Prediction of Business Process Outcome based on Historical Log," in *Proceedings of the 10th International Conference on Computer Modeling and Simulation*, ser. ICCMS 2018. New York, NY, USA: Association for Computing Machinery, Jan. 2018, pp. 118–122. [Online]. Available: http://doi.org/10.1145/3177457.3177465

[58] S. Baskoro and W. D. Sunindyo, "Predicting Issue Handling Process using Case Attributes and Categorical Variable Encoding Techniques," in *2019 International Conference on Data and Software Engineering (ICoDSE)*, Nov. 2019, pp. 1–5, iSSN: 2640-0227.

[59] W. Rizzi, C. Di Francescomarino, and F. Maggi, "Explainability in predictive process monitoring: When understanding helps improving," *Lecture Notes in Business Information Processing*, vol. 392 LNBIP, pp. 141–158, 2020, iSBN: 9783030586379.

[60] S. Biswas, M. Wardat, and H. Rajan, "The art and practice of data science pipelines: A comprehensive study of data science pipelines in theory, in-the-small, and in-the-large," *CoRR*, vol. abs/2112.01590, 2021. [Online]. Available: https://arxiv.org/abs/2112.01590

[61] S. J. van Zelst, M. Fani Sani, A. Ostovar, R. Conforti, and M. La Rosa, "Filtering spurious events from event streams of business processes," in *Advanced Information Systems Engineering.* Springer International Publishing, 2018, pp. 35–52.

[62] I. Teinemaa, M. Dumas, M. L. Rosa, and F. M. Maggi, "Outcome-oriented predictive process monitoring: Review and benchmark," *CoRR*, vol. abs/1707.06766, 2017. [Online]. Available: http://arxiv.org/abs/1707.06766

[63] M. L. Guilly, "Guided data selection for predictive models," Ph.D. dissertation, INSA Lyon, 2020.

[64] G. King and L. Zeng, "Logistic regression in rare events data," *Political Analysis*, vol. 9, pp. 137 – 163, 2001.

[65] G. Baader and H. Krcmar, "Reducing false positives in fraud detection: Combining the red flag approach with process mining," *International Journal of Accounting Information Systems*, vol. 31, pp. 1–16, 2018.