Les Cahiers du GERAD ISSN: 0711–2440

Comptes rendus du 13e atelier de résolution de problèmes industriels de Montréal, 21–25 août 2023

Proceedings of the 13th Montréal Industrial Problem Solving Workshop, August 21–25, 2023

Odile Marcotte, éditrice

G–2024–21

Mars 2024

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2024
– Bibliothèque et Archives Canada, 2024

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2024
– Library and Archives Canada, 2024

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Préface

Le Treizième atelier de résolution de problèmes industriels de Montréal, qui eut lieu du 21 au 25 août 2023, fut organisé conjointement par le Centre de recherches mathématiques (CRM) et l'Institut de valorisation des données (IVADO). Plus de 80 personnes s'inscrivirent à l'atelier et examinèrent neuf problèmes, fournis respectivement par Air Canada, Beneva, Desjardins, Ericsson, Hitachi, Intact, Radio-Canada et la Société de transport de Montréal. Je remercie chaleureusement ces partenaires et les coordonnateurs des équipes (Frédéric Quesnel, Juli Atherton, Sébastien Gambs, Fabian Bastin, Nassim Razaaly, Jean-François Plante, Sean Bohun, Moncef Chioua, Philippe Gagnon, Mike Lindstrom et Nima Akbarzadeh), ainsi que les conseillers IVADO dont la collaboration fut essentielle à la collecte des problèmes. Finalement j'exprime toute ma reconnaissance à Karine Hébert, qui a mis en forme ces comptes rendus.

Odile Marcotte
Conseillère spéciale aux partenariats, CRM
Professeure associée, UQAM et membre associé, GERAD

CENTRE DE RECHERCHES MATHÉMATIQUES

IVADO

Foreword

The Thirteenth Montreal IPSW took place on August 21-25, 2023, and was jointly organized by the Centre de recherches mathématiques (CRM) and the Institute for Data Valorization (IVADO). More than 80 participants registered for the workshop and studied nine problems, submitted by Air Canada, Beneva, Desjardins, Ericsson, Hitachi, Intact, Radio-Canada, and the STM (Société de transport de Montréal). I am grateful to our industrial partners and the team coordinators (Frédéric Quesnel, Juli Atherton, Sébastien Gambs, Fabian Bastin, Nassim Razaaly, Jean-François Plante, Sean Bohun, Moncef Chioua, Philippe Gagnon, Mike Lindstrom, and Nima Akbarzadeh), as well as the IVADO advisors who helped us find the problems submitted to the workshop. I am also very grateful to Karine Hébert, who put these proceedings together.

Odile Marcotte
Special Advisor, Partnerships, CRM
Adjunct Professor, UQAM and Associate Member, GERAD

Contents

1 Air Canada: Unit load device forecasting

Guilherme Augusto [a]
Tosin Babasola [b]
Ali Barooni [c, d]
Saeedeh Dehghani [e]
Joy Liu [f]
Frédéric Quesnel [e, d]

[a] *Air Canada*
[b] *University of Bath*
[c] *Polytechnique Montréal*
[d] *GERAD*
[e] *UQAM*
[f] *Dalhousie University*

March 2024
Les Cahiers du GERAD

Abstract: *Aircrafts typically carry cargo and luggage in or on pellets or containers called unit load devices (ULD). A good ULD management strategy is necessary to ensure airline operations run smoothly. A key challenge Air Canada faces is predicting the number of ULDs that will be used for each flight. Predicting the number of unit load devices (ULDs) of each type used for each flight is of great interest for Air Canada. It enables them to develop good ULD management strategies and save money by limiting the number of empty ULDs shipped. In the Thirteenth Montreal Industrial Problem Solving Workshop, we developed two models to predict the number of ULDs required for each flight. The first model is a statistical model based on the zero-inflated Poisson distribution. The second one is a machine learning model based on the XGBoost library. We show that both models have a relatively good level of accuracy and are competitive. In particular, our models are able to predict ULD usage within one unit on average. We highlight some limitations of our approach and propose some ways to overcome them.*

1.1 Introduction

1.1.1 Context

The airline transportation industry is extremely competitive, and as a result, airlines have relatively low profit margins. An airline's main source of expenditure is the purchase of fuel for its aircraft. It is therefore essential to maximize aircraft utilization to make every flight profitable. One strategy employed by many airlines is to carry cargo on commercial flights. Some airlines also have a fleet of aircraft dedicated solely to cargo.

Transporting cargo by air is a complex task. A planner must first assign items to unit loading devices (ULDs). ULDs are shipping containers or pallets specially designed for air freight. There are different types of ULDs, each distinguished by its geometric characteristics, which influence its loading capacity and the positions it can occupy in an aircraft. Some examples of ULDs are shown in Figure 1.1. Once items have been placed in suitable ULDs, the ULDs are loaded into the aircraft according to a specific pattern that balances the load in the aircraft (see Figure 1.2 for examples of configurations). Passenger baggage is assigned to separate ULDs and given priority (the aircraft will only carry cargo if there is space available).

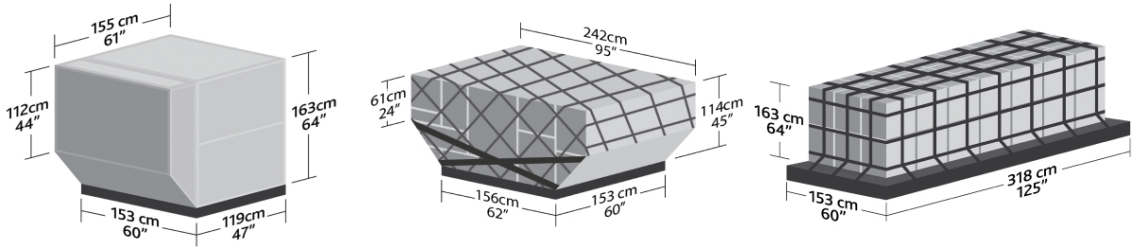


Figure 1.1: Examples of ULDs.

The management of ULDs poses a major challenge: the demand for each type of ULD, as well as the amount of space available on the aircraft (depending on the volume of passenger baggage), are not known until shortly before the flight. It is therefore difficult to predict exactly how many ULDs of each type will be needed for a given flight. Furthermore, demand for cargo is unbalanced between airports: many airports export more products than they receive, or vice versa. As a result, empty ULDs frequently have to be transported from one airport to another. Since this is a costly operation, it is advisable to limit empty transport.

Mismanagement of ULDs can lead to a number of problems. A slight shortage of ULDs could result in a reduction in cargo-carrying capacity, leading to a loss of revenue. It is also possible that

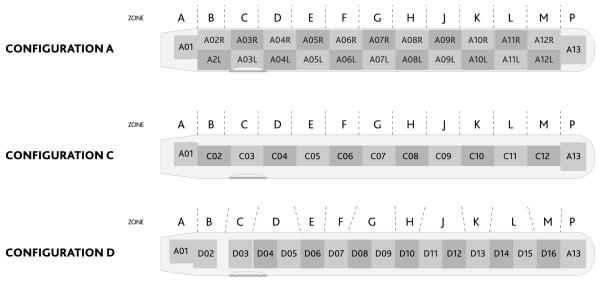


Figure 1.2: Examples of configurations.

cargo transport could be significantly delayed, leading to delay penalties. In the event of a serious shortage, the transport of passenger baggage could be compromised, resulting in immense costs and loss of reputation for the airline. Finally, storing too many ULDs at an airport or needlessly transporting empty ULDs could entail substantial extra costs.

A good long-term ULD management strategy requires a good forecast of the demand for each ULD type at each airport. Current ULD management strategies found in the literature are generic inventory management methods based on the notions of safety stock and reorder point. These methods are inefficient and costly for airlines, as ULD demand varies greatly from month to month [4]. Also ULDs are to some extent interchangeable (an item can be placed in any one of several types of ULD, and several types of ULDs can occupy the same space in the aircraft), which complicates inventory management.

1.1.2 Research questions

For the purposes of this workshop, our group decided to focus on the following three research questions:

1. How to predict the number of ULDs of each type that will be required for each flight?
2. Is it possible to refine those predictions by predicting how many ULDs of each type will be used for baggage vs. cargo?
3. Are those predictions physically feasible, i.e., can the predicted ULDs form a valid configuration for the aircraft?

For reasons explained in Section 1.2.2, we focus on the three ULD types that are most widely used, corresponding to around 80% of all ULDs used.

We propose making ULD predictions using statistical models built on the basis of historical data. Specifically, we create one model for each type of ULD considered. This historical data also specifies which ULDs were used for baggage and cargo, which will enable us to answer Question 2. To tackle Question 3, we will use the predictions of the statistical model and use a simplified set of validity rules to determine whether the predictions are realistic, i.e., whether they correspond to a valid ULD configuration.

The remainder of this report is structured as follows. In Section 6.2, we present the data provided by Air Canada. We analyze this data by making visual representation of the key features in Section 1.3. The two tested models are explained in Section 1.4. Results are presented in Section 1.5 and conclusions are drawn in Section 6.6.

1.2 Data

1.2.1 Raw data

Air Canada provided us with ULD data for an 18-month period (from January 2022 to June 2023). This data contains information on more than 2.5 million ULDs for around 145 000 flight legs. Each data point consists of 46 fields pertaining to a specific ULD on a specific flight leg. Of those 46 fields, only the following were deemed relevant:

Data related to the flight leg the ULD was loaded on:

- Date. Specifically, we found that the month of the flight is a good predictor of the number of ULDs needed.
- Flight number. The flight number only serves to identify a given flight. A flight leg is uniquely identified by its number, date, and in the case of multi-leg flights, its origin and destination.
- Origin.
- Destination.
- Whether the leg belongs to a *multi-leg flight*. A multi-leg flight occurs when the airplane visits some other airports between its origin and its destination. Multi-leg flights occur most often for pure cargo flights.

Data related to the ULD:

- ULD type.
- Whether the ULD contained baggage or cargo.

1.2.2 Data analysis

ULD types

There are 21 distinct ULD types in the data. Three of those, however, denoted by codes AKE, AKH, and PMC, account for around 78% of all ULDs used. For this reason, and given the short timeframe of the workshop, we decided to focus on predicting those ULDs.

A PMC is a pallet-type ULD, and as such, is almost exclusively used for cargo. It is larger than the other types of ULDs, taking 4 space units in the aircraft whereas other types take 1 each (more details are given in Section 1.2.2).

The AKH ULDs are specially made for aircraft types 320 and 321 (see Section 1.2.2). On the other hand AKEs and PMCs can be used on all aircraft types except 320 and 321. Note that those rules may be broken under extraordinary circumstances.

Aircraft type

The data contains 15 aircraft types. Only 8 of those, however, are *containerized*, meaning that they can accommodate ULDs. The data pertaining to the non-containerized aircraft types were thus excluded.

The remaining 8 containerized aircraft types are displayed in Table 1.1. For each aircraft type, we give the number of flight legs using that aircraft type, the number of space units available for each ULD type in that aircraft, and whether this aircraft type is used exclusively for cargo or not.

Table 1.1: Characteristics of aircraft types.

Aircraft type	Flight Number	ULD Positions			Only Cargo?
		AKE	AKH	PMC	
788	8325	14	0	9	No
77W	16768	22	0	14	No
789	27244	18	0	11	No
77L	4177	16	0	10	No
321	41740	0	9	0	No
333	16085	16	0	11	No
320	29828	0	7	0	No
76F	3235	16	0	20	Yes

Valid ULD configurations

The ULD placement in an aircraft is governed by a complex set of rules. Indeed, each ULD type has a different volume as well as distinct geometrical properties. Each aircraft type has several ULD placement configurations. It is also necessary to balance the overall weight inside the aircraft for security reasons.

For our application, it is not necessary that our predictions form a valid ULD configuration since the goal is to ensure enough ULDs of each type are available. Good predictions, however, should lead to outcomes that are close to valid configurations.

To tackle this question, we use simplified configuration rules. We consider only one configuration for each aircraft type (the most commonly used one). We then limit the number of ULDs of each type to the number of ULDs of that type in the chosen configuration. Those numbers are displayed in Table 1.1. To allow for some flexibility, we permit exceeding those limits by a certain factor, e.g., 20% in our experiments. A configuration is deemed valid if the aircraft has enough space units to accommodate all ULDs.

1.3 Data visualization

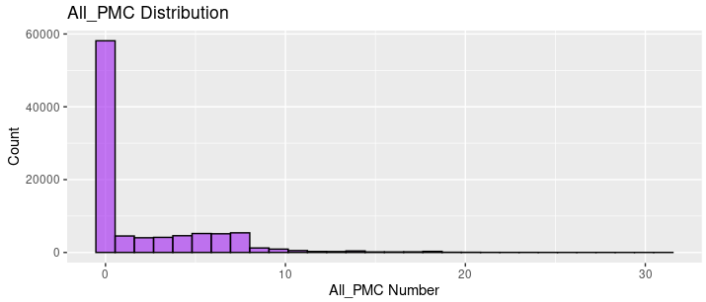
A preliminary step in creating a statistical model is understanding the relationship between the explanatory variables and the response. To do so, we created several box plots of the data, illustrating the relationship of each relevant variable with the number of ULDs of each type.

We first look at the overal distribution of the number of each ULD type (Figure 1.3). We notice that the shape of those distributions is that of a *Zero-Inflated Poisson distribution*. This observation will be the basis of the statistical model.

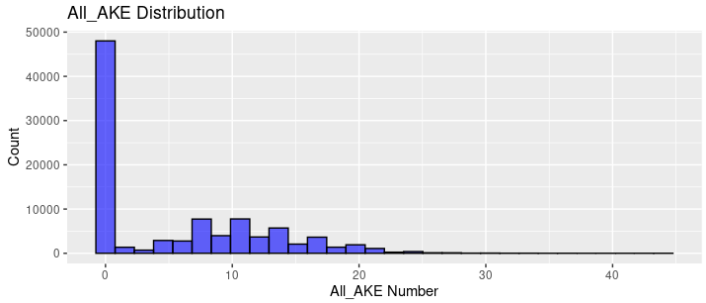
Then we further break down these distributions by aircraft type. Figure 1.4 present the ULD number distribution for each aircraft type and each ULD type as a box plot. As expected, PMCs and AKEs are almost never used on aircraft types 320 and 321. Similarly, AKHs are almost exclusively used on aircraft types 320 and 321. We notice that when permissible, AKEs are almost always used on legs. This is because AKEs are primarily used to transport passenger baggage. In the same spirit, we notice that PMCs represent a high percentage of all ULDs for aircraft type 76F. This is also expected because aircraft type 76F only transports cargo.

Next we look at seasonal pattern in ULD usage. To do this, we break down the ULD distribution by month and ULD type, displayed in Figure 1.5. The usage of AKE-type ULDs appear to be relatively stable. We notice, however, that the usage pattern of PMC-type ULDs is somewhat the inverse of that for the AKH type: whereas there are demand peaks in the months of November, December, March, April, June, and July for AKH, those are low-usage months for PMC. This is probably because PMCs are mostly used to transport cargo and AKHs mostly used for passenger baggage. In high-demand

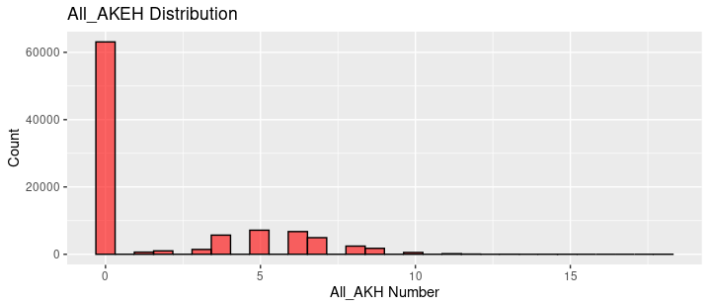
periods for passenger travel, more space is required for passenger baggage, so there is less cargo space remaining, hence the lower usage of PMCs.



(a) PMC distribution.



(b) AKE distribution.



(c) AKH distribution.

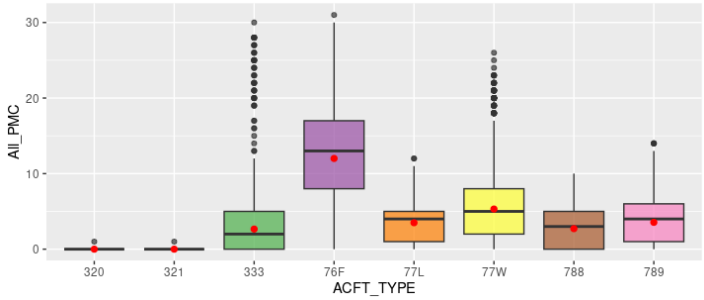
Figure 1.3: ULD number per ULD type.

1.4 Methodology

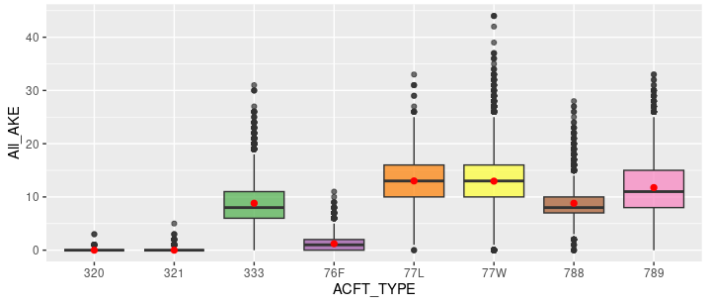
In our analysis, we evaluated two distinct predictive models. The initial model leverages the Zero-Inflated Poisson (ZIP) distribution, which is comprehensively detailed in Section 1.4.1. This model is particularly adept at handling count data characterized by an excess of zeros. The second model we explored is based on the eXtreme Gradient Boosting (XGBoost) algorithm, a powerful machine learning technique known for its efficiency and effectiveness in various predictive scenarios. The implementation and specifics of this model are described in Section 1.4.2.

1.4.1 Zero-inflated Poisson model

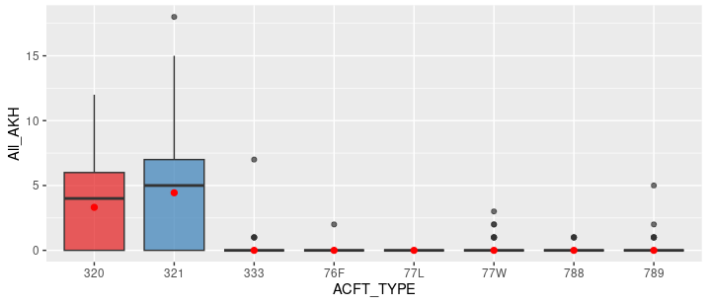
Zero-Inflated Poisson (ZIP) models are specialized tools for counting data with a high occurrence of zeros, a scenario where traditional counting methods, such as the standard Poisson distribution, fall short. These models are particularly valuable in fields like health studies, environmental research, and economics, where data often include many zeros. ZIP models operate on the premise that the data



(a) PMC type.



(b) AKE type.



(c) AKH type.

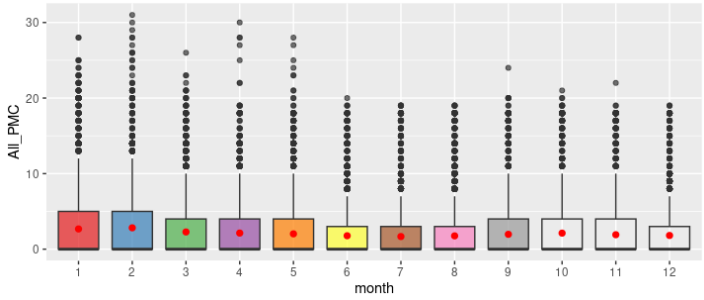
Figure 1.4: Box plot of the ULD distribution by ULD type and aircraft type.

arises from two distinct processes: one that only generates zeros and another that follows a conventional counting pattern (a Poisson distribution), producing both zeros and other numbers. This dual approach allows the ZIP model to deal more effectively with the abundance of zeros than traditional methods.

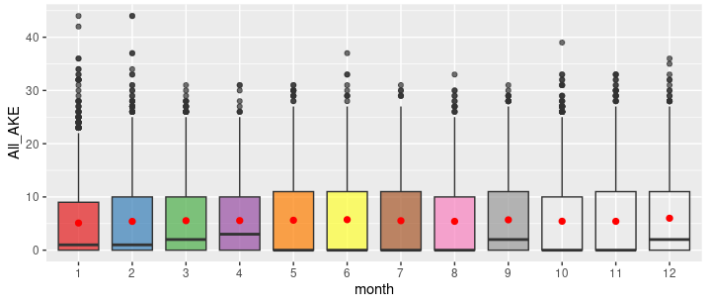
In our research we observe a pattern akin to the one found in the Printed Wiring Board experiment, as described by Lambert in her seminal paper on Zero-Inflated Poisson models [3]. As in Lambert's findings, our data analysis reveals a significant presence of zeros.

The ZIP model comprises two components: a traditional Poisson count model and a logistic model for the excess zeros. This setup enables the model to handle separately the likelihood of zeros generated exclusively by the zero-only process and the mixed counts from the Poisson process. Such a configuration is pivotal in dealing with over-dispersion, a common challenge in count data analysis where data variance surpasses the mean. ZIP models are thus instrumental for deciphering accurately and predicting scenarios prevalent in zero-inflated data.

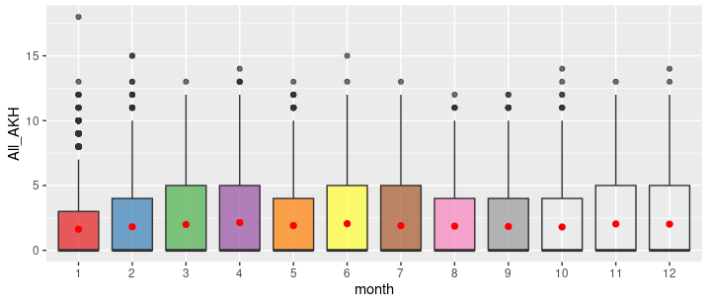
Our analysis, particularly the presence of numerous zeros, is consistent with Lambert's observations, as shown in the visualization section 1.3; it is also evident in the distribution depicted in Figure 1.3. The implementation of our analysis was conducted using RStudio for the ZIP model.



(a) Month vs PMC.



(b) Month vs AKE.



(c) Month vs AKH.

Figure 1.5: Box plot of the ULD distribution by ULD type and by month.

1.4.2 XGBoost model

XGBoost, an acronym for eXtreme Gradient Boosting, is a state-of-the-art machine learning algorithm that has revolutionized the field of data science. This advanced algorithm is an extension of gradient boosting machines and is renowned for its efficiency, flexibility, and portability. XGBoost has gained popularity for its exceptional performance in predictive accuracy and its ability to work on large and complex datasets. It is widely used in competitive data science platforms, like Kaggle, and in various industry applications [1]. As outlined in Chen and Guestrin's paper (2016) [2], XGBoost employs a novel tree learning algorithm that effectively handles sparse data and improves computational speed. Figure 1.6 depicts the process of summing up gradient and second-order gradient statistics on each leaf of the decision tree. It illustrates how XGBoost applies its scoring formula to these aggregated statistics to compute the quality score of the tree structure, demonstrating a key aspect of the algorithm's efficiency in model training.

As we transition our dataset into a tabular format (discussed in Section 1.2.2), XGBoost emerges as a superior choice. This preference is based on its proven effectiveness in similar dataset contexts. Its application often results in enhanced performance and accuracy, as illustrated in various case studies and benchmarks presented in the original paper [2].

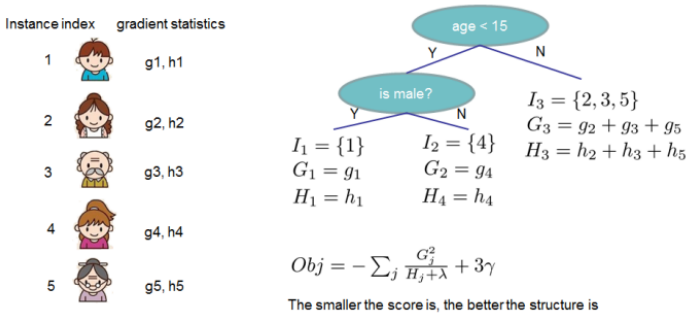


Figure 1.6: Efficient Handling of Sparse Data in the XGBoost Algorithm.

The table below outlines the specific parameters we have utilized in our XGBoost model. These parameters were chosen to optimize the model's performance for our specific dataset needs. The parameter selection is influenced by the guidelines and best practices suggested in the XGBoost paper.

Table 1.2: Parameters Used in the XGBoost Model.

Parameter	Value
Objective	'reg:squarederror'
Evaluation Metric	'rmse'
Number of Estimators	100
Verbosity Level	2
Test Size (for train-test split)	0.2
Random State (for train-test split)	42

In summary, XGBoost's design, as detailed by Chen and Guestrin (2016) [2], offers a robust framework for tackling complex machine learning challenges. Its ability to process large datasets with high efficiency and accuracy makes it an indispensable tool in the arsenal of data scientists.

1.5 Results

1.5.1 First research question

To assess the accuracy of each prediction model, we compute their *mean absolute error* (MAE) for each type of airplane and each ULD type. Results are presented in Table 1.3.

Table 1.3: Mean absolute error for each aircraft type and ULD type.

Aircraft type	AKE		PMC		AKH	
	XGB	ZIP	XGB	ZIP	XGB	ZIP
788	2.19	1.91	1.73	1.40	0.05	0.24
77W	3.17	3.02	2.42	2.12	0.03	0.18
789	2.68	2.36	2.02	1.70	0.04	0.10
77L	2.65	2.57	1.72	1.54	0.05	0.30
321	0.14	0.00	0.06	0.00	2.23	4.28
333	2.35	1.95	1.95	1.45	0.04	0.12
320	0.14	0.00	0.08	0.00	2.12	3.34
76F	1.36	1.09	4.19	4.29	0.03	0.28

We first examine the ZIP model. For ULD types AKE and PMC, and aircraft types 320 and 321, the ZIP model has perfect accuracy. This is expected since AKEs and PMCs don't fit into aircraft

types 320 and 321, so the value to predict is always 0. Similarly, AKHs are seldom used in aircraft types other than 320 and 321, which explains the high degree of accuracy for those predictions. We observe, however, that the ZIP model performs poorly for AKHs for aircraft types 320 and 321. This is because a single ZIP model was parametrized for all ULD types, so it predicts an overabundance of zero values for AKHs. For PMC and AKE ULDs and all aircraft types except 320 and 321, the ZIP predictions are generally between 1 and 3. Given the high variance in the data, those results are relatively good.

Table 1.4: Mean absolute error (standard deviation) per ULD type for both models.

Models	All_AKE	All_PMC	All_AKH
XGBoost	1.40 (1.48)	1.15 (1.20)	1.08 (1.21)
ZIP	1.19 (1.38)	0.95 (1.15)	1.98 (2.39)

The XGBoost model exhibits performances similar to those of the ZIP model. Although its accuracy is slightly higher for AKEs and PMCs, it is slightly lower for AKHs. Although XGBoost suffers from the same prediction problem as ZIP for AKH ULDs and aircraft types 320 and 321, the prediction error is significantly lower. This indicates that XGBoost is better able to distinguish between ULD types in making predictions.

1.6 Second and third research question

Unfortunately, we lacked time to tackle our second research question (refining our prediction for baggage/cargo). We had access to some of the historical baggage data but it was hard to interpret so we were unable to include it into our prediction. Furthermore, given the accuracy of the overall ULD predictions, it was unlikely that we could have built an accurate prediction model for baggage based on our current approaches.

We were able to give a partial answer to the third research question (whether the predicted ULDs form a valid configuration). To do so, we used simplified configuration rules and checked how often those rules were respected. Note that those rules are rudimentary and only take into account the three ULD types tackled in this study. Unfortunately, preliminary findings indicate that the predicted ULDs usually do not form a valid configuration, unless very few ULDs are predicted. This is somewhat expected since our models had no mechanisms to enforce the constraint that the predictions form a valid configuration.

1.7 Conclusions and recommendations

In this workshop, we were able to create statistical models that predict the per-leg ULD usage with reasonable accuracy, for the three main ULD types. Our models encapsulate the seasonality of cargo shipping as well as the lack of symmetry in the shipping volumes at airports. A surprise was that the models worked especially well for passenger legs (as opposed to pure-cargo legs), which were thought to be harder to predict.

This work is a proof of concept that needs to be refined before it is used in practice. In particular, we will need to be able to perform predictions on all ULD types. Given their sparse and sporadic usage, making predictions for those ULD types may be more challenging.

In addition, we may consider the following extensions to this project:

- **Considering additional data features.** Due to time constraints, several potentially interesting features were excluded, such as the day of the week the flight was on, or the general region (North America, Europe, Asia, etc.) of the flight.

- **Improving the predictor.** We may attempt to design a better predictor, perhaps by considering machine learning models. Indeed, ML models have the ability to combine features in complex ways, which may help improve the predictions. Some ML models can also do *multi-label* predictions, which would allow us to make predictions for all ULDs simultaneously.
- **Consider more complex ULD configurations.** This would enable us to assess how realistic our predictions are. It may also be possible to include ULD configuration rules in the prediction model to improve its accuracy.
- **Consider hybrid method.** Our analysis, as detailed in the above tables, indicates that we achieved commendable results using the XGBoost method for certain aspects and the Zero-Inflated Poisson model for others. A potential strategy for enhancing overall accuracy could involve combining these two methodologies. Specifically, we can integrate the outcomes from the Zero-Inflated Poisson model as an additional feature within the XGBoost framework. This approach aims to capitalize on the unique strengths of both models, potentially revealing more intricate patterns in the data and leading to a more robust and accurate predictive model.

Bibliography

[1] Adebayo, S. (2020). How the kaggle winners algorithm xgboost algorithm works. https://dataaspirant.com/xgboost-algorithm/.

[2] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 785–794. ACM.

[3] Lambert, D. (1992). Zero-Inflated Poisson Regression, with an Application to Defects in Manufacturing. Technometrics, 34(1):1–14.

[4] Tang, C.-H. and Yen, C.-Y. (2019). Airline unit load device dispatching considering service level and violation days. Journal of Air Transport Management, 79:101685.

2 Beneva: Competing risks analyses for return to work

Juli Atherton [a]
Sadoune Ait Kaci Azzou [b]
Carmelle Chango [a]
Houyam Dehbi [b]
Sorana Froda [a]
Lajmi Lakhal [c]
Janosch Ortmann [a,d]
Edna Paola Gutiérrez Vega [e]

[a] *UQAM*
[b] *BENEVA*
[c] *Université Laval*
[d] *GERAD*
[e] *Centro de Investigación en Matemáticas (CIMAT)*

March 2024
Les Cahiers du GERAD

2.1 Introduction

We analyze the amount of time that a client is disabled and receives disability benefits from the insurer Beneva. The event of interest is "return to work". There are other reasons, however, for the completion of a file at Beneva. In total, there are seven such reasons (competing risks): "return to work", "death", "lump sum payment", "cancellation", "retirement", "termination," and "other." The primary goal was to identify the profile of subjects who return to work faster and hence prioritize the processing of disability cases. In order to attain this goal different types of competing risk analyses were performed.

Two references for competing risks analysis using packages in R are [1] and [11]. Another reference for competing risks is [3]. Competing risks are a type of multi-state model; hence, references such as [2] on multi-state models are appropriate. Throughout the report we assume a basic knowledge of survival analysis such as found in the books [10] and [9]. Other more specific books, articles, and R packages will be introduced as needed throughout the report.

We begin by presenting data from Beneva, Section 6.2. In Section 2.3 we briefly introduce different approaches used in competing risks analysis. Section 2.4 presents non-parametric estimates used for a preliminary analysis and the results of two different approaches for competing risks analysis. Discussion for future work and conclusions are provided in Section 2.5.

2.2 Data set and descriptive statistics

The data source was internal data from Beneva having an experience period of 5 years. The population all have long term disabilities with an incurred date between 2018-01-01 and 2022-12-31 inclusively. No conditions on the end of disability date, the age, the type of insurance contract, or the location were applied. The variables in the data set and summary statistics are provided in Table 2.1.

Table 2.1: Description of variables and summary statistics for the whole data set. Summary statistics are calculated ignoring any missing data for that variable. When a proportion is approximated as 0% the number of individuals is provided.

Variable	Description and summary statistics
ID	Unique identifier (for a unique record) based on certificate number and disability file number
Gender	Gender of the insured: M(21.6%), F(78.3%)
Accident	Indicates if the disability is the result of an accident: Y(0.87%), N(99.1%)
ITP	Indicates if the disability was stated total and permanent: Y(0.82%), N(99.17%)
Revenue	Indicates if the disabled receives disability benefits from other sources than the insurer: Y(3%), N(96.9%)
Electronic	Indicates if the participant has an electronic statement subscription: Y(41.4%), N(58.4%)
Language	Language used by the participant: E(7.7%), F(92.28%), 9 (unspecified)(4; 0%)
Income	Gross monthly income at the beginning of the disability: mean(4530.603$), median(3909$), min(0$), max(833338$), sd(3301.257$)
Coverage	'other' indicates that Beneva provides administrative services and does not pay any claim: other(88%), FG(overhead)(104; 0%), LD(long-term)(11.95%)
Code	Illness that causes the disability if there is any: 1(0.52%), 2(0.055%), 3(0.22%), 4(0.32%), 5(0.11%), 6(0.58%), 7(0.08%), 8(0.38%), 9(0.97%), 10(0.001%), 11(4.01%), 12(0.225%), 13(5.25%), 14(0.042%), 15(0.13%), 99(86.97%)
Union	Marital status of the individual: other (76.22%), single(15.96%), common-law(1.92%), divorced(0.65%), married(4.9%), separated(0.22%), widowed(0.07%)
Employment	Job class according to activity level: unknown(84.6%), light(4.9%), heavy(3.5%), sedentary(9.9%)
Total	Total claims paid in dollars up to evaluation date: mean(154.46$), median(0$), min(0$), max(523800$), sd(3614.83$)

Table 2.1 – continued from previous page

Variable	Description and summary statistics
TIME	Duration (days) from the incurred date of disability to the evaluation date if the insured is still disabled at the evaluation date or to the disability end date if the insured is no more disabled at the evaluation date. Means: return to work(167.12), censored(664.90), death(516.34), payment(831.32), cancellation(780.78), retirement(440.70), termination(699.78), other(447.56)
RISKS	Standardized reasons why a disability would end: return to work(85.77%), censored(9.9%), death(0.58%), payment(61, 0%), cancellation(1.6%), retirement(0.32%), termination(0.92%), other(0.74%)
POSTAL	Indicates the geographic location (forward sortation areas (FSA)): 1221 categories
AGE	Group of age at the beginning of the disability: 55+ (18.44%), [47,55[(21.3%), [40,47[(19%), [32,40[(20.9%), [15,32[(20.1%)
RECIDIVE	Indicates the number of times the same insured was disabled for what is considered as the same disability (depending on the insurance contract): 0(99.47%), 1(0.47%), 2+(0.05%)

Referring to Table 2.1 we notice that about 10% of disability times for the record IDs are right-censored (due to the experience period ending). About 86% of people have experienced a return to work (see variable RISKS). It is also noticeable that the mean disability times are different for the different event types with "return to work" providing the shortest mean (see variable TIME). There are more women than men in the data set (see variable GENDER). Less than one percent had a disability that was the result of an accident (see variable ACCIDENT) and less than one percent had a disability that was total and permanent (see variable ITP). Only 3% received benefits from somebody other than the insurer (see variable REVENUE). Slightly less than half of the people used the electronic statement subscription (see variable ELECTRONIC). Less than 8% of individuals were anglophone (see variable LANGUAGE). Monthly salaries ranged from 0$ to 833, 338$ (see variable INCOME). About 88% of individuals are covered elsewhere (see variable COVERAGE). There is a fairly even distribution of ages in the data set (see variable AGE). Almost all individuals have had one disability occurrence for the given disability (see variable RECIDIVE). The variable UNION has 76.22% "other" and the variable EMPLOYMENT has 84.6% unknown. Concerning the total claims paid, 99.65% have an amount of 0$ (see variable AMOUNT). Since there are 1221 FSA, in Table 2.2 we simplify the variable POSTAL by creating a summary variable LETTER.

Table 2.2: Creation of variable LETTER to simplify the 1221 postal zones for variable POSTAL.

Variable	Description and Summary Statistics
LETTER	First letter in the postal code ('2'(1), '9'(2), 'A'(41), 'B'(84), 'C'(9), 'E'(134), 'G'(57546), 'H'(42454), 'J'(89120), 'K'(529), 'L'(696), 'M'(182), 'N'(398), 'P'(195), 'R'(159), 'S'(68), 'T'(581), 'V'(552), 'X'(3), 'Y'(12))

For reasons that will be addressed in the sequel, the data is treated differently for each analysis. Therefore separate discussions of data treatment are found in Sections 2.4.1, 2.4.2, and 2.4.3.

2.3 Theory

Our competing risks analysis aims to model the distribution of the random variable "time to complete a file", T, as a function of vector covariate values $\vec{z}$. The different reasons for completion of a file such as "return to work", "death", etc., are represented by D.

Considering the data description in Section 6.2, we see that there is no left truncation (see [9]) in the disability times and there is Type 1 independent right-censoring due to the end of the experience period.

The data (see Table 2.1) corresponds to observations $(t_i, d_i, \vec{z}_i)$ for individuals $i = 1, \ldots, n$. The different causes for the event are numbered from 1 to K and 0 indicates that an observation was right-censored. Therefore, we have $D_i \in \{0, 1, 2, \ldots, k, \ldots, K\}$ for each individual i. The number of covariates is p, so that $\vec{z}_i = (z_{i1}, \ldots, z_{ij}, \ldots, z_{ip})$ is the vector of covariate values for individual i. In the sequel, the subscript i denotes a record ID and the subscript j refers to a particular covariate. When appropriate, we suppress the subscripts.

2.3.1 Cause specific model

Generally, the following quantities are estimated in a competing risks analysis. This approach follows the (inhomogenous Markov) multistate model setting and is often referred to as the cause specific model.

- *Cause-specific intensity (hazard) function for cause k*

$$\lambda_k(t|\vec{z}) = \lim_{dt \to 0} \frac{P(t \leq T < t + dt, D = k | T \geq t, \vec{z})}{dt}\,; \tag{2.1}$$

- *Cumulative intensity (hazard) function for cause k*

$$\Lambda_k(t|\vec{z}) = \int_0^t \lambda_k(u|\vec{z})du\,; \tag{2.2}$$

- *Overall survival function*

$$S(t|\vec{z}) = \exp\left(-\sum_{k=1}^{K} \Lambda_k(t|\vec{z})\right) = P(T > t|\vec{z})\,; \tag{2.3}$$

- *Cumulative incident function (CIF) for cause k*

$$F_k(t|\vec{z}) = P(T \leq t, D = k|\vec{z}) = \int_0^t \lambda_k(u|\vec{z})S(u|\vec{z})du\,; \tag{2.4}$$

- *Cox regression model for cause k*

$$\lambda_k(t|\vec{z}) = \lambda_0(t)\exp(\vec{\beta}\vec{z})\,. \tag{2.5}$$

Note that when t tends to infinity the curve $F_k(t|\vec{z})$ tends to the value $P(D = k|\vec{z})$ which indicates the proportion of people with covariate vector value $\vec{z}$ who will experience risk k.

2.3.2 Subdistribution model

The Fine and Gray model [4] is often cited in the literature and for the sake of completeness we present this model. The purpose of this approach was to facilitate the calculation of the overall survival function and the cumulative incidence function, since as indicated in Expression (2.7), $F_k(t|\vec{z})$ can be calculated from a single subdistribution hazard function for cause k (see Expression (2.6)). We choose not to present our results for this model since we take the same view as [13] which favours the approach in Section 2.3.1.

- *Subdistribution hazard function for cause k*

$$\tilde{\lambda}_k(t|\vec{z}) = \lim_{dt \to 0} \frac{P(t \leq T < t + dt, D = k | \{T \geq t\} \cup \{T \leq t \cap D \neq k\}, \vec{z})}{dt}\,; \tag{2.6}$$

- *Cumulative incidence function (CIF) for cause k*

$$F_k(t|\vec{z}) = 1 - \exp\left(-\int_0^t \tilde{\lambda}_k(u|\vec{z})du\right)\,; \tag{2.7}$$

- *Cox regression model for cause k*

$$\tilde{\lambda}_k(t|\vec{z}) = \tilde{\lambda}_0(t)\exp(\vec{\beta}\vec{z})\,. \tag{2.8}$$

2.3.3 Random forests

Random forests for survival analysis are ensemble tree methods and are described in [7]. Subsequently, random forests for competing risks were presented in [5]. The methodology was developed for right-censored data, which corresponds to our observations.

A random forest of B trees is generated by drawing B bootstrap samples from the original data. Each sample is used to grow a binary tree. A splitting rule is used to determine each binary split in the tree. Different covariates $(z_j,\ j = 1,\ldots,p)$ and covariate values are used to create binary partitions of the data and the partition that in some sense "maximizes" the "difference" in survival times (according to the chosen splitting rule) is selected. The leaves in each tree are composed of homogeneous individuals. For example, a proposed split based on a continuous covariate z_j would be of the form $\{z_j \leq c\}$ and $\{z_j > c\}$ for a constant c.

Each covariate vector value $\vec{z} = (z_1,\ldots,z_j,\ldots,z_p)$ identifies with a leaf in each tree. An "in bag" estimate for a covariate vector value $\vec{z}$ is created by first identifying all the leaves in the forest corresponding to $\vec{z}$ and second, depending on the quantity one wishes to estimate, using data from the leaves corresponding to $\vec{z}$ to estimate non-parametrically Equations (2.1), (2.2), (2.3), or (2.4). The final "in bag" estimate is an average of the estimates for all the leaves in the trees corresponding to the covariate vector value $\vec{z}$.

Similarly, consider a record ID i with covariate vector value $\vec{z}_i$. For trees where case i was not selected in the bootstrap sample to build the tree, case i corresponds to "out of bag" data and meanwhile the covariate vector value $\vec{z}_i$ will correspond to a leaf in the tree. An "out of bag" estimator for a case i is an average of the "in bag" estimates of the leaves corresponding to $\vec{z}_i$ over trees for which case i was "out of bag" data. Again, depending on the estimate in question, non-parametric estimates of Equations (2.1), (2.2), (2.3), or (2.4) are used.

2.4 Results

In this section we first present non-parametric estimates for the CIF curves by different covariate values in Section 2.4.1. Next we present the Cox regression results in Section 2.4.2. The random forest results are presented in Section 2.4.3.

In each case the data treatment is slightly different. The treatment of data is described in Sections 2.4.1, 2.4.2, and 2.4.3. Since our main interest is in the competing risk "return to work" and there was no reason to model common parameters for the "return to work" cause and the other causes, we have combined the other causes into a single cause labelled "other". The parameters for "return to work" are estimated separately from the parameters for "other". Combining all the other causes into a single cause also simplifies the presentation of many of the figures.

2.4.1 Preliminary analysis

We first describe the data treatment in Section 2.4.1 and then present the results in Section 2.4.1.

Data treatment for preliminary analysis

The original data set contains 200,479 record IDs. Once lines with missing values in any covariate are removed the data set contains 192,766 record IDs. Of these 17,741 have censored observations.

For the CIF curves presented in the figures indicated below, we

- (for all figures) combined all competing risks except "return to work" into a risk called "other";
- (for Figure 2.2c) removed the four record IDs in category "9" for LANGUAGE;

- (for Figure 2.4b) created the variable LETTER (see Table 2.2) and used the seven categories with the most record IDs;
- (for Figure 2.3b) used the medical codes which have more than 1% of record IDs. These are the codes "11", "13", and "99" for CODE;
- (for Figure 2.3c) eliminated "other" and combined "married" and "common-law", "divorced" and "separated", and "single" and "widowed" for UNION;
- (for Figure 2.3d) eliminated the category "unknown" from EMPLOYMENT.

Results for preliminary analysis

Since our main interest is "return to work", a naïve approach is to view end of disability due to all other causes as right-censored observations and to consider the Kaplan-Meier curves for different covariate values. These estimates will be incorrect, see [13] for details. Therefore we consider non-parametric estimates of the cumulative incidence functions (CIF) for different covariate values. The reference [12] provides a nice discussion as to why the CIF curves are favoured over the survivor functions in a competing risks analysis.

In Figure 2.1a, which displays the CIF curves for all competing risks in the data set, we see that by far the highest CIF curve is for return to work. This implies that most record IDs terminate at Beneva due to a "return to work".

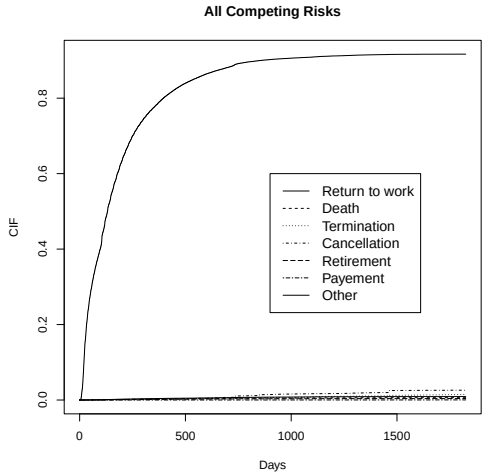
Since a main interest is identifying the covariates affecting return to work, the remaining figures (Figures 2.1b to 2.4d) present the CIF curves for "return to work" (with all other causes collected together into "other"). Different covariate values presented in Table 2.1 (or summaries of certain covariate values) are used for Figures 2.1b to 2.4d. Both the creation of a secondary cause by combining all other causes and the simplification of certain covariates were carried out to unclutter the figures thereby making the CIF curves more visible.

Using Figure 2.1b for the covariate GENDER as an example, the black curves are $P(T \leq t, D = \text{return to work} | Z_j = \text{M})$ and $P(T \leq t, D = \text{other} | Z_j = \text{M})$. The sum of the black curves is $P(T \leq t | Z_j = \text{M})$ and as t increases the sum of the black curves tends to one. Similar comments apply to the red curves for the females $\{Z_j = F\}$. When the curves for different covariate values are not similar, we may infer that the "return to work" CIF curves vary as a function of the covariate values.

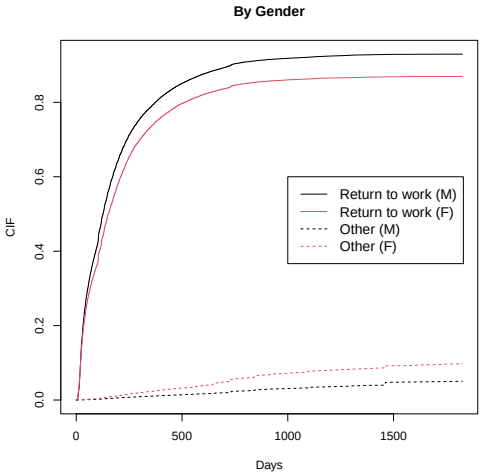
There are small differences in the CIF curves for the covariates GENDER, ELECTRONIC, INCOME, and UNION, moderate differences for ACCIDENT, REVENUE, EMPLOYMENT, AGE, and LANGUAGE, and larger differences for ITP, LETTER, COVERAGE, CODE, AMOUNT, and RECIDIVE. It is not clear that these differences will be significant since certain categories contain small numbers of individuals (see for example ACCIDENT in Table 2.1). Recall, as mentioned in Section 2.4.1, that to generate certain figures not all the data is used.

2.4.2 Cause specific intensity model

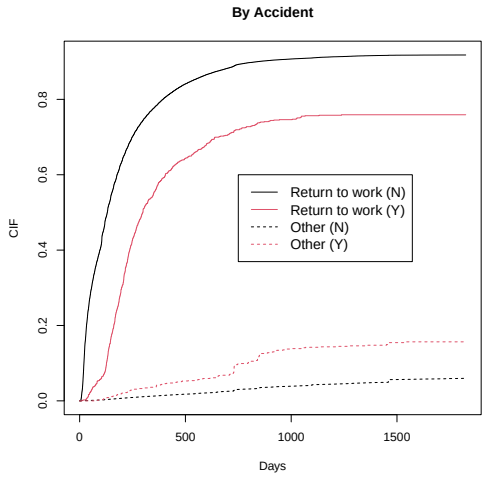
We first describe the data treatment in Section 2.4.2 and then present the results in Section 2.4.2.



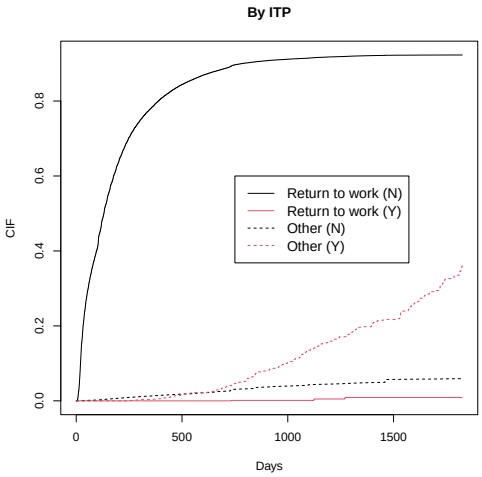
(a) The highest CIF curve is for the competing risk 'return to work'.



(b) Men tend to return to work faster than women.



(c) Those with no accident tend to return to work faster.



(d) Those without total and permanent disability tend to return to work much more quickly.

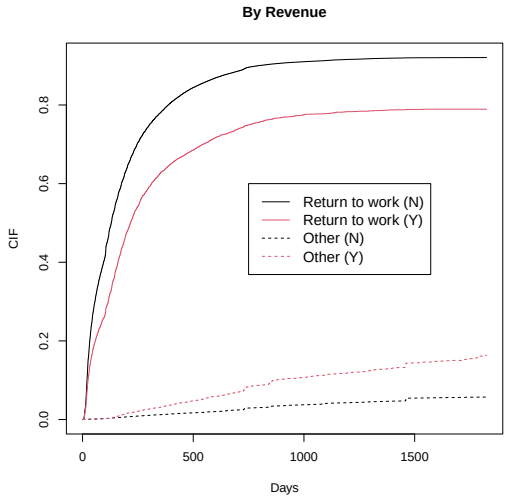
Figure 2.1: Non-parametric estimates: Comparison of CIF curves for all competing risks (a). Comparison of CIF curves 'return to work' and 'other' by Gender **(b),** Accident **(c) and** ITP **(d).**

Data treatment for cause specific intensity model

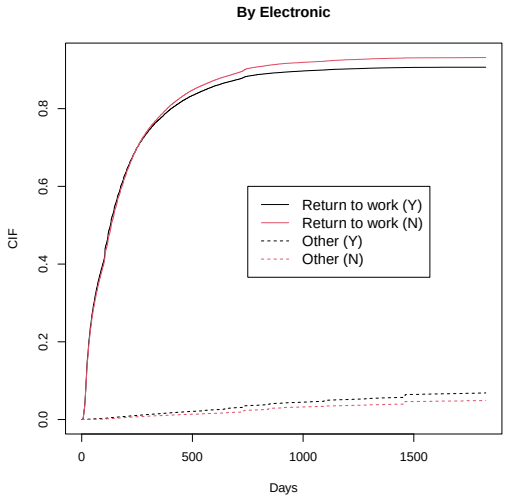
To simplify the presentation, we run the model on a large subset of the data. Firstly, for certain variables with a large number of categories we either combine categories (as for example with Letter) or we eliminate categories with a small number of individuals (as for example with Language). Secondly, we do not use certain variables having a large number of individuals with "other" or "unknown" (as for example with Union). The data treatment is as described in Section 2.4.1 with the modifications described below. Specifically, we

- kept all the postal code letters for variable Letter except "2", "9","C", "X", and "Y" (see Table 2.2);
- did not model the variables Coverage, Union, and Employment due to the large "other" and "unknown" categories.

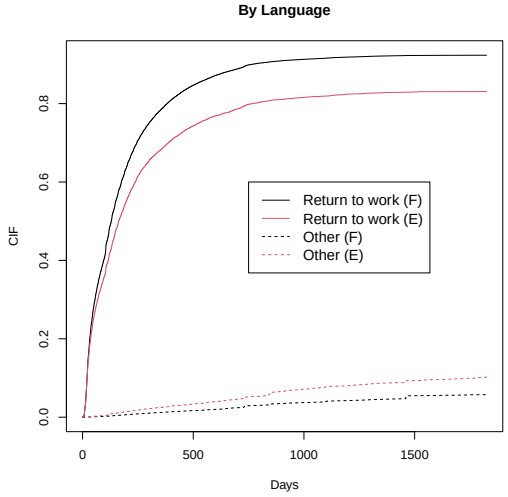
These changes result in 187,846 record IDs.



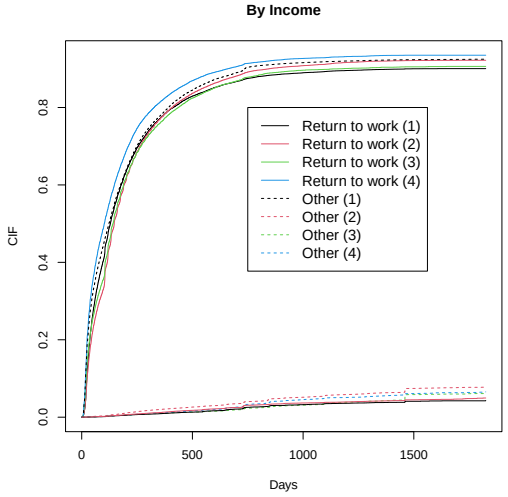
(a) Those who did not receive disability benefits from other sources tended to return to work more quickly.



(b) There was only a slight difference in the CIF curves for those who used electronic subscriptions versus those that do not.



(c) Francophones return to work faster than anglophones.



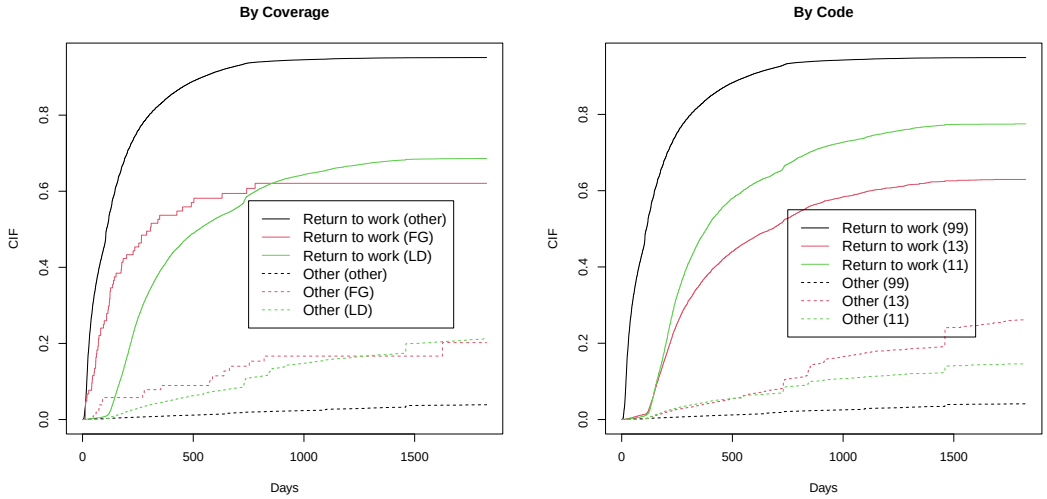
(d) Each category is a quartile of the salaries grouped together. There does not seem to be a big difference in the return to work as a function of salary.

Figure 2.2: Non-parametric estimates: Comparison of CIF curves by four covariates Revenue **(a),** Electronic **(b),** Language **(c) and** Income **(d).**

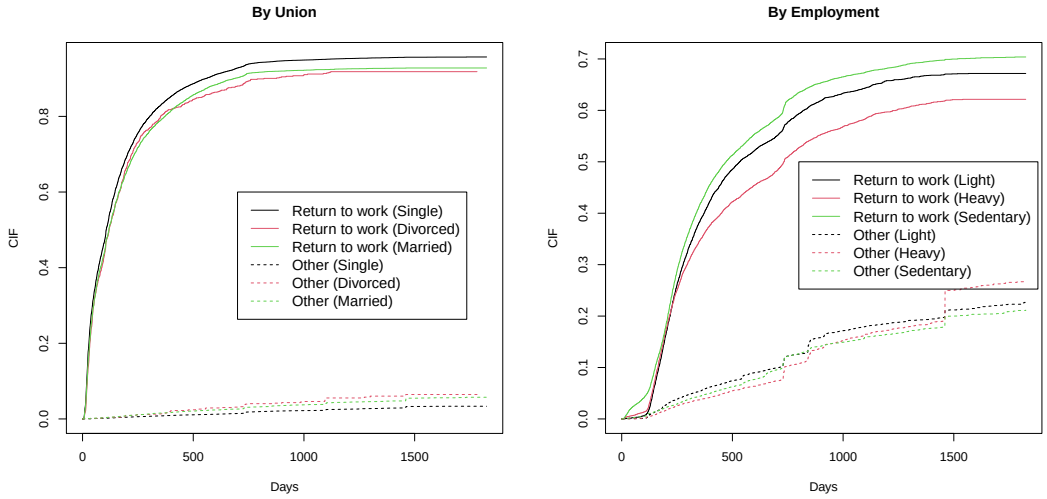
Results for cause specific intensity model

The transition intensity (2.1) is modeled using a Cox proportional hazards formulation (2.5). Considering a binary covariate Z_j taking the values 0 and 1, this means that the intensities for return to work for the groups $\{Z_j = 0\}$ and $\{Z_j = 1\}$ should be proportional at every value of t while holding all other covariates constant. Since we do not see many of the CIF curves for "return to work" crossing or even touching each other in Figures 2.1b to 2.4d, the Cox model is a plausible model for this data set.

Returning to the binary data example, if the group $\{Z_j = 0\}$ were used to form the baseline intensity function then the relative risk would be calculated by $\exp(\beta)$. This interpretation is presented in standard textbooks on survival analysis such as [10]. If the relative risk is 1 this implies that both groups are equivalent in terms of how fast they return to work, if the relative risk is less than one it



(a) People who have 'other' coverage return to work more quickly. **(b) The CIF curves vary as a function of the type of illness.**



(c) There is a slight difference depending on the type of union. **(d) 'Sedentary' returns more quickly and 'Light' returns the second fastest.**

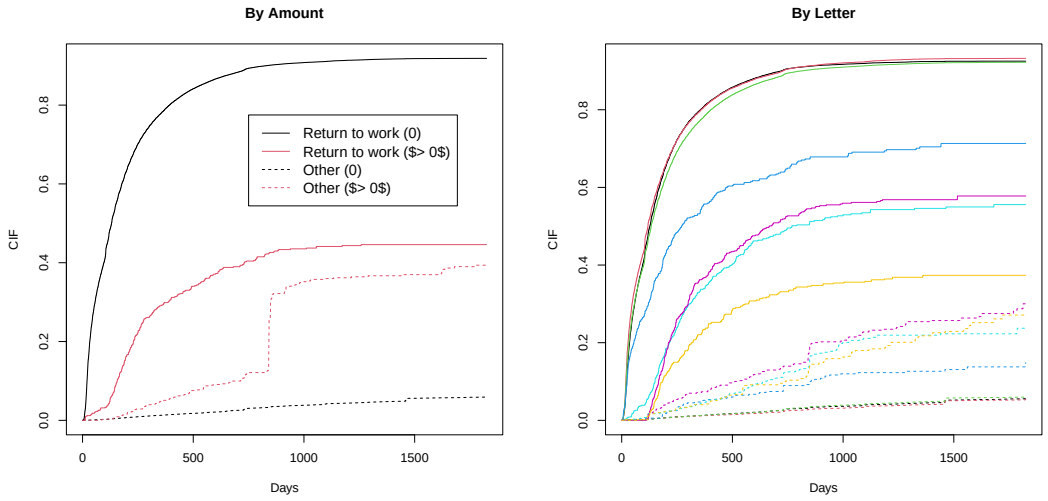
Figure 2.3: Non-parametric estimates: Comparison of CIF curves by four covariates Coverage **(a),** Code **(b),** Union **(c) and** Employment **(d).**

means the group $\{Z_j = 0\}$ is at a higher risk of returning to work (returns faster), and finally if the relative risk is greater than one this means the group $\{Z_j = 1\}$ is at higher risk of returning to work more quickly.

The intensity function for return to work can be estimated as a function of the covariates using the function `coxph` in R. As described in [13], one can allow different coefficient vectors $\vec{\beta}$ and different baseline intensity functions for all the competing risk intensity functions or force them to be the same for certain competing risks. Since there is no indication in Figures 2.1b to 2.4d and in other analyses (not shown) that such constraints are necessary, we model the return to work intensity function with its own baseline intensity function and its own parameter $\vec{\beta}$.

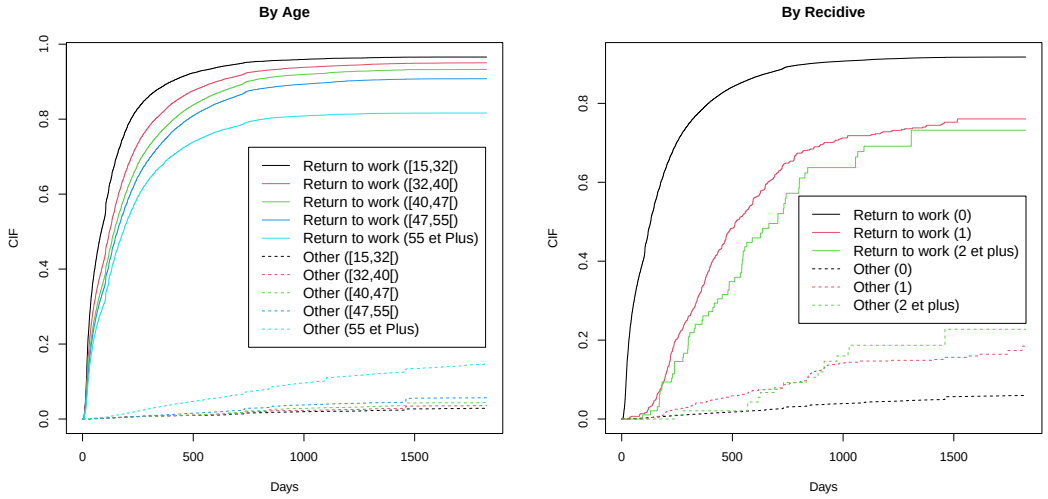
The results of the Cox model are shown in Table 2.3. Even though this is a large sub-sample of the main data set and we have left out three covariates, only a couple of trends that we noticed in the

Figures 2.1b to 2.4d are reversed. Note that ACCIDENT is significant and LANGUAGE is not significant. The behaviour of the other variables is consistent with Figures 2.1b to 2.4d.



(a) The CIF curves vary in function of an amount equal to 0 and an amount greater than 0.

(b) The CIF curves vary by first letter of postal code (region): 'G'=black, 'H'=red, 'J'=green, 'K'=blue, 'L'=light blue, 'T'=pink, 'V'=yellow.



(c) The CIF curves vary by age groups.

(d) The CIF curves vary by recidives.

Figure 2.4: Non-parametric estimates: Comparison of CIF curves by four covariates AMOUNT (a), LETTER (b), AGE (c) and RECIDIVE (d).

2.4.3 Random Forest analysis

We first describe the data treatment in Section 2.4.3 and then present the results in Section 2.4.3.

Data Treatment for Random Forest Analysis

We use the data treatment described in Section 2.4.2 and then we take a simple random sample of size 30,000 without replacement. We do this because the package `randomForestSRC` (see [8]) is limited in terms of the amount of data it can handle. Such a sample will have a good representation of record IDs who "return to work" and who are "censored". Typically, a sample of this size might not select any (or

very few) record IDs from all the other causes but since we combine the other causes into a common cause "other" and model the "return to work" cause independently, this creates no problems for the analysis.

Table 2.3: Results of the Cox model for the intensity function for the cause 'return to work' using the data treatment presented in Section 2.4.2.

Variable category	Baseline category	Relative risk	p-value
Gender (F)	Gender (M)	0.950	0.0005(<)
Accident (Y)	Accident (N)	1.470	0.0005(<)
ITP (Y)	ITP (N)	0.008	0.0005(<)
Revenue (Y)	Revenue (N)	0.912	0.0005(<)
Electronic (Y)	Electronic (N)	1.056	0.0005(<)
Language (E)	Language (F)	1.002	0.878
Income		1	0.0005(<)
Code (11)	Code (99)	0.302	0.0005(<)
Code (13)	Code (99)	0.391	0.0005(<)
Amount		1	0.0005(<)
Letter (A)	Letter (G)	0.329	0.002
Letter (B)	Letter (G)	0.683	0.030
Letter (E)	Letter (G)	1.048	0.714
Letter (H)	Letter (G)	1.014	0.053
Letter (J)	Letter (G)	0.952	0.0005(<)
Letter (L)	Letter (G)	0.696	0.0005(<)
Letter (M)	Letter (G)	0.556	0.0005(<)
Letter (N)	Letter (G)	0.777	0.007
Letter (K)	Letter (G)	0.987	0.827
Letter (P)	Letter (G)	0.982	0.866
Letter (R)	Letter (G)	0.646	0.011
Letter (S)	Letter (G)	0.467	0.017
Letter (T)	Letter (G)	0.868	0.016
Letter (V)	Letter (G)	0.607	0.0005(<)
Age ($[32, 40[$)	Age ($[15, 32[$)	0.783	0.0005(<)
Age ($[40, 47[$)	Age ($[15, 32[$)	0.688	0.0005(<)
Age ($[47, 55[$)	Age ($[15, 32[$)	0.646	0.0005(<)
Age (55 plus)	Age ($[15, 32[$)	0.570	0.0005(<)
Recidive (1)	Recidive (0)	0.886	0.0005(<)
Redicive (2 plus)	Redicive (0)	0.669	0.00586

Results for Random Forest Analysis

The description of the forest grown is provided in Table 2.4. In the R function `rfsrc`, we used the composite (weighted) splitting rule with the cause set to "return to work" and the generalized log-rank test selected as the splitting rule.

Table 2.4: Results for the random forest analysis for the cause 'return to work' using the data treatment presented in Section 2.4.3.

Sample size	29997
Number of events	26331, 1017
Number of trees	500
Forest terminal node size	15
Average no. of terminal nodes	1471.334
No. of variables tried at each split	4
Total no. of variables	12
Resampling used to grow trees	swor
Resample size used to grow trees	18958
Analysis	RSF
Family	surv-CR
Splitting rule	logrank *random*
Number of random split points	10
(OOB) Requested performance error	0.3910744, 0.41358587

Figures 2.5b to 2.7d plot the individual estimated values for the CIF curves evaluated at 1000 days (probability of return to work before 1000 days) by covariate values. In general, we see the same trends observed in Figures 2.5b to 2.7d. There might be a slight difference with GENDER and LANGUAGE.

Table 2.5 displays the minimal depth (MD) and variable importance (VIMP) for the covariates used in the forest. Generally the covariates that are higher up in the list (i.e., are higher in importance) correspond with the covariates that have lower p-values in the Cox regression model.

Table 2.5: Minimum depth (MD) and variable importance (VIMP) for the competing risk cause 'return to work' as described in [6].

Variable	MD	VIMP
CODE	1.2	10.78
AGE	1.4	5.23
INCOME	1.9	6.21
ITP	2.0	2.93
LETTER	2.5	3.29
REVENUE	3.4	1.29
GENDER	3.6	0.99
AMOUNT	3.7	2.14
LANGUAGE	3.8	0.21
ELECTRONIC	4.0	0.11
ACCIDENT	4.2	1.01
RECIDIVE	4.4	3.12

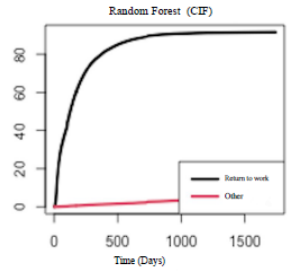
2.5 Concluding remarks

In general we found that there is good agreement between the three approaches and this was despite the need to use a smaller sample of data for the random forest approach. The agreement between the cause specific Cox model and the random forest analysis is both in terms of the significance in the Cox model (see p-values in Table 2.3) and the importance in the random forest analysis (see Table 2.5) and in terms of the behaviour of the covariates (see the relative risks in Table 2.3 and Figures 2.5b to 2.7d).

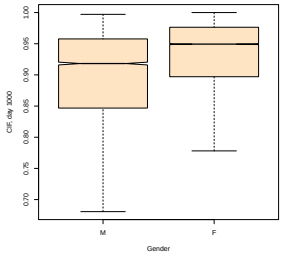
We make the following comments:

- Concerning the data set itself, it would be more informative to obtain the actual values instead of "other" for the variable UNION and instead of "unknown" for the variable EMPLOYMENT. These categories comprise 76.22% and 84.6% of the data respectively;
- One should consider confounding in the covariates and begin looking for associations among the covariates;
- There is a R package called `largeRDRF` [1] (see [14]) that can run a random forest analysis on larger data sets for competing risks, however it does not contain many functions to extract information from the forest;
- Comparing the performance of the models using Brier score or another method should be done. Also one should verify the assumption of proportional hazards for the Cox model in more detail;
- It would also be interesting to consider the effect of covariates on other competing risks and not to lump the six other causes together. One problem is that some of these competing risks have such a small number of record IDs that the simple random sampling we used for the random forest could be problematic. It might be interesting to consider some sort of stratified sampling (see for example [15]).

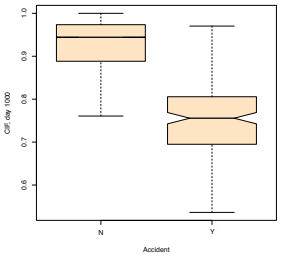
[1] Java JDK is required.



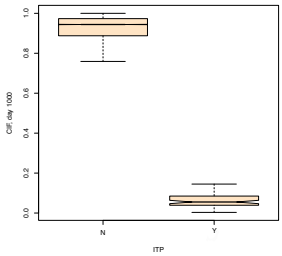
(a) The averaged CIF curves for 'return to work' and 'other'.



(b) Very slight trend that a higher proportion of females tended to return to work before 1000 days than males.

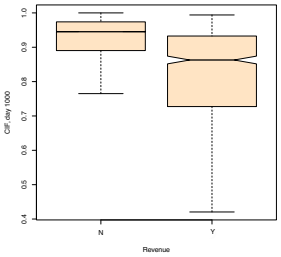


(c) A higher proportion of people whose disability was not the result of an accident tended to return to work before 1000 days than those whose disability was the result of an accident.

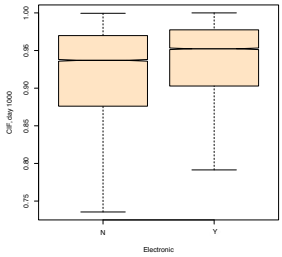


(d) A much higher proportion of people whose disability was not total and permanent tended to return to work before 1000 days than those whose disability was total and permanent.

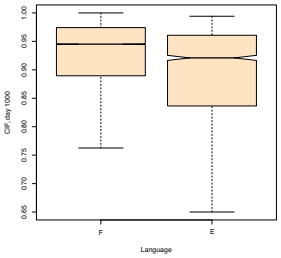
Figure 2.5: Results of random forest analysis: Averaged 'out of bag' estimates of CIF curves (a) and boxplots of individual CIF curves evaluated at 1000 days by GENDER (b), ACCIDENT (c) and ITP (d).



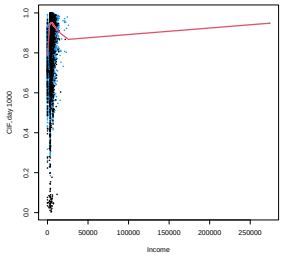
(a) Slight trend that a higher proportion of people receiving disability benefits from Beneva return to work before 1000 days than those that receive disability benefits elsewhere.



(b) Very slight trend that a higher proportion of people using the electronic form return to work before 1000 days than those that do not use the electronic form.

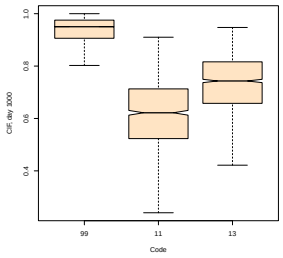


(c) Very slight trend that a higher proportion of francophones return to work before 1000 days than anglophones.

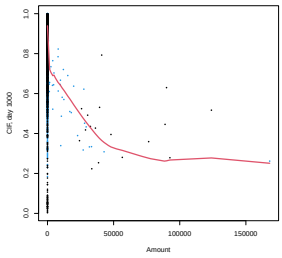


(d) The proportion of people returning to work before 1000 days seems to increase and then decrease as a function of income.

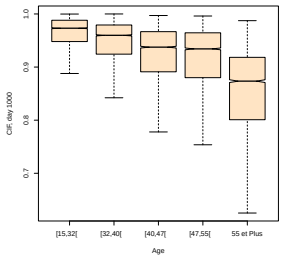
Figure 2.6: Results of random forest analysis: Boxplots of individual CIF curves evaluated at 1000 days by REVENUE (a), ELECTRONIC (b), LANGUAGE (c) and INCOME (d).



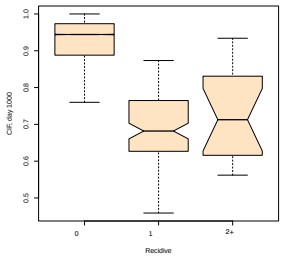
(a) General trend that a higher proportion of people who return to work before 1000 days have Code '99'. Similarly, a lower proportion of people who return to work before 1000 days have Code '11'.



(b) A higher proportion of people who return to work before 1000 days have a total amount paid of 0$. This proportion decreases as a function of the total paid amount.

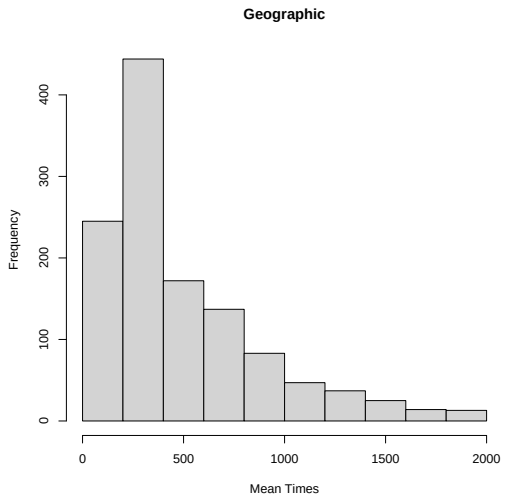


(c) Generally, the younger the individual, the higher the probability that they will return to work before 1000 days.

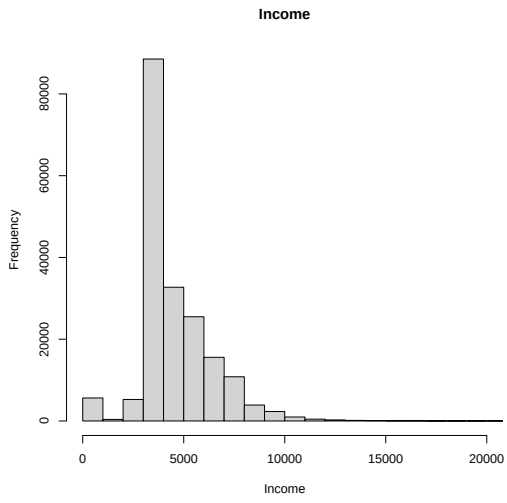


(d) More individuals with Recidive of 0 will return to work before 1000 days.

Figure 2.7: Results for the random forest analysis: Boxplots of individual CIF curves evaluated at 1000 days by Code (a), Amount (b), Age (c) and Recidive (d).



(a) The distribution of the mean disability time for each of the 1221 regions is right skewed.



(b) The distribution of incomes is slightly right skewed.

Figure 2.8: A histogram of the mean observed times (including censoring times) for the 1221 different FSA (a) and a histogram of the values for Income (b).

Bibliography

[1] Beyersmann, J., Allignol, A., and Schumacher, M. (2011). Competing risks and multistate models with R. Springer Science & Business Media.

[2] Cook, R. J. and Lawless, J. F. (2018). Multistate models for the analysis of life history data. CRC Press.

[3] Crowder, M. J. (2012). Multivariate survival analysis and competing risks. CRC Press.

[4] Fine, J. P. and Gray, R. J. (1999). A proportional hazards model for the subdistribution of a competing risk. Journal of the American statistical association, 94(446):496–509.

[5] Ishwaran, H., Gerds, T. A., Kogalur, U. B., Moore, R. D., Gange, S. J., and Lau, B. M. (2014). Random survival forests for competing risks. Biostatistics, 15(4):757–773.

[6] Ishwaran, H., Gerds, T. A., Lau, B. M., Lu, M., and Kogalur, U. B. (2021). randomForestSRC: competing risks vignette. http://randomforestsrc.org/articles/competing.html.

[7] Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. (2008). Random survival forests.

[8] Ishwaran, H., Kogalur, U. B., and Kogalur, M. U. B. (2023). Package 'randomforestsrc'. Breast, 6(1):854.

[9] Kalbfleisch, J. D. and Prentice, R. L. (2011). The statistical analysis of failure time data. John Wiley & Sons.

[10] Klein, J. P., Moeschberger, M. L., et al. (2003). Survival analysis: techniques for censored and truncated data, volume 1230. Springer.

[11] Pintilie, M. (2006). Competing risks: a practical perspective. John Wiley & Sons.

[12] Porta Bleda, N., Gómez Melis, G., and Calle Rosingana, M. L. (2008). The role of survival functions in competing risks.

[13] Therneau, T., Crowson, C., and Atkinson, E. (2020). Multi-state models and competing risks. CRAN-R (https://cran. r-project. org/web/packages/survival/vignettes/compete.pdf).

[14] Therrien, J. and Cao, J. (2022). Random competing risks forests for large data. arXiv preprint arXiv:2207.11590.

[15] Zhou, B., Latouche, A., Rocha, V., and Fine, J. (2011). Competing risks regression for stratified data. Biometrics, 67(2):661–670.

3 Desjardins: Privacy evaluation of synthetic data generation

Astou Ndime [a]
David Ayotte [b]
Emilio Villa Cueva [c]
Mohammad Ebrahimi [d]
Md Hassib Udin Molla [e]
Parisa Davar [b]
Vincent Racine [f]
Zaniar Ahmadi [b]
Sébastien Gambs [g]
Adel Benlagra [h]
Antoine Langevin [h]

[a] *African Institute for Mathematical Sciences, Sénégal*
[b] *Concordia University*
[c] *CIMAT, Guanajuato, Mexico*
[d] *HEC*
[e] *University of Calgary*
[f] *University of Waterloo*
[g] *UQAM*
[h] *Desjardins*

March 2024
Les Cahiers du GERAD

Abstract: *In this report we present our initial assessment of the TabDDPM model used as a synthetic data generator at Desjardins. In particular, we demonstrate that it is possible to discriminate between synthetic data generated by TabDDPM and real data used to train it. This allows us to obtain insights into which features TabDDPM is struggling to replicate. Additionally we provide evidence that TabDDPM is susceptible to attribute inference attacks. Consequently, despite its great potential, we advise Desjardins to conduct a more thorough assessment of the capabilities of TabDDPM before using it as a synthetic data generator.*

3.1 Introduction

The problem addressed in this workshop, proposed by Desjardins, revolves around synthetic data and its utilization to enhance data while safeguarding the confidentiality of the company's members and customers. TabDDPM [9], a novel AI methodology based on diffusion models [6, 15], can generate synthetic tabular data. It offers a promising approach to create synthetic data that is not linked to any Desjardins member or client but shares the same statistical properties as the confidential data.

Desjardins aims to leverage the synthetic data generated by TabDDPM to develop analytical models that are valuable without posing privacy risks. Generating data that respects the privacy of clients is of particular significance at Desjardins, especially in light of the increased scrutiny the company faces following a massive data breach in 2019.[1]

The primary objective of the problem studied in this workshop is to provide an initial assessment of the viability of TabDDPM as a synthetic data generator at Desjardins. To address this objective, two key concerns need to be explored.

- Is it possible to derive similar insights from the synthetic data generated by TabDDPM as it would from the real data?
- Does the synthetic data generated by TabDDPM pose any privacy risks?

More precisely, we have investigated the following two specific questions.

- Can we discriminate between the real data used to train the TabDDPM model and the synthetic data generated by the TabDDPM model? (referred to as Q1)
- Can we successfully conduct an attribute inference attack, a form of privacy attack, on TabDDPM? (referred to as Q2)

The remainder of this report is structured as follows. First, in Section 3.2, we provide the necessary background on synthetic data generation, discussing in particular the dual objectives of synthetic data: usefulness and privacy, and how they are typically measured in the literature. We also provide an in-depth description of the TabDDPM model. Afterwards in Section 3.3, we elaborate on Q1 and Q2 and outline our methodology to address them. Then in Section 3.4, we document the results of our initial assessment of the viability of TabDDPM as a synthetic data generator before concluding remarks in Section 3.5.

3.2 Background

This section provides the foundational context for our research project. More precisely in Subsection 3.2.1, we outline the objectives of synthetic data generation while in subsection 3.2.2, we elaborate on the TabDDPM model.

[1] https://www.cbc.ca/news/canada/montreal/desjardins-data-breach-lawsuit-settlement-1.6288428

3.2.1 Usefulness and privacy

As mentioned in the introduction, synthetic data generation within the context of the problem at Desjardins serves a dual purpose. First, the synthetic data should yield the same insights as the real data, and second, it must not be associated with any Desjardins members or clients. These objectives are referred to as "usefulness" and "privacy," respectively. These two goals can be viewed as competing since any improvement in privacy typically results in a reduction in usefulness, and vice versa, as illustrated in Figure 3.1.

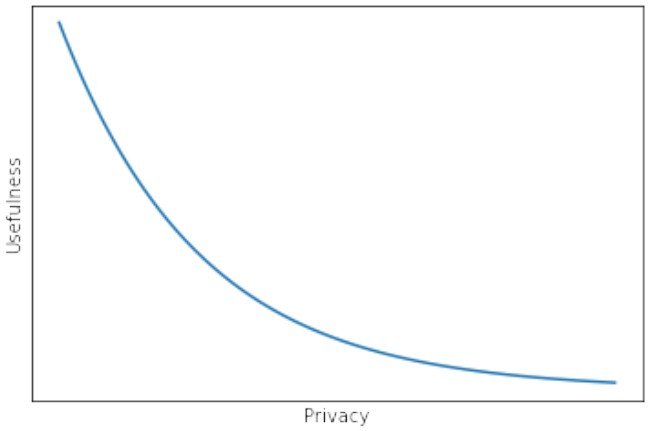


Figure 3.1: Visualization of the tradeoff between usefulness and privacy.

We refer to this tradeoff as the "usefulness-privacy tradeoff," and we conceptualize it in a framework similar to the mean-variance tradeoff in portfolio selection [11]. In this analogy, the maximum usefulness generating policy would be akin to simply replicating the real data, while the maximum privacy generating policy would involve generating purely random noise. As with portfolio selection, the objective is to identify a generating strategy that lies on the Pareto Front. In other words, for a fixed privacy threshold, we aim to discover the generating strategy that provides the highest level of usefulness.

Metrics for assessing usefulness can be broadly categorized into two groups: 1) fidelity metrics and 2) utility metrics. Fidelity metrics gauge the proximity of statistical properties in synthetic data to those in real data. Common fidelity metrics include exploratory statistical comparison (measuring the distance between mean, median, maximum, minimum, standard deviation, etc., in synthetic and real data), histogram similarity score (evaluating the similarity of marginal distributions between real and synthetic data), mutual information score (quantifying the similarity in the relationships between two features), and correlation score (assessing the degree to which correlations in the original dataset are replicated in the synthetic dataset).

Utility metrics assess how well synthetic data performs in common data science tasks. Common utility metrics include machine learning efficiency (determining whether a machine learning model trained on synthetic data can perform on par with a model trained on real data), feature importance score (assessing whether a machine learning model trained on synthetic data utilizes the same features as a model trained on real data), and Q-score (evaluating whether aggregation-based queries on synthetic data yield the same results as those on real data). We refer the reader to the following blog: https://aws.amazon.com/blogs/machine-learning/how-to-evaluate-the-quality-of-the-synthetic-data-measuring-from-the-perspective-of-fidelity-utility-and-privacy/ for more details on usefulness metrics.

In contrast, privacy is often measured by the success level of privacy attacks. More precisely, a generative model is considered private when an attacker is unsuccessful with respect to these privacy

attacks. There are three main types of privacy attacks: 1) membership inference attacks, 2) attribute inference attacks, and 3) reconstruction attacks [8].

In a membership inference attack, the attacker is provided with synthetic data generated by the generator and real data not used during the generator's training. When given a new real observation, the attacker's goal is to predict whether the observation was part of the training set for the generative model. A common heuristic in membership inference attacks is to measure the distance to the closest synthetic samples. If the example is close to the synthetic data, it is more likely to be have been used to trained the generative model since generative models tend to overfit their training data.

An attribute inference attack aims to deduce sensitive attributes from an incomplete target record of a real observation that was used in training the generative model, using the synthetic data. For example, in the context of Desjardins, an attacker may have access to synthetic data and a real observation used to train the generative model, with the client's gender (the sensitive attribute) removed. The attacker's objective is to infer the gender of the real observation.

Finally a reconstruction attack seeks to reconstruct the complete training set using synthetic data. This type of attack remains relatively underexplored in current research, resulting in limited existing attack strategies within the literature.

3.2.2 TabDDPM

TabDDPM [9] introduces an innovative approach for generating synthetic tabular datasets. The key novelty of TabDDPM lies in proposing preprocessing steps for tabular data that enable the application of denoising diffusion probabilistic models (DDPM) to this type of data. Specifically, the authors suggest transforming numerical features into a normal distribution and categorical features into one-hot vectors, which enables the use of a Gaussian diffusion model [6, 15] for numerical features and a multinomial diffusion model [7] for categorical features. In addition they recommend employing a multi-layer perceptron model with relatively shallow depth to model the reverse process, and the hyperparameters of TabDDPM are tuned using Optuna [1]. The objective function is machine learning efficiency, measured with Catboost [14] on a validation dataset. TabDDPM garnered attention at Desjardins due to its promising results in terms of both usefulness and privacy.

In their original paper, the authors have assessed usefulness using correlation scores and machine learning efficiency. More precisely, they have compared TabDDPM with four alternative data generation methodologies: TVAE [16], CTAB-GAN [17], CTAB-GAN+ [18], and SMOTE [4], on 16 datasets. Note that SMOTE was originally designed to address class imbalance but can be adapted to generate synthetic samples. The authors observed that TabDDPM generated synthetic data with more realistic pairwise correlations in most cases. Furthermore, TabDDPM achieved the highest machine learning performance on 10 out of 16 datasets when measured using the standard approach in the literature, which involves averaging results from multiple weak models. They also found that TabDDPM outperformed the other generative models on 13 of the 16 datasets when measuring machine learning efficiency using Catboost. The authors argue that the score obtained by Catboost is a good metric since Catboost generally excels on tabular datasets.

Privacy was assessed by measuring the distance to the closest real samples. As mentioned in Subsection 3.2.1, this is often a heuristic used to estimate the probability of success of a membership inference attack. TabDDPM generates samples with greater distances from its training dataset than SMOTE for all 16 datasets. Other generative models, however, were not compared with TabDDPM in terms of privacy, thus limiting the scope of the privacy evaluation.

We also note that TabDDPM was recently rejected for publication at ICLR 2023[2] but was later accepted for publication at ICML 2023. Some of the concerns raised by the reviewers were the lack of novelty in the paper and the absence of proper privacy attacks to evaluate whether TabDDPM is indeed

[2] https://openreview.net/forum?id=EJka_dVXEcr

private. One reviewer also raised concerns that by using a multinomial diffusion model to represent categorical features and a Gaussian diffusion model for numerical ones, TabDDPM might be unable to model relationships between numerical and categorical variables. The authors of the TabDDPM paper, however, contend that TabDDPM can indeed capture relationships between categorical and numerical features, and our initial assessment aligns with their claim.

3.3 Formulation and methodology

The goal of this workshop was to obtain a better understanding of TabDDPM and the viability of its use for generating synthetic data at Desjardins. To address this objective, Desjardins suggested considering two questions:

1. Discriminating between real data used for training TabDDPM and synthetic data generated by TabDDPM;
2. Conducting an attribute inference attack on TabDDPM.

We will refer to these as Q1 and Q2, respectively. In the remainder of this section we will formulate each question and describe our proposed methodology for answering them.

3.3.1 Q1

The problem statement for Q1 is as follows: we have a dataset that includes real examples used to train TabDDPM and synthetic examples generated by TabDDPM. The objective is to develop discriminative models that can distinguish between these two types of data. The dataset used is derived from the "Bank Marketing" dataset in the UCI public repository[3], augmented with five socio-economic variables. The training dataset comprises 76019 observations, with an even split between synthetic and real examples. The test dataset, which is also balanced, contains 19005 observations to be able to measure against a baseline that would be equivalent to a random guess. We will evaluate the performance of our discriminative model using the Area Under the Receiver Operating Characteristic Curve (AUC).

Intuitively, we can consider the AUC obtained by a good discriminative model as a heuristic for assessing the tradeoff between usefulness and privacy in TabDDPM. This intuition stems from the relationship between AUC and the usefulness/privacy tradeoff at extreme points. When TabDDPM generates pure noise, a good discriminative model should achieve an AUC of 100%. Conversely, if TabDDPM merely replicates the real examples, even the best discriminative model cannot achieve an AUC statistically different from 50%.

However, the broader connection between the usefulness/privacy tradeoff and the AUC achieved by the best discriminative model remains unclear and requires further investigation. For example, if we have two generative models, model A and model B, and a discriminative model yields an AUC of 75% for the synthetic data generated by model A and 70% for model B, can we conclude that model A is more private and less useful than model B? Is it possible that model A is both more useful and private than model B? These questions remain unanswered in the current state of our knowledge. Nevertheless, for the purposes of this one-week workshop, we will assume that a higher AUC obtained by a good discriminative model implies poorer usefulness of the generative model.

To the best of our knowledge, using the AUC obtained by a discriminative model to differentiate between real and synthetic examples has not been employed as a measure of the usefulness/privacy tradeoff in a generative model. Therefore, before delving into our methodology to address Q1, we must clarify a few details.

The first question pertains to how we should measure the performance of a generative model in the context of Q1 as different discriminative models will yield varying AUC scores. Should we consider the

[3] http://archive.ics.uci.edu/ml/datasets/Bank+Marketing

average AUC obtained by various machine learning models or should we focus on the AUC produced by a high-performing machine learning model? If we opt for the latter, how do we define "good"? Should it be a model that generally excels on tabular datasets, or should it be the best-performing model among a limited selection of machine learning models for the specific dataset? Although we do not claim this is the optimal strategy, we have decided to measure the generative model's performance using the AUC of the top-performing model on the given dataset.

The second question revolves around whether we can gain additional insights from addressing Q1. Specifically, can we interpret the decision-making process of the discriminative model to identify areas for improvement in TabDDPM? In its current implementation, TabDDPM is tuned to maximized machine learning efficiency. Based on our unproven assumption that a high AUC indicates poor generative model usefulness, if the discriminative model relies on specific features to differentiate between real and synthetic data, it suggests that TabDDPM struggles to replicate that feature or its relationship with other features, leading to poorer machine learning efficiency. By forming concrete hypotheses about why TabDDPM faces challenges with certain features, we can develop potential strategies to enhance TabDDPM.

In summary, we can reframe Q1 into two subquestions.

- Q1a: Find a good discriminative model that distinguishes synthetic examples generated by TabDDPM from real examples used to train it by selecting the best-performing models from a set of candidates.
- Q1b: Determine which features the discriminative model uses to differentiate between real examples and synthetic examples generated by TabDDPM.

Our approach to addressing both subquestions is straightforward. For Q1a, we will explore various standard ML models, with the requirement that they have standard library implementations in Python. In addition we will experiment with various data preprocessing strategies. Regarding Q1b, we will employ the standard feature importance measures associated with the particular ML model and Shap values [10]. We will also test various feature selection strategies and measure the impact on the AUC as we remove features.

3.3.2 Q2

The problem statement for Q2 explores the feasibility of reconstructing an incomplete real-world observation using synthetic data as a starting point.

The problem entailed predicting undisclosed labels for an incomplete test dataset comprising real-world observations, in which 25% of the data was missing. For this, we were provided with a complete, but synthetic, training dataset. This scenario can be framed as an attribute inference attack on real-world censored data. In this context, the primary objective was to enhance the classification accuracy for three distinct labels: Sex (binary), Marital Status (multi-class), and Foreign (binary).

Upon analyzing the occurrences of missing data within the test dataset, we noted that the number of blank entries per sample roughly conformed to a normal distribution. Specifically, this distribution exhibited an approximate mean of 3.5 and a standard deviation of 1.55, as illustrated in Figure 3.2.

We approached the classification challenge by using two distinct ensemble strategies: an imputation approach based on the MICE algorithm, and another based on a set of models trained with noise-injected datasets. Our findings suggest that these approaches can significantly increase the baseline classification performance, particularly for the provided test dataset. In addition we conducted experiments involving a combination of both approaches, resulting in further enhancements in the performance metrics.

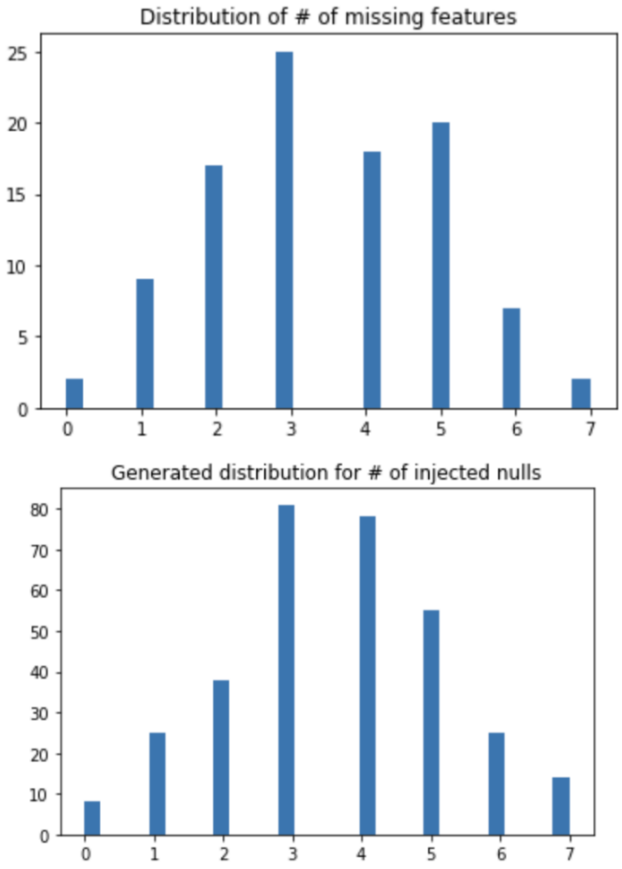


Figure 3.2: (Above) Distribution of number of missing entries in the test set. (Below) Example of simulated distribution for the validation set.

3.4 Results

In this section we present the results of our experiments, dividing our discussion into two parts: Q1 and Q2.

3.4.1 Q1

As discussed in Section 3.3, Q1 can be subdivided into two distinct subquestions: Q1a, which entails identifying a good discriminative model, and Q1b, which involves examining the discriminative model to gain insights that could potentially enhance the TabDDPM model.

Q1a

Our goal for Q1a is to find a good discriminative model. As defined in Section 3.3, a good discriminative model is the best-performing model on the given dataset from a predefined list of machine learning models.

In our experiment, the list consists of three different machine learning models: logistic regression [12], random forests [3], and XGBoost [5]. We use the implementations from Scikit-learn [13] for logistic regression[4] and random forest[5], while we use the implementation from the authors of the XGBoost

[4] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
[5] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

paper for XGBoost.[6] More precisely, we use the default hyperparameter settings for all models, except for Random Forest and XGBoost, for which we use 200 estimators instead of the default 100. This list is not exhaustive and future work may explore different machine learning models and/or different hyperparameter settings.

We also consider various data preprocessing strategies. For categorical data, we explore three different preprocessing strategies: 1) one-hot encoding[7]; 2) target encoding[8]; and 3) no encoding. The last option is only available for XGBoost, as it is the only model in our list that supports non-numeric features. For numerical data, we investigate three different preprocessing strategies: 1) transformation to a uniform distribution, 2) transformation to a normal distribution, and 3) no transformation. Both transformations to the uniform and normal distribution are performed using the scikit-learn quantile transformer[9] with default hyperparameters.

Table 3.1 displays the results of our experiments. All scores correspond to the average score over a five-fold cross-validation on the training set. We use the term "raw" to denote no encoding.

Table 3.1: AUC obtained by discriminative models using different preprocessing strategies.

	Logistic Regression	Random Forest	XGBoost
Raw + Raw			0.857
Raw + Uniform			0.857
Raw + Normal			0.857
One Hot + Raw	0.552	0.704	0.858
One Hot + Uniform	0.581	0.704	0.858
One Hot + Normal	0.58	0.704	0.858
Target + Raw	0.508	0.759	0.856
Target + Uniform	0.578	0.762	0.856
Target + Normal	0.576	0.763	0.856

From the table, we can draw the following observations:

- The XGBoost model performs the best, achieving an AUC of 86%;
- Transformations on the numerical variables have very little to no impact on the Random Forest and XGBoost models. This is expected as both transformations are monotonic;
- Transformations on the numerical variables enhance the performance of the logistic regression model, but it still lags significantly behind the other two models;
- Using one-hot encoding benefits the logistic regression and random forest models but has little impact on the XGBoost model.

In summary, our experiment reveals that XGBoost is the "good" model and preprocessing strategies have minimal impact on its performance. Moreover, with an AUC of 86%, XGBoost clearly possesses the capability to discriminate between real and synthetic samples generated by TabDDPM, implying clearly that TabDDPM has room for improvement. In the following subsection, we attempt to understand why the XGBoost model can discriminate between real and synthetic examples, seeking insights that may lead to potential enhancements of the TabDDPM model. We also observed that other machine learning models, such as Random Forests, also exhibit discriminative abilities and may employ different decision boundaries from those of XGBoost to distinguish between synthetic and real data. Exploring these decision boundaries could also provide valuable insights, although this is left for future work.

We should also mention that we attempted to provide the anomaly score predicted by the Local Outlier Factor[10] as an additional feature to the XGBoost model. Our assumption here was that

[6] https://xgboost.readthedocs.io/en/stable/python/python_intro.html
[7] https://contrib.scikit-learn.org/category_encoders/onehot.html
[8] https://contrib.scikit-learn.org/category_encoders/targetencoder.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html
[10] https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.LocalOutlierFactor.html

synthetic samples would have a larger anomaly score than real samples. This conjecture was based on the fact that the distance to the closest synthetic sample is often used as a heuristic in membership inference attacks, and our setting has some similarity with membership inference attacks. More precisely, in both cases, we are trying to establish whether a sample was used to train the generative model. We did not see any improvement, however, with the addition of the anomaly score.

Q1b

Our goal for this section is to gain insights into how XGBoost is capable of discriminating between real and synthetic samples. Note that for this part of the experiment, we use the target encoder to preprocess categorical features. This choice to "target encode" the categorical variables was made because the feature importance of target-encoded features is easier to interpret than the feature importance of one-hot-encoded variables, and the Shap package does not support raw categorical variables. We do not apply any preprocessing steps to the numerical variables. Indeed, as established in Section 3.4.1, preprocessing steps have a limited impact on the performance of XGBoost. Nonetheless, future work could explore whether different insights are obtained with different preprocessing methods. Our first approach to understanding the method used by XGBoost to discriminate between real and synthetic data is to examine the feature importance score and the magnitude of the Shap values. The measures of feature importance are computed over an arbitrary cross-validation fold. The results are shown in Figures 3.3 and 3.4.

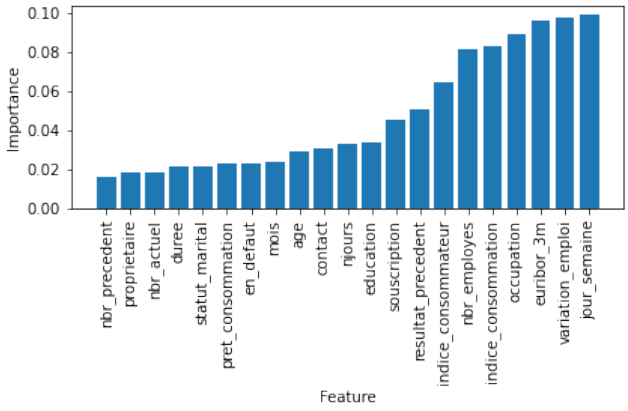


Figure 3.3: XGBoost Feature Importance.

From this, we can observe that both the feature importance and Shap values identify the same six features as the most important: euribor_3m, indice_consommation, nbr_employes, jour_semaine, variation_emploi, and occupation, although with a different ordering. This seems to imply that TabDDPM has difficulty replicating these features or at least that they are synthesized in a distinguishable manner.

To further test that hypothesis, we train the XGBoost model with one feature removed to measure the associated impact on the AUC. Figure 3.5 shows the results of the experiment with the AUC displayed in this figure being the average over a five-fold cross-validation. We observe that for the most part, removing one feature from the model has limited impact on the AUC obtained by XGBoost. There are two exceptions, however: removing euribor_3m (71.6%) and jour_semaine (77.6%) leads to a significant drop in the AUC, implying that XGBoost has great difficulty generating realistic representations of these features or their relationships with other features.

We also performed a test in which we iteratively added and removed features based on the magnitude of the Shap values and measured the impact on the AUC. Specifically, at iteration i, we considered two models: one consisting of the $i+1$ most important features according to Shap and one consisting of the $n-i-1$ least important features (in which $n=20$ corresponds to the total number of features).

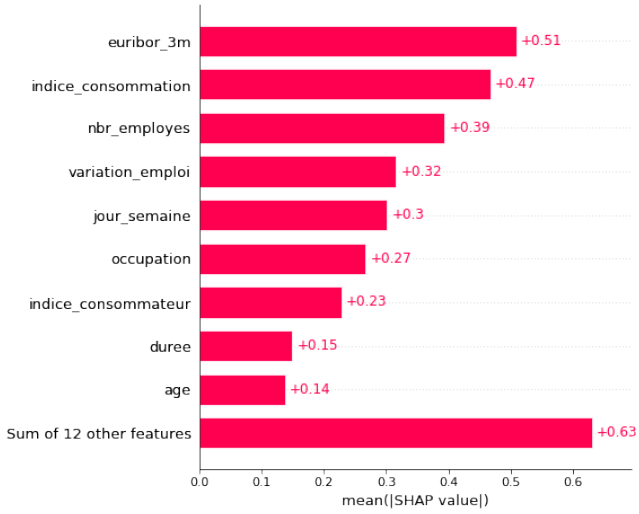


Figure 3.4: Shap Values.

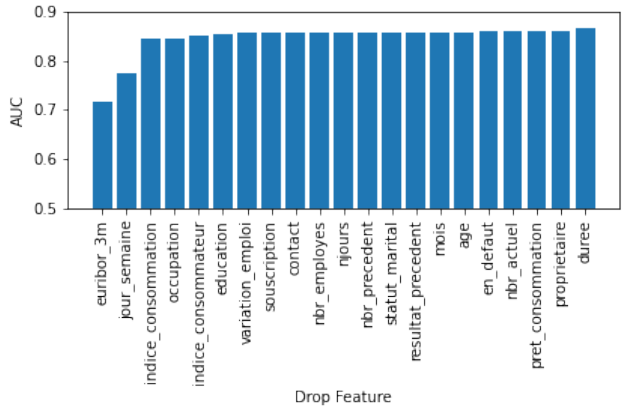


Figure 3.5: AUC obtained by XGBoost without given features.

We will refer to the first model as "keep" and the second as "drop." For example, at iteration 3, the "keep" model is fitted with euribor_3m, indice_consommation, nbr_employes, and variation_emploi, while the "drop" model is fitted with all the other features.

Figure 3.6 shows the results of the experiment. We can make the following observations from this figure.

- The score of the "keep" model is approximately monotonically increasing with the iterations. This is expected because at each iteration, we provided it with additional information. The score of the "drop" model is approximately monotonically decreasing with the iterations. Again this is expected, since at each iteration we remove information from the model.
- The "keep" model plateaus around iteration 4. In fact, the AUC of the XGBoost model with only euribor_3m, indice_consommation, nbr_employes, variation_emploi, and jour_semaine achieves an AUC of 84.8%, whereas a model with all features achieves only a slightly better AUC of 85.6%. This implies that TabDDPM is struggling to replicate the relationship between those 5 features.
- Looking closely at iteration 4, we see that the AUC of the "keep" model increases significantly, but the AUC of the "drop" model was barely affected. This implies that jour_semaine is important only because TabDDPM is struggling to replicate its relationship with euribor_3m,

indice_consommation, nbr_employes, and variation_emploi, whereas if we look closely at iteration 5, we see that the AUC of the "drop" model decreases significantly, but the AUC of the "keep" model does not change significantly. This implies that occupation is important for another reason than its relationship with the five features with the largest Shap value magnitude.

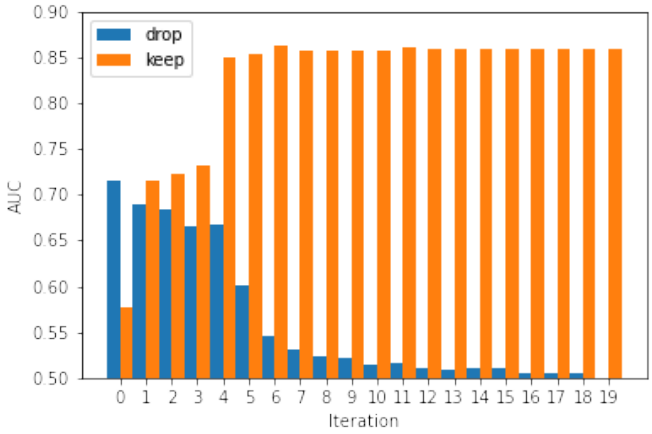


Figure 3.6: AUC obtained by XGBoost at each iteration.

Based on these three experiments, we can conclude that TabDDPM is struggling to model euribor_3m, indice_consommation, nbr_employes, variation_emploi, and jour_semaine, but for the most part, it does a good job of modelling the other features. An important next step, which we leave for future research, is to obtain a better understanding of why TabDDPM is struggling to model euribor_3m, indice_consommation, nbr_employes, variation_emploi, and jour_semaine. It is important to note that during the presentation at the end of the workshop, we presented a conjecture that TabDDPM was struggling to model multimodal numerical features. While this hypothesis may turn out to be true, after further analysis of the evidence, we decided that the evidence supporting our conjecture at this point was minimal, and as a result, we decided to remove that conjecture from this report.

3.4.2 Q2

Evaluation strategy and performance metrics

The official evaluation metric for the task was Mean Accuracy, which is defined as

$$\text{Mean Acc} = \frac{1}{3}\left(\text{Acc}_{Gender} + \text{Acc}_{Foreign} + \text{Acc}_{MaritalStatus}\right).$$

Given that testing was conducted on the Kaggle platform, which limits the number of daily submissions to two, we partitioned the training data into separate train and test sets for conducting internal evaluations. As we anticipated encountering incomplete test data, we introduced missing values into the validation partition features. The distribution of these missing values was derived from the details outlined in Section 3.3.2. This approach enabled us to obtain a preliminary estimate of the expected performance of the proposed systems.

Data encoding

Given that our dataset consisted of tabular data, we anticipated encountering either categorical or numerical entries. Using an approach similar to that in Q1, we processed numerical data using a Quantile Transformer, while categorical data was represented using one-hot vectors. Subsequently, we concatenated all the encoded features and inputted them into the classification system. In situations in which a numerical feature was missing, we imputed its encoded value with the mean of that particular feature. For categorical data, we assigned a value of zero to all entries in the event of missing data.

Baselines

We conducted an evaluation of several baseline classifiers, including K-Nearest Neighbors (KNN), XGBoost, LightGBM, and Support Vector Machine (SVM). To optimize their performance, we employed a grid-search cross-validation strategy on the training dataset to fine-tune their parameters.

The performance results of the evaluated classifiers is shown in Table 3.2. Due to its optimal performance in these preliminary tests, we employ a Support Vector Machine as the base classifier for our proposed ensemble approaches.

Table 3.2: Baseline mean accuracy for various classifiers. It is obvious that the presence of missing data in the testing partition leads to a decrease in performance. Therefore our objective is to design a system that can, at a minimum, achieve performance levels equivalent to evaluating with a complete test set.

	Mean accuracy	**Mean accuracy**
Model	validation + injected nulls	validation complete
Majority Label	0.6008	
KNN	0.7330	0.7593
SVM	0.7335	0.7623
XGBoost	0.7191	0.7572
LightGBM	0.7130	0.7706

In the baseline configuration, these models were trained using the entire synthetic dataset and applied to predict labels for the real but incomplete test dataset. Our objective was to enhance the average classification accuracy beyond the baseline performance of these models to the greatest extent possible.

A multi-MICE approach

The MICE (Multiple Imputation by Chained Equations) algorithm, as described in [2], is designed for imputing missing values in tabular data. It follows an iterative approximation process, which can be summarized as follows.

1. Begin by imputing missing values in each feature using temporary data derived from the available non-missing values for that particular variable, such as the mean.
2. Remove the temporary data for a given feature and regress it using the other features that are either observed or previously imputed.
3. Use the fitted regression model to predict the missing values in that specific feature.
4. Repeat steps 2–3 iteratively for each variable that still has missing values.
5. Perform multiple cycles of steps 1–4.

While this approach is effective, it can exhibit sensitivity to the random seed, as the order in which different features are imputed during the algorithm's cycles can result in different imputations. Consequently, the performance gain may vary significantly due to the randomness.

To enhance the robustness of our system, we adopt a strategy that involves employing multiple sets of MICE-imputed features with different seeds. Subsequently, we perform soft-voting over their outputs when used in conjunction with a given classifier (see Figure 3.7).

Null-injection ensemble

In addition to imputation, we hypothesize that a model trained on incomplete data might exhibit superior performance compared to a model designed for complete input features when evaluated on incomplete data. As such, we introduce missing values into the training data using the distribution outlined in Section 3.3.2, employing various seeds to create multiple incomplete training sets.

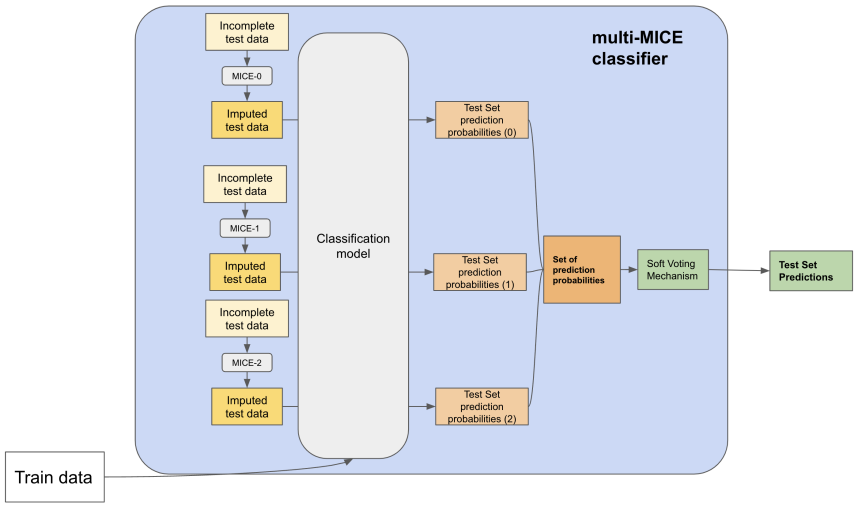


Figure 3.7: Diagram of the multi-MICE system.

For each of these modified datasets, we train distinct classifiers and subsequently combine their output predictions through a soft voting mechanism. Note that this approach is relatively simpler to implement when compared to the MICE algorithm, as it does not involve running the MICE algorithm but does require fitting a classifier for each of the training sets with missing entries. In contrast, the multi-MICE approach necessitates only a single fitted model (see Figure 3.8).

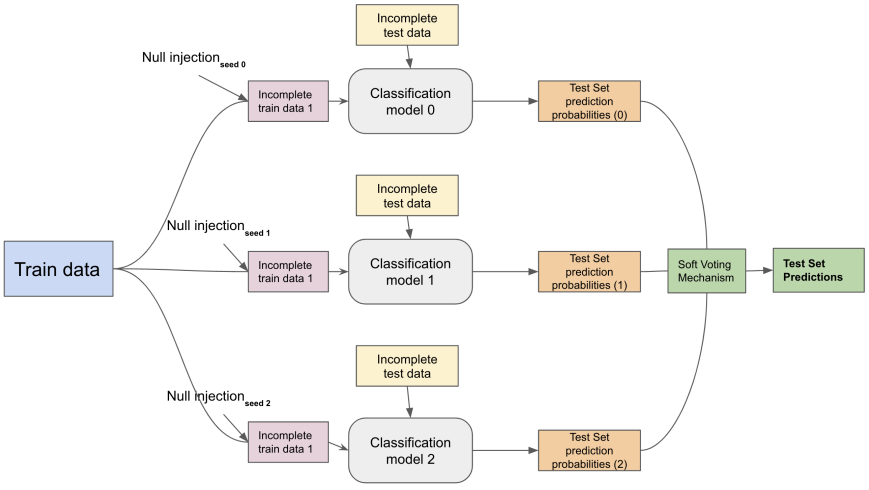


Figure 3.8: Diagram of the null-injection system.

Combining both approaches

It is worth emphasizing that, given the nature of the two proposed ensemble approaches, it is entirely feasible to integrate them into a single pipeline, potentially enhancing the effectiveness of the attribute inference attack. This setup involves working with multiple training sets and multiple test sets, as illustrated in Figure 3.9.

The performance results on the validation partition for both the multi-MICE and null-injection approaches, as well as their combined utilization, are presented in Table 3.3. It is obvious that in the

validation set, both approaches substantially outperform the baseline metrics. Furthermore there is a noteworthy performance gain achieved by combining these two approaches, underscoring their potential complementary nature.

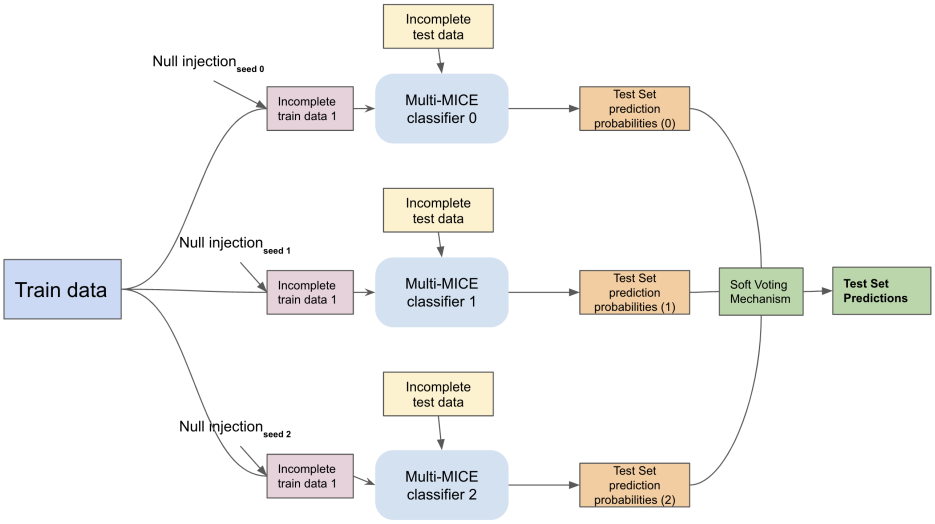


Figure 3.9: Diagram of the combined approach.

Table 3.3: Performance of proposed methods in the validation partition. We observe that both multi-MICE and null-injection approaches outperform the baseline methods.

Model	Method	Mean accuracy (Test)	%Δ over baseline
SVM	none (baseline)	0.73297	N/A
SVM	null injection ensemble	0.7737	5.552
SVM	multi-MICE	0.7634	4.148
SVM	null injection ensemble + multi-MICE	**0.7757**	5.832

Results on the test partition

At the conclusion of the workshop, we had the opportunity to examine the evaluation results on the official testing partition hosted on Kaggle. The results are presented in Table 3.4. Notably we included submissions that used multiple classifiers in an ensemble configuration.

From the table, it is obvious that the proposed method has yielded improvements in baseline performance. Specifically, we observed an enhancement of approximately 8% for the SVM classifier and an even more substantial improvement of up to 12% when employing a diverse set of classifiers in the ensemble configuration.

Table 3.4: Results on official testing partition, we can observe a clear improvement in Mean Accuracy. The Classifier Ensemble consists of SVM+XGBoost+KNN.

Model	Method	Mean accuracy (Test)	% improvement
SVM	none (baseline)	0.7063	N/A
Classifier Ensemble	none	0.7222	2.248
SVM	null injection	0.7460	5.619
SVM	multi-MICE	0.7698	8.990
SVM	null injection + multi-MICE	0.7619	7.866
Classifier Ensemble	null injection + multi-MICE	**0.7937**	12.361

3.5 Conclusion

In this report, we presented our initial assessment of the privacy of the TabDDPM model as a synthetic data generator. We raised the following two concerns:

1. It is possible to discriminate between real and synthetic data generated by TabDDPM;
2. TabDDPM is susceptible to attribute inference attacks.

Due to the high risk associated with synthetic data and the limited time available during the workshop, we are unable to provide a definitive recommendation on the viability of TabDDPM as a synthetic data generator at Desjardins at this point. We suggest that Desjardins conduct a more thorough investigation before determining whether TabDDPM satisfies their synthetic data generation needs. A comprehensive investigation should aim to answer the following questions.

- What is the relationship between the AUC obtained by a good discriminative model for separating synthetic from real data and the tradeoff between usefulness and privacy?
- Why is TabDDPM struggling to generate euribor_3m, indice_consommation, nbr_employes, variation_emploi, and jour_semaine but not other features?
- Do other models or preprocessing steps lead to different insights regarding which features TabDDPM struggles with?
- Is TabDDPM susceptible to membership inference attacks?

In a broader perspective, TabDDPM shows promise as a synthetic data generation method but requires further investigation before it can be applied in real-world settings. In relation to the system developed for the attribute inference attack, our empirical findings demonstrate the possibility of reconstructing censored real-world datasets solely by utilizing synthetic data generated through TabDDPM. In particular, our proposed system achieves an accuracy rate of more than 79% on the provided test set. This result shows that in order to render it impossible to reconstruct the real-world observations, a more complex censoring algorithm may be necessary, as mere random feature deletion appears to be insufficient.

The observation that the model itself attains a higher mean accuracy compared to the majority label, coupled with the fact that both the multi-MICE imputation and the null-injection ensemble contribute to enhancing the classification accuracy of the model, hints at the possibility that the synthetic data may indeed contain information about the underlying correlations between the features. This information, in turn, enables these systems to effectively reconstruct the observations.

Acknowledgments

We would like to express our sincere gratitude to Prof. Sébastien Gambs for his guidance and support as the supervisor of our group. We would also like to thank Dr. Adel Benlagra and Dr. Antoine Langevin for their assistance during the project. Finally, we would like to express our appreciation to Desjardins for providing us with the opportunity to work on an interesting and relevant problem during the workshop.

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 2623–2631, 2019.

[2] Melissa J Azur, Elizabeth A Stuart, Constantine Frangakis, and Philip J Leaf. Multiple imputation by chained equations: what is it and how does it work? International journal of methods in psychiatric research, 20(1):40–49, 2011.

[3] Leo Breiman. Random Forests. Machine Learning, 45:5–32, 2001.

[4] Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, pages 321–357, 2022.

[5] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 13-17-Augu:785–794, 2016.

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. Advances in Neural Information Processing Systems, 2020-Decem(NeurIPS 2020):1–25, 2020.

[7] Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions. Advances in Neural Information Processing Systems, 15(NeurIPS):12454–12465, 2021.

[8] Florimond Houssiau, James Jordon, Samuel N. Cohen, Owen Daniel, Andrew Elliott, James Geddes, Callum Mole, Camila Rangel-Smith, and Lukasz Szpruch. TAPAS: a Toolbox for Adversarial Privacy Auditing of Synthetic Data. pages 1–8, 2022.

[9] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling Tabular Data with Diffusion Models. pages 1–15, 2022.

[10] Scott Lundberg and Su-In Lee. A Unified Approach to Interpreting Model Predictions. In 31st Conference on Neural Information Processing Systems, 2017.

[11] Harry Markowitz. Portfolio Selection. The Journal of Finance, 7(1):77–91, 1952.

[12] P McCullagh and J.A. Nelder. Generalized Linear Models. Chapman and Hall, 2 edition, 1989.

[13] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[14] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: Unbiased boosting with categorical features. Advances in Neural Information Processing Systems, 2018-Decem(Section 4):6638–6648, 2018.

[15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. 32nd International Conference on Machine Learning, ICML 2015, 3:2246–2255, 2015.

[16] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. Advances in Neural Information Processing Systems, 32(NeurIPS), 2019.

[17] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. CTAB-GAN: Effective Table Data Synthesizing, volume 157. Association for Computing Machinery, 2021.

[18] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. CTAB-GAN+: Enhancing Tabular Data Synthesis. (1):1–13, 2022.

4 Hitachi Energy: Towards an effective alarm flood understanding

Antony Hilliard [a]
Moncef Chioua [b]
Antony Hilliard [a]
Slim Ibrahim [c]
Judith Tavarez [d]
Piotr Morawiecki [e]
Benjamin Nguyen [b]
Rohan Singh [b]

[a] *Hitachi Energy*
[b] *Polytechnique Montréal*
[c] *University of Victoria*
[d] *Centro de Investigación en Matemáticas (CIMAT)*
[e] *University of Bath*

March 2024
Les Cahiers du GERAD

4.1 Introduction

Humanity is facing new challenges due to climate change, one of the most significant being the need to transition to a more sustainable way of living. In the context of electrical energy, the capacity to generate renewable energies must increase several-fold from its current levels. For example, electric vehicle sales are expected to increase 18-fold compared to current levels. In the industrial sector, emissions must decrease by 20% by 2030 and 90% by 2050[1] [1]. Issues such as these drive demand for electrical grid services, requiring adding more and "smarter" equipment, which adds more alarms in existing network management systems.

Alarm systems are a vital support tool for operators of electrical grids and an essential component of the interface between automated and manual control. Alarms play a crucial role in the detection and mitigation of abnormal situations that require the operator's attention. A well-configured alarm system can greatly facilitate operations. Conversely, a poorly configured alarm system can distract the operators' attention from important information and increase their workload.

The number of alarms in electrical systems is rapidly increasing, exacerbated by the lack of guidelines for creating coherent and informative alarms. During a disturbance, the quantity and rate of alarms can be too high for operators to comprehend and act upon, potentially hindering their ability to manage the situation.

A key challenge is that alarms are often coupled, meaning one alarm is followed by a sequence of related alarms. Due to the speed of electric phenomena and the nature of distributed alarm processing systems, it can be difficult to determine which alarm initiates the sequence or understand the underlying process connecting the alarms.

Alarm floods occur when a very high rate of alarms (up to several hundred alarms in a few seconds) results from disturbances in the electrical network or poorly configured alarms. Therefore, another significant challenge is the need to summarize a flood of alarms to detect the primary issue at hand. In other words, it is crucial to describe the characteristics of a group of numerous alarms in an informative manner to help the operator understand what happened and identify the most critical problem that should be addressed first.

The problem addressed in this report stems from the Hitachi Energy proposal in the 13th Montréal Industrial Problem Solving Workshop (13th IPSW). It involves analyzing a large dataset of historical alarms to identify which alarms might follow others or can be predicted based on them. It also aims to summarize alarm floods more effectively to identify the main issues.

These challenges are significant because:

- Causal relationships within the electrical network may not be stable, as network equipment may be connected differently over time;
- Alarm timestamps can be inaccurate: for example, even if alarm A occurs before alarm B, sometimes alarm B may reach the alarm system before alarm A;
- The dataset is disordered, with unrelated alarms occurring within sequences of related alarms.

The provided dataset for demonstrating solutions consists of periods of alarm floods ($\sim$4680) identified out of 0.5 million alarms collected over an eight-month period. An indication of the type of disturbance is provided for a limited number of alarm floods. Each alarm includes the following information: an alarm text that generally describes the alarm (a brief description + limit value + violation value), an object identifier that generated the alarm, a general geographic area to which

[1] IEA's Net Zero by 2050 report

the object belongs (substation), a data type (discrete or continuous measurement), an object class describing the type of measurement (voltage, current, etc.), and a state transition value (normal limit to lower or upper limit from 1 to 2). The information described is contained in the following fields.

timestamp	alarm_priority	station_id
alarm_reason_code	control_center_id	scada_object_id
previous_state	new_state	new_state_text
unacknowledged_alarm	persistent_alarm	alarm_type
record_number	not_normal_state	alarm_parameter
source_time	alarm_subtype	flood_no

Solving the problem of identifying causal and process-based relationships among alarms in real data could be used in online decision support tools to help electrical grid operators address disturbances. For this reason, during the IPSW week, two approaches were explored:

- The first approach aims to address the problem of summarizing alarms to focus on high-priority issues;
- The second approach focuses on solving the problem of identifying the origin of alarms and defining the alarm sequence.

Each method employed distinct data columns from the dataset and applied different data preprocessing techniques. Details on the processes of these two methodologies are provided in the subsequent sections.

4.2 First approach: Clustering by Jaccard distance

4.2.1 Introduction

Our first strategy for addressing the problem was to create a tool which would quickly diagnose an alarm flood based on the historical data and provide the operator with actionable suggestions. In order to accomplish this, we decided to divide the alarm flood data into clusters of floods with similar alarms, verify that the patterns for each cluster are physically meaningful, assign the floods in the clusters labels, and then train a classifier based on these labels (Figure 4.1). The trained classifier can then be deployed online for real-time diagnosis of alarm floods.

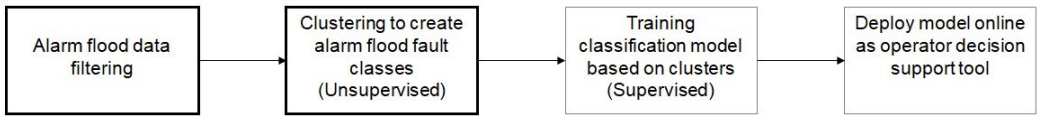


Figure 4.1: Pipeline of alarm flood diagnostic tool implementation. The bolded squares correspond to what was completed during this workshop.

Due to time constraints, only the data filtering and clustering steps were completed during this workshop. The design and training of the classification model and on-line deployment are saved as future work. The methodology used was based on [2], who clustered alarm floods for a chemical process application. In this work the method is adapted to power systems and is outlined in the following sections.

4.2.2 Preprocessing

Before performing clustering, we must define the criteria for a unique alarm and then filter the data to avoid generating clusters with redundant and less useful information.

Defining a unique alarm

Each alarm in the dataset contains multiple descriptors that give different types of information. These include the location (Station ID), time it occurred (Record number, time-stamp), type of equipment affected (SCADA object ID, alarm subtype), alarm description (reason code, previous and new state), and other features which are functions of the other states (Figure 4.2). We need to choose a combination of these features to decide what constitutes a unique alarm and what to base the alarm flood similarity on. This becomes a choice of too many vs. too few unique alarms while also considering what we want the clusters to mean. A choice of too many unique alarms results in sparsity (where it may be difficult to form clusters) while a choice of too few unique alarms would result in clusters that are too general to be useful.

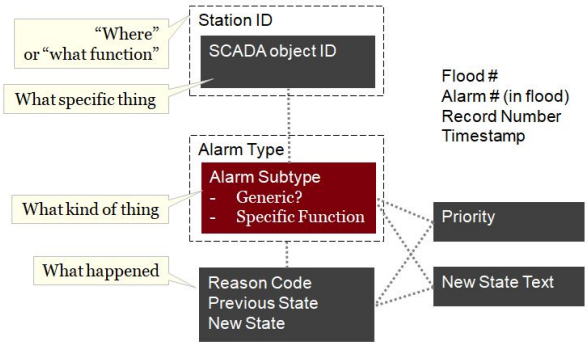


Figure 4.2: Diagram indicating the relevant fields of information contained within an alarm.

Given the above considerations, a combination of "Alarm Subtype" and "New state" was chosen as the criterion for a unique alarm. Alarm subtype represents what kind of object triggered the alarm and the new state is a good descriptor for what happened as the new state is often either a physical (e.g. "OFF") or a specific fault state. The alarm subtypes were further grouped into more general types using domain knowledge to improve the interpretability of the clusters. Using this combination, we can find clusters that describe alarm floods that trigger the same types of objects in the same way (e.g., a switch that opens and a generator that turns off). With this method the clusters represent patterns with a more general fault cause that are independent from location. The other descriptors were omitted during clustering but analyzed afterwards for pattern recognition and interpretation.

Alarm filtering

After choosing a definition of alarm uniqueness, the alarms still needed to be filtered to avoid generating clusters with redundant and less useful information. During preliminary analysis, we noticed that there was an abundance of alarms containing the string "RDV" in the "new state text" representing a communication loss alarm. The domain expert confirmed that these alarms were not critical and perhaps nuisance alarms, and any alarms containing this string were removed. After further consulting the domain expert, we also removed alarms with subtype "O" (meaning "generic, uncategorized"), as well as all discrete alarms (alarm type 1) that had new states other than 1 or 2, since for such binary alarms other states were illogical. Removing these types of alarms simplified and focused the problem on alarms that have an easier physical interpretation and are more useful for diagnosis.

Furthermore, alarms that were deemed as "chattering" were also removed. Chattering occurs when an alarm makes repeated transitions from normal to fault state more frequently than physically possible [2]. This can be caused by a combination of noise and operating close to the alarm limit [2]. Chattering alarms do not provide more information to the alarm flood: therefore to remove these

alarms, we eliminated repeated alarms in the same flood with the same SCADA object ID, alarm subtype, and new state.

After filtering, many of the alarm floods were reduced to a few alarms only and thus by definition were no longer alarm floods: in theory they can be managed by the operator. A properly tuned alarm system could automatically perform the filtering above and so the resulting floods of fewer than four alarms were also removed. The original dataset had 5860 instances of alarm floods and after filtering, contained 729 floods.

4.2.3 Clustering

Once the filtered dataset was obtained, clustering was carried out to find groups of floods with similar patterns. First, a similarity metric had to be chosen. Similar to [2], we chose the Jaccard distance due to its simplicity and invariance to sequence. For the latter, although the sequence of alarms in an alarm flood could be used to diagnose and perhaps determine causality, we were advised by the domain expert that an accurate sequence could not be conclusively determined from the time descriptors. This is because the time delay between alarms in the floods are very short (milliseconds), which is characteristic of power systems.

The Jaccard distance measures the level of similarity or dissimilarity between two alarm floods and is defined in the following equation.

$$J_{S_1,S_2} = \frac{b+c}{a+b+c} \tag{4.1}$$

In this equation a denotes the number of alarms found in both alarm floods S_1 and S_2, b the number of alarms that are in S_1 but not S_2, and c the number of alarms that are in S_2 but not S_1. This distance is a pairwise distance value normalized between 0 (the exact same set of alarms in both floods) and 1 (no alarm appears in both floods). An example for three alarm floods is given in Figure 4.3.

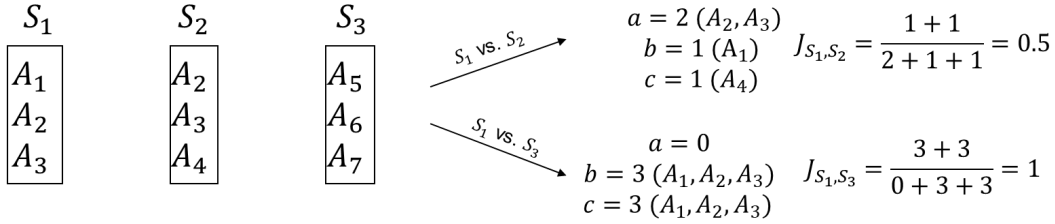


Figure 4.3: An illustrative example of Jaccard distance calculation.

Here we have three alarm sets S_1, S_2, and S_3, and we are measuring the similarity by calculating the Jaccard distance between S_1 and S_2 and between S_1 and S_3. Through inspection, we can see that S_1 has two alarms in common with S_2 (A_2 and A_3) while having none in common with S_3. Thus it is more similar to S_2 than S_3. This is reflected in the calculated Jaccard distance where $J(S_1, S_2)$ is less than $J(S_1, S_3)$. Intuitively it makes sense because similar sets or floods are closer together in this "similarity space."

The Jaccard distance was then calculated for all pairs of alarm floods in the filtered dataset and clustered using an Agglomerative Hierarchical Clustering (AHC) algorithm, similar to that in [2]. In this algorithm, each entity, in this case an alarm flood, begins as an individual cluster and is grouped with other floods based on the lowest Jaccard distances. Once a cluster is formed, the average distance between floods in the cluster (linkage criterion) is used to calculate the distances to the other clusters. This is done until there is just one large cluster. The AHC algorithm was carried out using Scikit-learn's "Agglomerative Clustering" module. Figure 4.4 shows heat maps of the Jaccard distance between alarm floods before and after clustering.

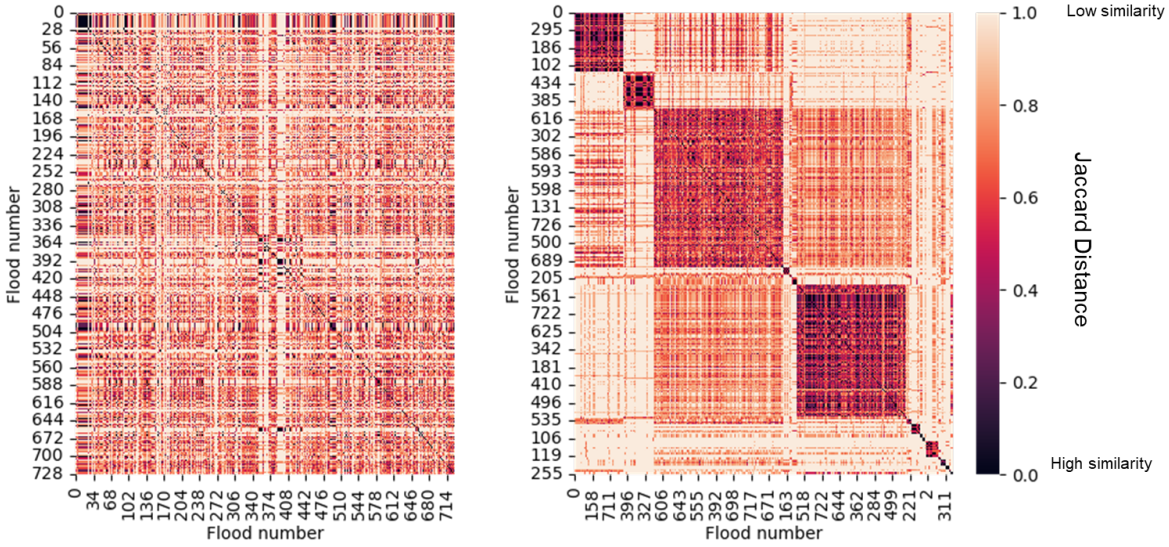


Figure 4.4: Heatmaps of pairwise Jaccard distances between alarm floods, before (left) and after (right) clustering.

The advantage of this method is that it outputs a hierarchical structure that can be viewed as a dendrogram and is more informative than a flat clustering method like k-means [3]. This is particularly useful for our application when we were interpreting and determining the validity of clusters, especially when considering them as future classes for on-line diagnosis. The number of clusters can be tuned for this application. After optimizing the number of clusters and confirming they have a useful meaning, the floods can then be labeled as their respective clusters and used to train a classifier.

4.2.4 Results and discussion

After the clustering phase described above was completed, we arbitrarily decided to analyze the data according to 20 clusters. At this level, alarm floods with a Jaccard distance less than about 0.6 are grouped together and the dendrogram in Figure 4.5 is produced.

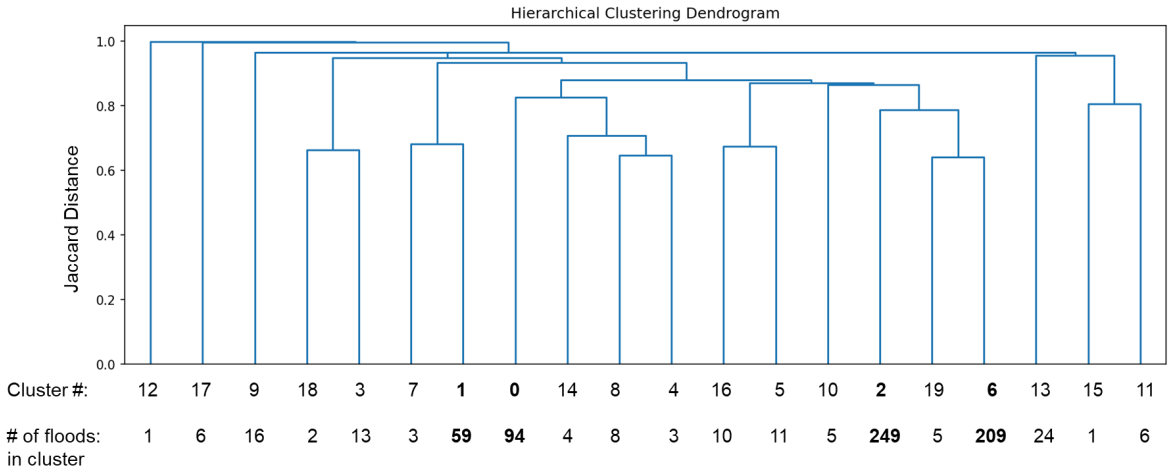


Figure 4.5: Hierarchical clustering dendrogram cut at 20 clusters. The major clusters (0, 1, 2, 6) are in bold.

The cluster number and the number of floods are labeled on the x-axis for each cluster/branch. There appears to be four major clusters (0, 1, 2 and 6) that make up 84 percent of the floods in the

dataset. These clusters were then analyzed together with the domain expert. Similarities in the alarm sub-type and new state within clusters are expected but other descriptors like the SCADA object ID, new state text, previous state, and control centre ID (which are not in the clustering criterion) also provide more information. All these factors were used to characterize the faults that cause the alarm floods in a given cluster. Figure 4.6 gives a summary of the physical meanings of the four main clusters. A more detailed description is given below and in Table 4.1.

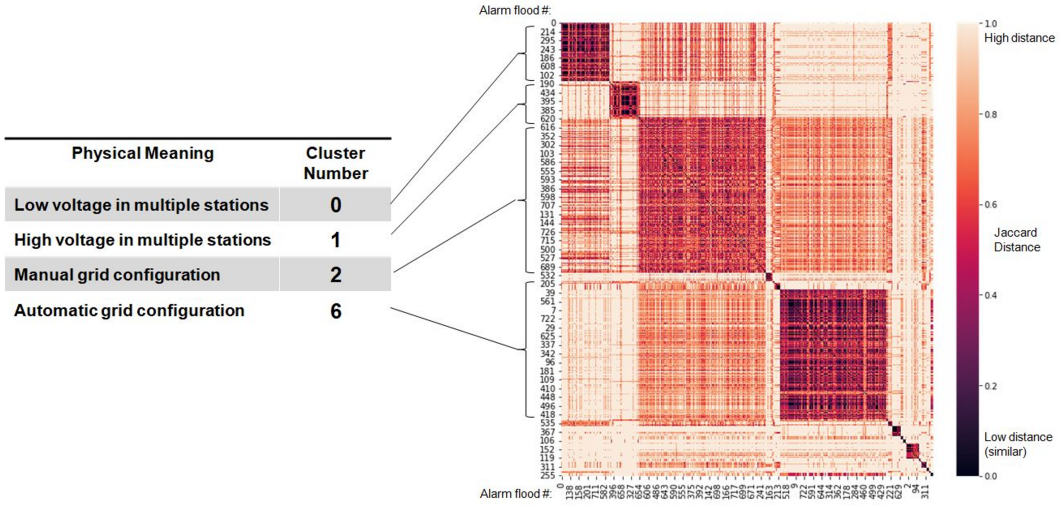


Figure 4.6: Clusters with their inferred meaning.

Clusters 0 and 1

The floods in these two clusters appear to be caused by voltage issues, as indicated by the high frequency of alarms of subtype "V". They differ, however, in New State with cluster 0 having 4 and 6 while cluster 1 has 15 and 17. The clustering algorithm treats the four combinations of type and state as completely separate alarms so the similarity in subtype is not recognized. This is also displayed in Figure 4.6 by lighter sections not along the diagonal where clusters 0 and 1 overlap. To decipher the new states, we looked at the "new state text," which for cluster 0 reads "Into LIM2LO zone" or "Into LIM3LO zone" and for cluster 1 reads "Into LIM2HI zone" or "Into LIM3HI zone." This means that the issue causing the alarm floods in cluster 0 relate to low voltage while the floods in cluster 1 relate to high voltage. Both also have similar priorities, which makes sense given that the priority is a function of the subtype and new state. We also looked at the station IDs and noticed that there were multiple different stations involved. This means that both faults may have a kind of propagation path or cascade with a given root cause. This will be explored further in the second method.

Table 4.1: Summary of descriptors with high frequencies in each cluster. Identified using word clouds.

Cluster number	Alarm sub-types and New state	New state text	Priority
0	V-6, V-4	Into LIM2LO zone, Into LIM3LO zone	14, 24, 29
1	V-15, V-17	Into LIM2HI zone, Into LIM3HI zone	14, 16, 24
2	P-1, V-6, V-4, B-2, V-10, B-1, V-15	IN, OUT, Into LIM2LO zone, Into LIM3LO zone, Entered zero zone	1, 14, 24, 29, 37, 38
6	W-1, B-1, P-2, W-2, B-2, E-2, V-10	OUT, IN, Entered zero zone	29, 37, 38, 63

Clusters 2 and 6

We concluded that the floods in these two clusters were caused by grid configuration, with cluster 2 comprising manual actions while cluster 6 includes automatic reactions. Both have subtypes P, B,

which may pertain to switch positions, while cluster 6 contains subtype W, which could correspond to automatic switches. The floods in these two clusters have more similar combinations of alarm subtypes and new states so their overlap in Figure 4.6 is darker. They also appear closer together in the dendrogram in Figure 4.5, being only separated by two partitions.

4.2.5 Limitations and extensions

This method of using the Jaccard distance with Agglomerative Hierarchical Clustering was seemingly successful in producing meaningful clusters using the alarm subtype and new state. A full analysis of all the clusters and further tuning are needed, however, to determine its true efficacy in clustering well enough to create meaningful classes. This can be done by analyzing each cluster and determining via dendrogram whether they should be combined or split.

Another limitation of these results is the amount of data removed during filtering. A high percentage (88 percent) of the floods were removed due to ambiguity in sub-type or communication errors. We must explore whether the clustering method would be able to automatically filter out these control system alarms and cluster them together without affecting the "real" (physical grid) faults. This would be beneficial with the on-line tool, which determines whether a fault is grave or not.

Further studies can also be carried out by clustering using the geographical descriptors like station ID or control centre ID. It would be interesting to see whether the method can cluster faults that are localized or propagate to specific stations during an alarm flood. Then the "sub-type + new state" clusters could be compared to the geographical clusters to see whether any interesting commonalities can be observed.

Afterwards the next steps in the pipeline (training a supervised model and deploying online) could be completed. For flood classification, one could use state of the art methods such as support vector machines [4], first nearest neighbours [5], or even recurrent neural networks [6]. From there, online deployment presents its own challenges regarding user interface and user experience design, generalizability, and software development.

4.3 Second approach: Maximum likelihood root alarm identification

4.3.1 Introduction

As demonstrated in Section 4.2, alarm floods seem to follow certain patterns, with similar types of alarms being activated within each of the identified clusters. A pattern can also be observed in the geographical location of the alarms, represented in the data by the "`station id`" attribute. Figure 4.7 depicts a graph with nodes representing different stations. Edges connect pairs of stations that tend to occur in the same alarm flood. Additionally, colours were used to highlight nodes closely connected to each other, forming network clusters. This graph clearly demonstrates a correlation between different alarms, but a question remains: what are the causal relations between individual stations? This relates to one of the main problems formulated by Hitachi, namely identifying the original alarm that caused the alarm flood to occur.

Finding causality based on data is a challenging problem, and as we will demonstrate in the next section, it cannot be fully solved. In other words, we cannot definitively identify causal relationships from correlations. As we will show, however, we can attempt to identify causal relations using a maximum likelihood approach, which we will focus on in this section.

4.3.2 Simple toy example

Here, we will motivate our approach by constructing a simple scenario. Let us consider two possible alarms labelled as A and B, with a potential interaction between them. For instance, alarm A may

trigger alarm B, or alarm B may trigger alarm A. The causal relation between them is unknown, but we do have information about the frequency at which the alarms were triggered in the past.

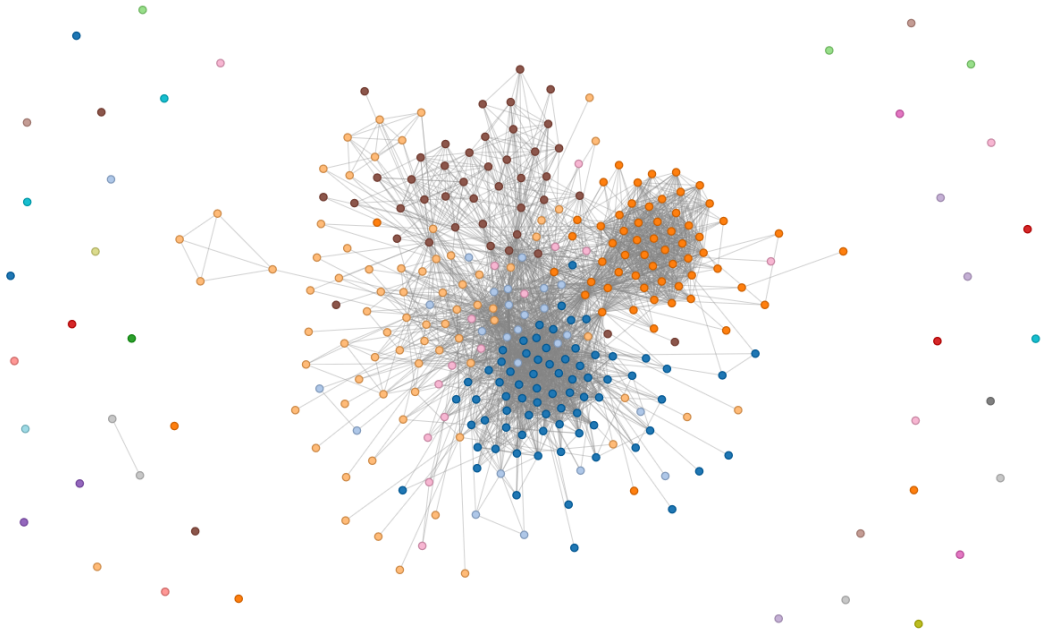


Figure 4.7: Network illustrating the co-occurrence of different stations represented as nodes within the same alarm flood. Nodes were coloured to represent individual network clusters based on their connectivity.

Maximum likelihood approach

Let $N(A)$ be the number of times alarm A was activated, $N(B)$ the number of times alarm B was activated, and $N(A \wedge B)$ the number of times both alarms A and B were triggered simultaneously. We can estimate the following conditional probabilities.

$$\Pr(A|B) = \frac{N(A \wedge B)}{N(B)} \tag{4.2}$$

$$\Pr(B|A) = \frac{N(A \wedge B)}{N(A)} \tag{4.3}$$

For example, if we have $N(A) = 20$, $N(B) = 100$, and $N(A \wedge B) = 10$, as represented in the Venn diagram in Figure 4.8, then the conditional probabilities are $P(A|B) = 0.1$ and $P(B|A) = 0.5$. This could be interpreted in various ways, such as:

- Alarm A triggers alarm B with a probability of 50%, or
- Alarm B triggers alarm A with a probability of 10%,

or other intermediate scenarios where both alarms have certain chances of triggering each other.

Hence it seems that we cannot definitively identify the causal relations based on the available historical records. We can, however, still assess which of these scenarios is more likely.

If both alarms A and B are activated, it is more likely that alarm A was activated first, causing alarm B to trigger with a probability of 50%, than the scenario where alarm B was activated first, causing alarm A to trigger with a five times lower probability. This maximum likelihood approach allows us to point to the most likely root node, even though other scenarios are also possible. We will generalize this approach to larger alarm networks in Section 4.3.3.

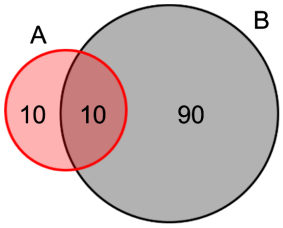


Figure 4.8: Venn diagram representing the data available in the toy example. The numbers represent alarm occurrences where only alarm A was activated (10), both alarms A and B were activated (10), and only alarm B was activated (90).

Incorporating a priori information

In large alarm networks, there may be some alarms that are rarely activated. As a result, the conditional probabilities given by (4.2) could be highly inaccurate. For instance, if $N(A) = 1$ and $N(A \wedge B) = 1$, meaning there was a single event in which alarm A triggered accompanied by alarm B, it does not necessarily imply that alarm B is always activated with alarm A.

To address such scenarios in a more reliable fashion, it is useful to possess some a priori knowledge or guesses about the expected conditional probabilities. This initial information can then be updated based on historical data using Bayesian inference principles.

Let us denote the estimated conditional probability as

$$\gamma = \Pr(A|B). \tag{4.4}$$

Bayes' theorem provides us with:

$$\Pr(\gamma|\text{data}) \propto \Pr(\text{data}|\gamma)\Pr(\gamma). \tag{4.5}$$

In this formula $\Pr(\gamma|\text{data})$ represents the probability distribution of γ values given the available data, $\Pr(\text{data}|\gamma)$ represents the likelihood of observing the data given the conditional probability γ, and the probability distribution $\Pr(\gamma)$ embodies our a priori knowledge concerning possible values of the γ parameter.

Using (4.5), we can deduce the expected value of γ:

$$E(\theta) = \int_0^1 \theta \Pr(\gamma|\text{data})\mathrm{d}\gamma = \frac{\int_0^1 \gamma \Pr(\text{data}|\gamma)\Pr(\gamma)\mathrm{d}\gamma}{\int_0^1 \Pr(\text{data}|\gamma)\Pr(\gamma)\mathrm{d}\gamma}. \tag{4.6}$$

In our case, the likelihood function $\Pr(\text{data}|\gamma)$ represents the probability that there will be $N(A \wedge B)$ activations of alarm A causing alarm B to trigger. This can be described using the binomial distribution as follows:

$$\Pr(\text{data}|\gamma) = \binom{N(A)}{N(A \wedge B)} \theta^{N(A \wedge B)} (1-\theta)^{N(A) - N(A \wedge B)}. \tag{4.7}$$

When employing a uniform distribution $\Pr(\gamma) = 1$ as the a priori distribution, (4.6) simplifies to:

$$E(\theta) = \frac{\int_0^1 \gamma \Pr(\text{data}|\gamma)\mathrm{d}\gamma}{\int_0^1 \Pr(\text{data}|\gamma)\mathrm{d}\gamma} = \frac{N(A \wedge B) + 1}{N(B) + 2}. \tag{4.8}$$

Note that when no data is available, $E(\theta) = \frac{1}{2}$, which is consistent with our initial notion that any θ value between 0 and 1 is equally probable. As available data increases, the above expression approaches $E(\theta) \to \frac{N(A \wedge B)}{N(B)}$, which is consistent with (4.2).

Other a priori distributions could be proposed, perhaps to reduce the estimated conditional probability value when data is limited. For example, if we choose the beta distribution, i.e.,

$$\Pr(\gamma) = B(\gamma; \alpha, \beta) \propto \gamma^{\alpha-1} (1-\gamma)^{\beta-1}, \tag{4.9}$$

from (4.6) we obtain

$$E(\theta) = \frac{N(A \wedge B) + \alpha}{N(B) + \alpha + \beta}. \tag{4.10}$$

In our work, we arbitrarily assign a relatively low a priori conditional probability by taking a beta distribution with $\alpha = 1$ and $\beta = 9$, i.e.,

$$E(\theta) = \frac{N(A \wedge B) + 1}{N(B) + 10}. \tag{4.11}$$

4.3.3 Generalization to large networks

Let us now extend the maximum likelihood approach from the previous section to networks with multiple potential alarms. Our objective is to utilize historical data to identify the most probable root node in a graph of an alarm flood involving multiple alarms. We propose a three-step procedure, as illustrated in Figure 4.9.

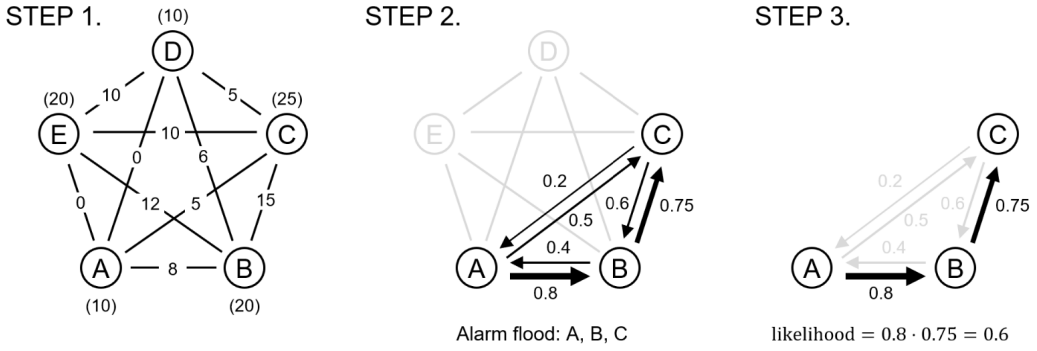


Figure 4.9: Illustration of the three method steps. Step 1 involves determining the total number of floods for each node and node pair. Step 2 entails estimating conditional probabilities between nodes triggered in a given flood (A, B, and C in this instance). Step 3 involves identifying the most likely causal tree.

Step 1. Aggregating Historical Data

Based on all historical floods, we count the activations of each alarm ($N(A)$, $N(B)$, $N(C)$, etc.) and the activations of each pair of alarms ($N(A \wedge B)$, $N(A \wedge C)$, $N(B \wedge C)$, etc.). This real-time process involves incrementing the appropriate counters after each alarm flood (including single alarm activations).

Step 2. Estimating Conditional Probabilities

Whenever a new alarm flood occurs, we need to estimate conditional probabilities $P(A|B)$, $P(B|A)$, $P(A|C)$, etc. for all pairs of triggered alarms in that flood. We achieve this by employing (4.6), computed using a chosen a priori distribution, such as (4.8) for the uniform distribution $\Pr(\gamma) = 1$.

Step 3. Identifying the Most Likely Root Alarm

For each alarm in the flood, we estimate its likelihood to trigger other alarms in the same flood. This involves constructing a probable causal network resembling a directed tree where edges signify activations from one alarm to another. We search for the tree configuration in which the product of conditional probabilities associated with the tree's edges is maximized. For instance, in the scenario depicted in Figure 4.9, node A triggers node B, which in turn triggers node C with a probability of $0.8 \cdot 0.75 = 0.6$.

While Step 3 is the most computationally intensive, the so-called Chu–Liu/Edmonds' algorithm allows us to find the most likely tree with a specific root node in time proportional to the number of nodes. Since it maximizes the sum of all edge weights, instead of their product, for weights we use logarithms of conditional probabilities. Since by definition we have

$$\log(p_1) + \log(p_2) + \log(p_3) + \ldots = \log\left(p_1 \cdot p_2 \cdot p_3 \cdot \ldots\right), \tag{4.12}$$

minimization of the sum of logarithms of conditional probabilities p_1, p_2, p_3 etc., is equivalent to minimization of their product.

Hence we use Chu–Liu/Edmonds' algorithm to find the most likely tree for each node. Then for each tree generated we estimate its likelihood (product of conditional probabilities) and select the root node corresponding to the network with the highest likelihood.

4.3.4 Numerical validation

Numerical implementation

We implemented the procedure described in Section 4.3.3 using Python. The source code, in IPYNB format, is available upon request.

The alarms are selected using a unique combination of the "`station id`" and "`alarm subtype`" attributes (different from the definition used in Section 4.2). For example, an alarm of subtype "V" at station 91986251 is considered to be a single node in the network. Based on historic flood data provided by Hitachi, by this definition 1928 unique alarms were identified.

Then data from 4680 alarm floods provided by Hitachi were used to aggregate historic data in Step 1. In Step 3, we used an implementation of the Chu–Liu/Edmonds' algorithm by Wen Xiao, as described and shared at https://wendy-xiao.github.io/posts/2020-07-10-chuliuemdond_algorithm.

Example of an alarm flood analysis

In this section, we present the results obtained for the first alarm flood, recorded from 2022-01-01 at 03:31:44. In this flood, 10 alarms were activated: one alarm of subtype "V", one of subtype "B", and 8 alarms of subtype "O' associated with different station IDs. The most likely causality tree is presented in Figure 4.10.

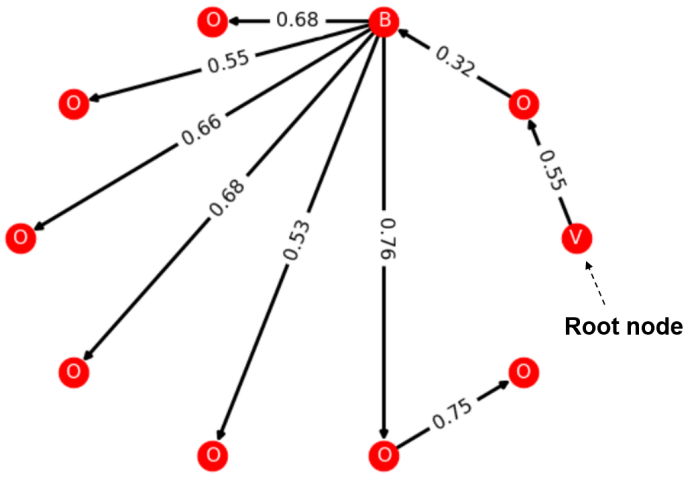


Figure 4.10: The most likely causality tree identified for the first flood event in the provided dataset.

We make two comments on these results.

- In the presented graph, alarm “V” triggers alarm “O”, which in turn triggers alarm “B”, leading to the activation of almost all other alarms. This sequence seems plausible, since “V” and “B” are physical alarms that often lead to generic “O” alarms. Alarm “O”, however, is not likely to be a direct cause of alarm “B” (and it also has a relatively low conditional probability). An alternative explanation is that there is some independent event not captured in the data that triggers both alarms “V” and “B”, leading to the formation of two independent causality trees with two separate root nodes. This scenario, however, is not considered in the current implementations.
- Graphs originating from other nodes are also possible, but they have a lower likelihood. The relative likelihood values for the most likely trees originating from each of the ten possible alarms are summarized in Table 4.2. For example, a tree with the root node “B” has a likelihood nearly two times lower than the one presented earlier; however, it is still a plausible scenario.

Table 4.2: Relative likelihood of the most likely tree starting from the given root node.

station id	alarm subtype	relative likelihood
91986251	‘V’	30%
91985251	‘B’	16%
972618211	‘O’	10%
91985251	‘O’	9%
795849211	‘O’	7%
671083211	‘O’	7%
91986251	‘O’	7%
640376211	‘O’	7%
91987251	‘O’	6%
640395211	‘O’	1%

Accuracy assessment

In this section we shall assess the accuracy of the above procedure. We do this by checking how often the most plausible root node, found using the maximum likelihood approach, was the first alarm to be triggered during the alarm flood.

The main difficulty is that, in general, we do not know the true reason for the past alarm floods, since all of them happen over a very short period of time (e.g., a few milliseconds). The order in which they are registered in the database is often not the same as the order in which they are triggered.

Therefore, we performed the validation only on those alarm floods that were sufficiently long to identify, with a relative confidence, the first triggered alarm. From all alarm floods, we selected those whose duration exceeds 10 seconds, with at least 4 unique alarms being triggered. Flood 959 was not included in the validation set due to its computational complexity, related to the huge number of activated alarms. In total, 306 alarm floods were selected following these criteria. For the true root alarm, we picked the earliest non-generic alarm that appears in the dataset for the given flood.

As a performance benchmark, we use a random node selection method in which the root node is randomly picked from all involved nodes. For example, if four alarms are involved, then the benchmark accuracy is 25%.

Our method identified the correct root cause in 99 out of 306 alarm flood events, which is significantly better than if a random node selection had been used. In that case, an average of 53 nodes are correctly indicated.

As presented in Figure 4.11, the method’s accuracy depends on the number of alarms in the flood. In the case of medium-sized floods, the maximum likelihood approach is approximately two times more accurate than random node selection.

It seems, however, that the method becomes inaccurate in the case of large floods. This may be due to the fact that much less historical data is available for large floods, and as a result, we should expect

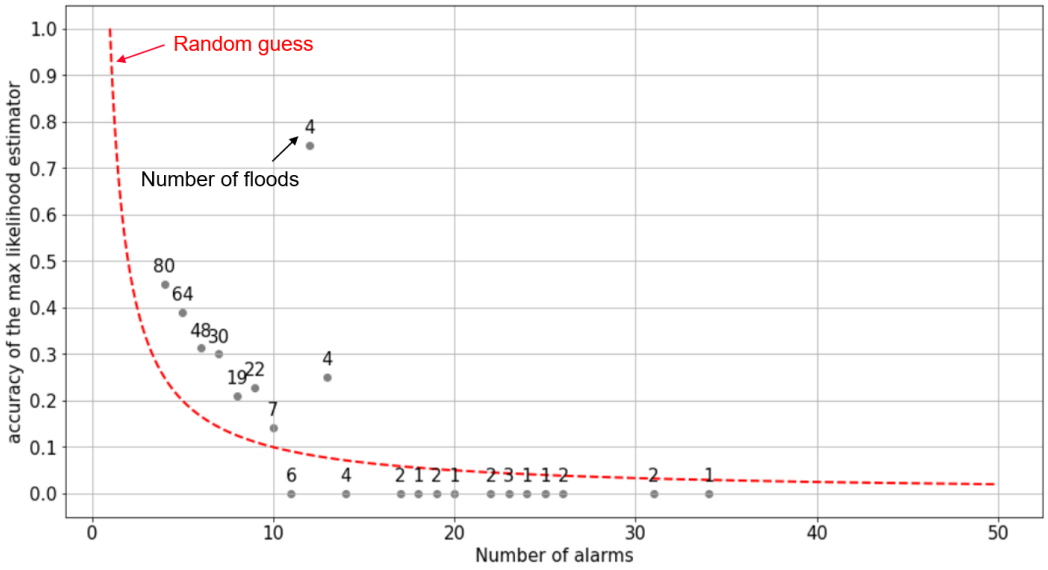


Figure 4.11: The accuracy of the maximum likelihood approach compared to the random node selection for floods of different sizes.

more outliers to appear on the graph. For example, for floods of size 12, 3 out of 4 root nodes were correctly identified, while for floods of size 11, none out of 4 root alarms were identified. Alternatively, this may also be a systematic effect. For example, the tree representation may not be appropriate for these large-scale events, and a different approach should be used (e.g., a method involving many independent root alarms being activated).

4.3.5 Limitations and extensions

The maximum likelihood method turned out to identify the root cause significantly better than random node selection. Its accuracy, however, is still limited. The presented approach has a number of limitations that we summarize in this section, along with possible extensions.

1. **Representation of complex scenarios**
   We assumed that alarms involved within a flood are directly related. While this can be true in the case of some floods, it is often the case that many alarms are activated at similar times due to external reasons (e.g., a thunderstorm). These scenarios, rather than being characterized by a single causality tree, should be represented by a few independent trees. A good example is the flood analyzed in Section 4.3.4, where it is likely that nodes "V" and "B" are triggered independently of each other, triggering other generic "O" alarms.
   Also, some alarm floods may involve alarms that are triggered independently of the main flood itself. This situation is not properly represented in our model. The presented model can be extended to widen the scope of possible causality representations and characterize each of them by a proper likelihood function.
2. **Choice of parameters for the analysis**
   We based our analysis on alarms characterized by only a pair of parameters: "`station id`" and "`alarm subtype`." While this was a good choice when it comes to including both the geographical location of the sensor and the reason for the alarm, there is much more data that can help us understand better the causality relations between alarms.
   Potentially useful information includes the alarm's previous state and new state (used for the clustering described in Section 4.2), alarm priority, or different levels of alarm location (such as its control center id and SCADA object id). For example, we may expect that stations should be

more correlated when they are within the same control center, and we may expect higher priority alarms to have a higher chance of being a root cause of the whole flood.

Generally, we do not recommend using more information to specify unique alarms, as this would significantly increase the number of nodes in the network. A better approach may be to include additional information to refine our a priori guesses about the correlation (and possibly causation) between individual nodes. One can also use Bayesian inference to affect the choice of the most plausible root node.

$$\Pr(\text{root node}|\text{data}) \propto \Pr(\text{data}|\text{root node})\Pr(\text{root node}) \tag{4.13}$$

For example, a prior knowledge Pr(rootnode) can enhance the selection of high-priority alarms, their specific subtypes, or the early alarms as root nodes to increase the method's accuracy.

3. **Data availability**
   There is more information available to experts than we had available during the project. Potentially useful information includes the structure of the electric grid, which would allow us to know a priori which alarms can physically affect each other. Also, for validation purposes, we recommend enhancing the alarm flood dataset by indicating the actual reason(s) why a particular alarm flood occurred based on expert knowledge. Only then can validation be performed in a reliable way.
4. **Other approaches**
   The presented approach was constructed to gain as much information as possible about the grid structure. There are, however, many other approaches that could be used to identify root alarms. Such approaches can range from simple statistical models (e.g., regression) to more complex machine learning approaches (neural networks, random forests, etc.).

4.4 Conclusions and future work

With the increased demand and reliance on electricity due to the energy transition, effective alarm management will be needed for future grids. Floods of alarms are particularly challenging to human operators since they happen almost instantaneously in electric networks. In this paper, two methods were explored to summarize alarm floods.

The first involved applying to power systems an industrial alarm flood clustering method taken from the process industry with a focus on finding clusters with an interpretable physical meaning. This involved focusing on physical alarms only (e.g., excluding communication alerts) and defining uniqueness only in terms of equipment sub-types and end states to have a reasonable number of "unique" alarms. Then the Jaccard Distance was used as a measure of (dis-)similarity between the alarm floods and hierarchical clustering was performed. A reasonable number (20) of clusters were obtained, and the top four most populous were interpreted by the domain expert. This method suggests that some industrial alarm management methods can be transferred to power systems if their input is adapted. Future work is described, including validating the quality of the clusters, using feedback to improve the selection of input data, and demonstrating the use of clusters to train flood classifiers. These can then be used on-line as an operator assistance tool to diagnose real-time floods.

The next method attempted to infer causality in alarm floods based on historic probabilities. Again, by defining alarm uniqueness in terms of a reduced set of features (equipment sub-type and location), the limited historic data available could be used and the number of co-occurrence probabilities was manageable. This method was applied without filtering the input dataset (retaining e.g. communication alarms), and the maximum likelihood formulation was able to identify the root cause better than random selection. This method was also effective in distinguishing "interesting, rare" alarm floods from recurring or common issues. Validating this method could be a next step in determining whether it is suited to identifying underlying structure(s) of electric power system alarm floods.

Both methods share strengths in that they are independent of textual data and any variables not chosen in defining a unique alarm. They also have potential explanations, obtained by analyzing the

proximity of clusters in the dendrogram for the first method and the probabilities in the causal tree for the second method. They share weaknesses, however, and may not perform well when scaled to a large number of unique alarms. In summary, both methods are proven to be effective to a certain extent but much further work is needed in order to evaluate and incorporate them into a proper alarm management system.

Bibliography

[1] S. Bouckaert et al. (2021). Net zero by 2050: A roadmap for the global energy sector, IEA, Paris https://www.iea.org/reports/net-zero-by-2050, License: CC BY 4.0. Last accessed: 13/10/2023.

[2] K. Ahmed, I. Izadi, T. Chen, D. Joe, and T. Burton. (2013). Similarity analysis of industrial alarm flood data, IEEE Transactions on Automation Science and Engineering, 10(2):452–457.

[3] C. D. Manning. (2009). An introduction to information retrieval. Cambridge University Press.

[4] M. Lucke, M. Chioua, C. Grimholt, M. Hollender, and N. F. Thornhill. (2019). Advances in alarm data analysis with a practical application to online alarm flood classification, Journal of Process Control, 79:56–71.

[5] S. Lai, F. Yang, and T. Chen. (2017). Online pattern matching and prediction of incoming alarm floods, Journal of Process Control, 56:69–78.

[6] G. Dorgo, P. Pigler, M. Haragovics, and J. Abonyi. (2018). Learning operation strategies from alarm management systems by temporal pattern mining and deep learning, in Computer Aided Chemical Engineering, Elsevier, 43:1003–1008.

5 Intact: Intact workshop report

Kylian Ajavon [a]
Philippe Gagnon [b]
Gavin Orok [c]
Yasmin Kalhor [d]
Tommy Mastromonaco [e]
Francois Milot [f]
William Morissette [f]
Ernest Tafolong [d]
Zehai Wen [b]
Yue Zhan [d]
Arsene-Brice Zotsa-Ngoufack [g]

[a] *Concordia University*
[b] *Université de Montréal*
[c] *University of Waterloo*
[d] *HEC Montréal*
[e] *Université du Québec à Montréal*
[f] *Intact Assurance*
[g] *Université d'Aix-Marseille*

March 2024
Les Cahiers du GERAD

5.1 Introduction

The insurance industry is a dynamic sector that operates in a high-stakes environment. It involves the assessment and management of numerous variables, ranging from risk evaluations to understanding customer behaviour. In this context, the ability to gain actionable insights is crucial.

The **Customer Lifetime Value** (or CLV) refers to the total expected profit a company expects from a client throughout their entire relationship. It is an important tool for navigating the complexities of the insurance industry. At its core, the CLV serves as a unified metric that encompasses several variables, allowing insurers to understand better, predict, and manage the information they have about their clients. As a result, the CLV facilitates the decision-making process and allows insurers to make more informed decisions about their clients.

It can be modeled as follows:

$$CLV(a) = \mathbb{E}\left[\sum_{t=1}^{T} \gamma^t Profit(S_t) \mid S_0 = a\right], \tag{5.1}$$

where γ is a discounting factor capturing the time-value of money and $Profit(S_t)$ is a function representing the expected profit from a client given his state S_t.

5.2 Method

First and foremost, we need to model $\{S_t\}$ in order to compute the CLV. A simple approach is to view $\{S_t\}$ as a sequence of random variables and assume that $\{S_t\}$ satisfies the Markov property, that is, the following holds:

$$\mathbb{P}(S_{t+1} = s \mid S_t, S_{t-1}, \ldots, S_0) = \mathbb{P}(S_{t+1} = s \mid S_t). \tag{5.2}$$

With this assumption the CLV can be computed using a method described in [1] consisting of the following three steps:

1. Fit a regression tree on the data to identify groups (i.e., the states of the Markov chain) using the profit as a target variable;
2. Estimate the transition probabilities between each group/state;
3. Compute the CLV by using the Monte Carlo method.

Let us dive into the method and illustrate it with a basic example, which involves a portfolio of three customers labeled as **A**, **B** and **C**, and observed at $t = 0, 1, 2$. Each observation consists of feature variables labeled as **X1**, **X2**, and **X3**, and a **Profit** variable.

Step 1 Combine the data from all time periods into one dataset (assuming that customer characteristics are time-independent) and use this dataset to fit a regression tree.

After merging all the data, the regression tree divides the space of features into classes or groups based on the profit variable. Once the classes are formed (groups 0, 1, and 2 in our example), a new feature named **Group** is created by determining which group each observation belongs to, as shown in Figure 5.1 (only $t = 0, 1$ are shown to lighten the illustration). Note that this variable is ordered, since the tree associates a mean profit with each group.

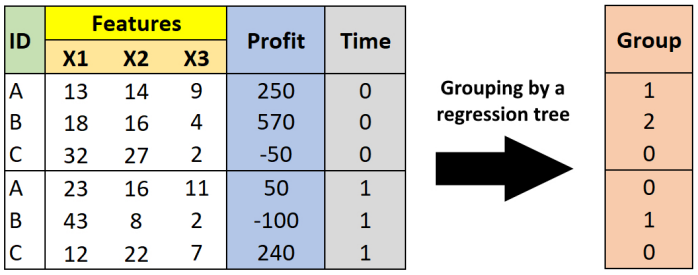


Figure 5.1: Grouping using a regression tree.

Step 2 Build a transition matrix using empirical transition probabilities, assuming the Markov chain is homogeneous:

$$p_{ij} := \mathbb{P}(S_{t+1} = j \mid S_t = i) = \mathbb{P}(S_t = j \mid S_{t-1} = i) \quad \forall t. \tag{5.3}$$

The assumption of time-homogeneity is very helpful since it allows us to compute a single transition matrix instead of having to compute a different matrix for each time interval. Thus the empirical transition probabilities are given by

$$\hat{p}_{ij} = \frac{\#\text{ of transitions from } i \text{ to } j}{\#\text{ of transitions from } i}. \tag{5.4}$$

Figure 5.2 illustrates the computation of the transition probabilities in our example. On the left are shown the transitions between groups and on the right the resulting transition matrix. If we consider, e.g., $\hat{p}_{01}$, we see on the left that there are three transitions from group 0, two of which lead to group 1. Hence $\hat{p}_{01} = 2/3$.

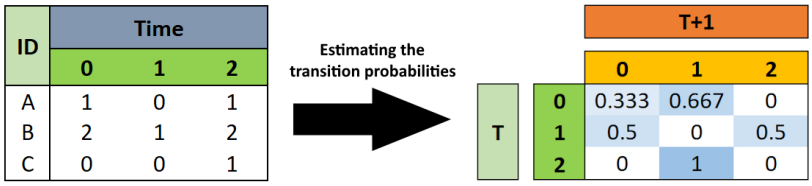


Figure 5.2: Estimating the transition matrix.

Step 3 Compute the CLV by simulating Markov chains with the Monte Carlo method.

In order to compute $CLV(a)$, we simulate multiple paths of $\{S_t\}$ starting from $S_0 = a$ by using the empirical transition matrix. If for the i^{th} path out of N paths the computed CLV is $\widehat{CLV}_i(a)$, then by the Monte Carlo method the estimated CLV is given by the formula

$$\widehat{CLV}(a) = \frac{1}{N} \sum_{i=1}^{N} \widehat{CLV}_i(a). \tag{5.5}$$

5.3 Results

We implement the above methods on synthetic data. The synthetic data has fewer features than the real data. A generalized linear model is given to compute the profit of each client. Only the features necessary for the algorithm are present. Many other features, especially the categorical ones, are omitted in the synthetic data. Each synthetic feature is generated uniformly from the bounds of the corresponding features in the real data set. Here is what we expect to happen.

1. Our implementation of decision tree will give a reasonable number of states.
2. The CLV values of all states will be bounded (within reason).
3. The CLV values of all states will be random and centered around zero.

This is indeed what we observe.

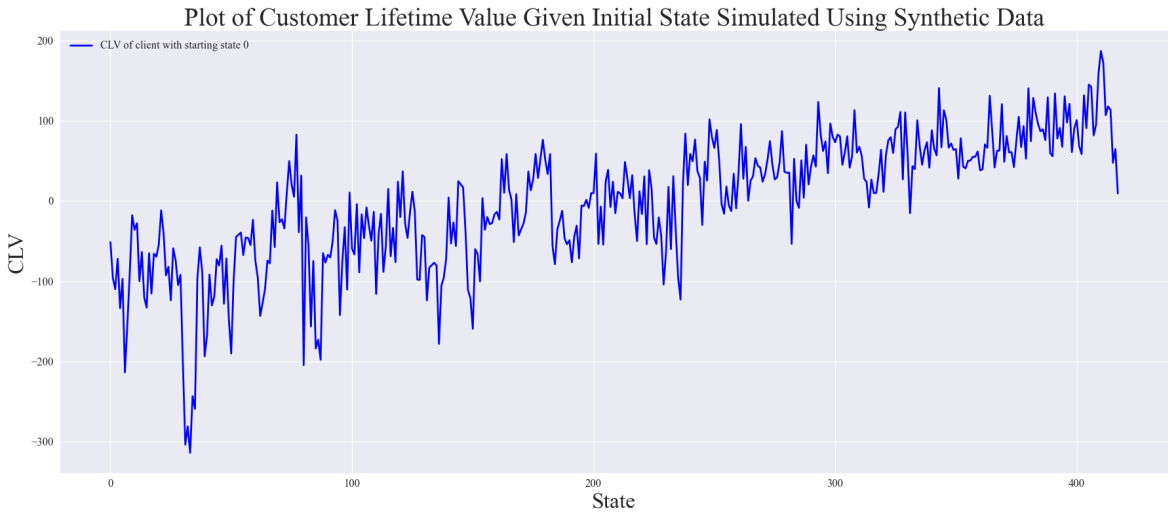


Figure 5.3: Simulation of the CLV.

Attempts were made to apply this method to the real data set, after the success obtained on the synthetic data. Due to programming difficulties, however, cleaning the real data took too much time and we were not able to get results on the real data because of time constraints.

5.4 Discussion

In this section, we look at other approaches to calculating CLV.

The CLV has been extensively studied in both academic research and companies, for marketing purposes. One of the most famous references is that of Gupta [2], who presents methods for modelling the CLV. There are several of these, including the Markov chain approach, which consists first of all in determining the possible customer states using various algorithms, such as the regression tree used here, or the K-means method, and then calculating the CLV using the formula (5.1): we refer to [3, 4] for more details. On the other hand we also have the Pareto/NBD model, a famous model of the CLV. This model assumes that recency and frequency are independent of monetary value and is divided into two sub-models: one for the expected number of transactions, the other for the expected average order value. Multiplying the results gives the CLV. We refer to [3] for the assumptions for the transaction Pareto/NBD sub-model.

Bibliography

[1] Haenlein, M., Kaplan, A. & Beeser, A. (2007). A Model to Determine Customer Lifetime Value in a Retail Banking Context. European Management Journal, 25.

[2] Gupta Sunil, Hanssens Dominique, Hardie Bruce, Kahn Wiliam, Kumar V., Lin Nathaniel, Ravishanker Nalini and Sriram, S. (2006). Modeling customer lifetime value. Journal of service research, 9(2):139–155.

[3] Jasek, P., Vrana, L., Sperkova, L., Smutny, Z., & Kobulsky, M. (2018). Modeling and application of customer lifetime value in online retail. Informatics 5(1).

[4] Jablecka, M. (2020). Modelling CLV in the Insurance Industry Using Deep Learning Methods. Master's Thesis at KTH Royal Institute of Technology.

6 Radio-Canada: Determining the right moment for suggesting the creation of an account

Mario Canche [a]
Andrea Ek [a]
Michael R Lindstrom [b]
Corentin Lonjarret [c]
Patrick Mesana [d]
Carlos Montes [b]
Nicolas Schönau [c]
Omar Sharif [b]
Marziyeh Talebian [e]
Louis Willems [c]

[a] *Centro de Investigación en Matemáticas (CIMAT)*
[b] *UTRGV*
[c] *Radio-Canada*
[d] *HEC Montréal*
[e] *Université Concordia*

March 2024
Les Cahiers du GERAD

6.1 Introduction

Radio-Canada is a Canadian news media organization that provides content for radio, television, and the online world [1]. Radio-Canada supports Canadian democratic values by providing a range of articles, videos, and audio content covering a range of topics. Part of the organization's growth plan is to encourage visitors to the Radio-Canada website to create accounts, enabling a more personalized web experience. At present, however, only a small fraction (less than 1% of website visitors) create accounts.

The problem of conversion is known across the business world [6], where most websites experience far more visitors than customers. Although Radio-Canada is not seeking to gain user accounts for profit, it is still the case that account creation is a significant hurdle.

The Radio-Canada website offers several areas of content, including audio, news, and cooking sections. Users can interact with content in different ways, too. For example, some may read several articles; others only listen to audio; others visit the cooking pages and then listen to audio; and others may read a single article, leave the browser window open, and not interact with the webpage again for a day or more; etc. In order to gain a better understanding of the behaviour of website visitors, Radio-Canada has begun using cookies to track anonymously returning visitors for some duration in order to see what content they engage with, how long they spend on a given page, etc. The use of cookies as part of analytics is common [5] and has helped many websites improve understanding of their visitors.

In the 13th Montreal IPSW, organized by the CRM and IVADO, Radio-Canada proposed a data science problem to the participants [2]. Participants were provided with sequences of events by visitors to the website who did and did not create accounts. Event information included data such as: date and time of day, day of week, the webpage visited, and the type of event (audio, video, etc.). Participants were asked some open questions.

- What sequences of events are associated with imminent account creation?
- Given the current status of a user, what is the probability that he or she will create an account?
- What common sets of features distinguish account-creators from those who do not create accounts?

To handle these questions, our team combined data visualization and clustering to understand groups of users, a logistic regression model to predict the probability a user will create an account as their next event, and an event embedding analysis to identify events correlated with account creation.

Here is the outline of the rest of this paper: a detailed description of our dataset is given in Section 6.2; the approaches and results of data visualization are presented in Section 6.3; the logistic regression model and its results are presented in Section 6.4; the event embedding and analysis results are presented in Section 6.5; and our conclusions are presented in Section 6.6.

6.2 Data

6.2.1 High level

The data are a collection of webpage interactions with timestamps from April-May 2023, including the type of activity, and unique ID per visitor: see Figure 6.1.

Examples of fields include: visit number, page visit number, hit time, browser height/width, whether cookies are enabled, the country of the visitor, the date and time, various representations for the type of the event, etc. The event types could include visiting the news pages, visiting the cooking pages, listening to audio, or combinations of these.

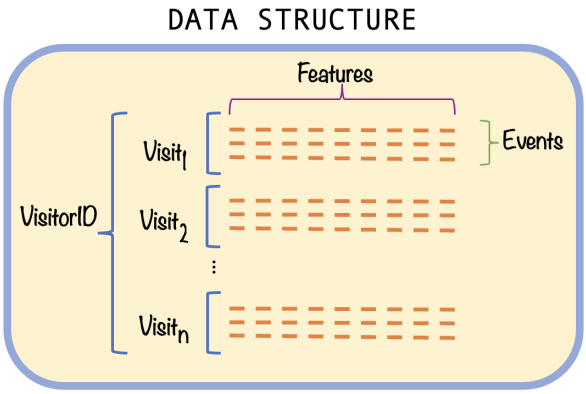


Figure 6.1: Representation of data.

6.2.2 Processing and limitations

Some filtering of the data was necessary, including the filtering out of events of users after they had created accounts (since the objective was to understand behaviour prior to account creation). Some models required filtering out users who had webpage events prior to the observation interval (because their past activity and time on the website cannot be accurately modelled).

A few notable limitations are present in the data set: some users may delete their cookies and appear under different IDs, unique users visiting through multiple devices appear as different visitors, and bots may be in the dataset. In our analysis we ignored these issues and focused on getting a higher-level understanding.

6.3 Data visualization and clustering

Through data visualization and clustering, we seek to identify noteworthy patterns that could give us clues into how and why some users create accounts and when.

One of the first things we note is that there are clear differences over the hours of the day and days of the week for when users create accounts: see Figure 6.2. For example, it seems that Sunday, Monday, and Tuesday are the days having the highest rate of account creation. We also note that as far as hours of the day are concerned, most accounts were created between 1pm and 4pm.

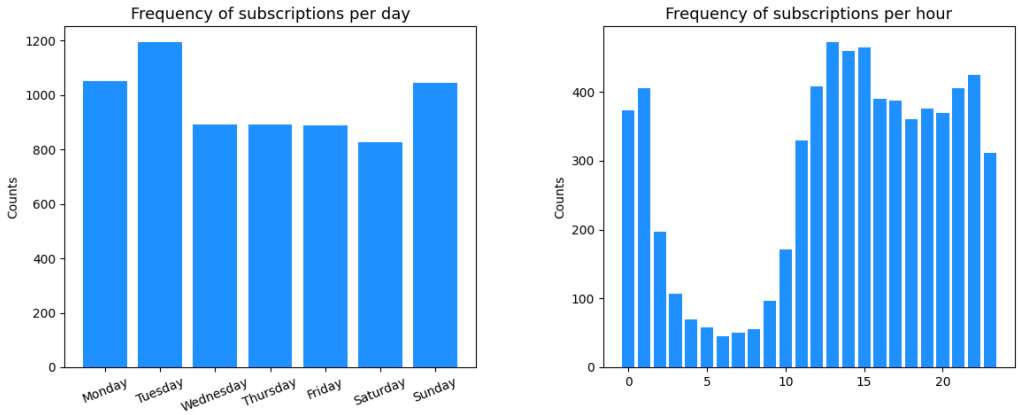


Figure 6.2: The day and hour of account creation.

By taking a sample of users who did and did not create accounts, we can study what sort of webpage events each user group engages in more frequently. Figure 6.3 studies the most recent event type of users and, among all events documented, presents the percentage of those events attributed to both user classes. The graph is not normalized and very few users do register, so seeing very little green (registered) is to be expected. Since there is a relative prevalence of registered users having "Multiple" event types (engaging in more than one of Info (news), Mordu (cooking), and Ohdio (audio)), it seems that users who create accounts engage in more facets of the website.

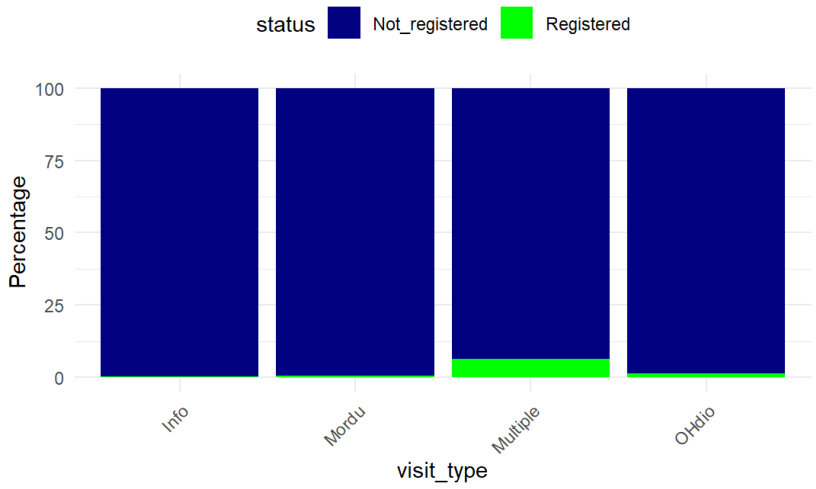


Figure 6.3: Most recent visit type among those who did and did not create accounts.

We work with a subsample of 1300 events for account creators and 530 events for those not creating accounts. Through a simple clustering analysis (Kmeans) [4], we can study whether the fraction of time spent on different actions can help identify which group a user belongs to. For each user we either take his or her entire event history, if the user was not an account-holder, or we take all events they engaged in up to but not including their account creation. We can then examine the fraction of time users spent on different parts of the website. In our sample we have 130 users who did create accounts and 53 users who did not create an account. We then perform Kmeans clustering with 2 clusters on these time fractions and study the ratios of users who do and do not create accounts in each cluster (high/low fractional time). The case for the "Video Info" event is found in Figure 6.4 with cluster sizes of 133 (lower) and 50 (upper). We note that the duration of the final event was filled in as the mean of the user's previous event times. Given the users in each group, we roughly expect a 130:53 ratio of account creation to no account creation users in each cluster. It appears, however, that spending more time on "Video Info" may be associated with a lower rate of account creation.

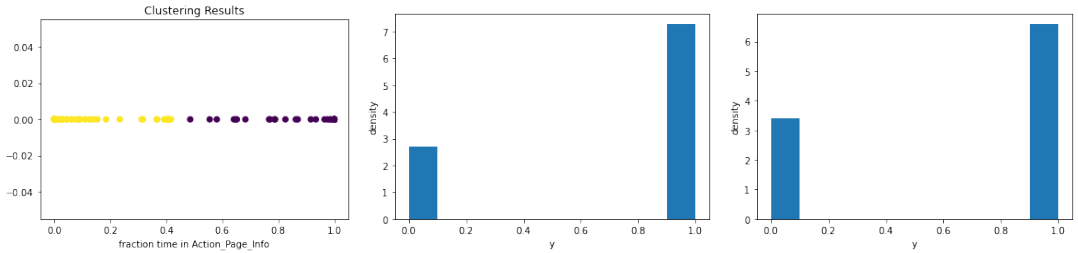


Figure 6.4: Kmeans analysis of "Video Info" events by fraction of time and comparisons of how many went on to create accounts. Left: the Kmeans clustering of total duration. Centre: the fraction of visitors in the lower cluster who did not ($y = 0$) and did ($y = 1$) create accounts. Right: the fraction of visitors in the upper cluster who did not ($y = 0$) and did ($y = 1$) create accounts.

6.4 Logistic regression model

6.4.1 Formulation

We assume that there are U users in the dataset. Each of their events is encoded in a vector $x \in \mathbb{R}^k$ — this encodes information such as time of day, duration of the event, the one-hot-encoded action, etc. For each such x we let $Y_x \in \{0, 1\}$ be a random variable representing a subsequent account creation (0=no, 1=yes).

With logistic regression [3] we seek to evaluate the probability $\Pr(Y_x = 1)$ or $\Pr(Y_x = 0)$. Since not creating an account is more common, we model the latter and assume that $\Pr(Y_x = 0|x) = \sigma(\theta^T \hat{x})$ holds, where $\theta \in \mathbb{R}^{k+1}$ is a parameter, $\hat{x}$ is defined as $(1, x^{(1)}, ..., x^{(k)})^T$, and $\sigma : z \mapsto \frac{1}{1+\exp(-z)}$ is the sigmoid function.

We now look at the overall dataset. For user i, let N_i be either the number of interactions leading up to but not including account creation (if the user does create one), or their total number of interactions otherwise. Let $x_{ij} \in \mathbb{R}^k$ be the jth interaction of user i. Then the dataset is $\mathcal{D}|_x = \{\{y_{ij}\}_{j=1}^{N_i}\}_{i=1}^{U}$, a collection of creation statuses with y_{ij} the realization of $Y_{x_{ij}}$. If *each user event is treated as independently leading to creating/not creating an account* then the log-likelihood is

$$\mathcal{L}(\mathcal{D}|_x|\theta) = \sum_{i=1}^{U} \mathbb{I}_{y_{i,N_i}=1} \left(\sum_{j=1}^{N_i-1} \log \sigma(\theta^T \hat{x}_{ij}) + \log\left(1 - \sigma(\theta^T \hat{x}_{i,N_i})\right) \right) + \sum_{i=1}^{U} \mathbb{I}_{y_{i,N_i}=0} \left(\sum_{j=1}^{N_i} \log \sigma(\theta^T \hat{x}_{ij}) \right),$$

where $\mathbb{I}$ is the indicator function. This can be maximized with gradient ascent.

We note that there is a loose coupling between events, because data such as the cumulative time spent on the website may be used and could increase from one event to the next.

Due to the large imbalances in the dataset, our best results are obtained when we attempt to balance the data. The approach we take in our results amounts to studying only the users who do create accounts. If such a user had N events up to but not including account creation then that yields $N - 1$ events resulting in no account creation and 1 event resulting in an account creation.

6.4.2 Results

Looking at θ gives insight into user account creation patterns and probabilities of account creation. Table 6.1 provides details of θ for one fit (the optimal θ found from 50 gradient ascents for different random initial values). When regression coefficients of θ are positive, it means the corresponding features have a negative impact in account creation (account creation is predicted to be less probable), and the fact that the coefficients are negative means that the corresponding features have a positive impact on account creation (account creation is predicted to be more probable). Thus the model *suggests* (causal mechanisms cannot be inferred) that on Tuesdays, Wednesdays, and Thursdays, users are more likely to create accounts; that the shorter the duration of each event, but the more visits a user has, the more likely a user will create an account; that disabling cookies means a user is less likely to create an account, but having them enabled means they are more likely to create an account; and that an event such as "detailed_event_Action_autre" makes a user less likely to create an account, whereas an event such as "detailed_event_Video_Info" makes a user more likely to create an account.

Based on our "balanced" dataset approach, a sample confusion matrix and summary table are given in Figure 6.5. These results depict the predictive power of the model found in Table 6.1. There are a lot more cases of not creating an account than account creations, even in this balanced set. The most

Table 6.1: Logistic regression analysis coefficients. Positive coefficients make a user less likely to create an account, negative coefficients make a user more likely to create an account.

θ component	value
intercept	-0.0331
day_of_visit_4	-0.0474
day_of_visit_2	-0.0284
day_of_visit_3	-0.0102
day_of_visit_6	0.0005
day_of_visit_5	0.0023
day_of_visit_7	0.0335
day_of_visit_1	0.0421
duration	0.0009
num_visits	-0.0236
cookies_Y	-0.0415
cookies_U	-0.0396
cookies_N	0.0388
detailed_event_Action_autre	-0.0205
detailed_event_Video_Mordu	-0.0081
detailed_event_Page_Info	0.0044
detailed_event_Écoute_OHdio	0.0101
detailed_event_Page_accueil_Info	0.0106
detailed_event_Page_accueil_OHdio	0.0277
detailed_event_Page_accueil_Mordu	0.0338
detailed_event_Page_OHdio	0.0345
detailed_event_Video_Info	0.0367
detailed_event_Page_Mordu	0.0398

noteworthy observation may be that the model has a high recall score for account creation: 49% of the time, it identifies users whose next action is to create an account — despite the ratio of 542:49 for negative:positive events.

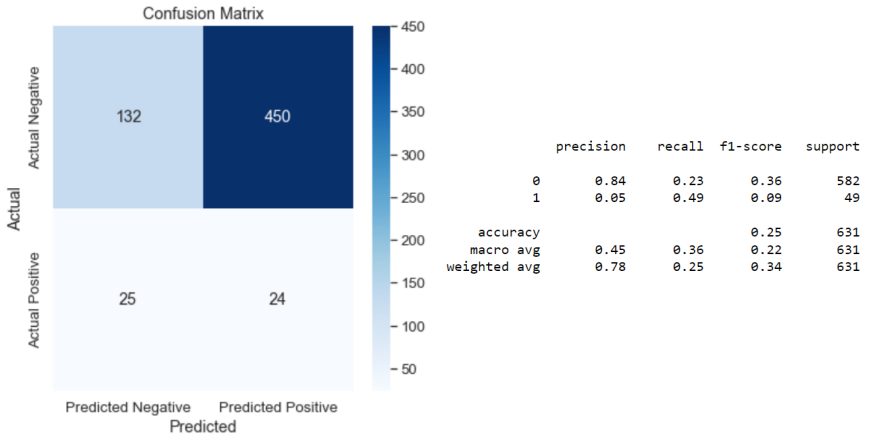


Figure 6.5: The confusion matrix and summary table for the logistic regression model.

6.5 Events embedding analysis

6.5.1 Introduction

To begin to understand the event sequence that leads users to create an account, we first asked what the immediate context of an account creation event is. Insights from Radio-Canada revealed that when a user wants to add OHdio content to their favourite list, the website immediately prompts them to create an account. This information is found in the `"post_evar102"` (a field provided by the analytics

software) column of an action event. It was confirmed that a significant proportion of account creations come from users attempting to add OHdio content to their favourites.

We select all the account creation events to see how many accounts were created following these navigation paths. On Figure 6.6, we see the distribution of `"post_evar102"` column for all the creation events of type OHdio. It is clear that many users who create accounts are first trying to access features that only users with accounts can use, such as “Ma liste”, “Mon compte” and “Mes favoris”.

Unfortunately, that analysis cannot be repeated for all account creation events since `"post_evar102"` values are often not present or are not interpretable. We thus look at previous events to infer some context in the creation of an account. The problem is that the previous event is not necessarily a good representation of the context (the data are also very noisy). The good news is we have a lot of data so that we can represent the events by their statistical proximity. Our idea is to utilize a representation learning technique called skip-gram with negative sampling. This technique is used in Word2Vec [7], a model that represents words in a vector space, enabling arithmetic operations to be performed with words.

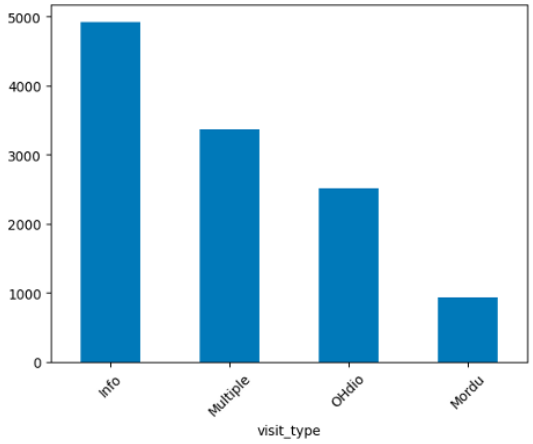


Figure 6.6: Distribution of the field “event-type” of account creation events.

6.5.2 Skip-gram Model for Graph Nodes

In the context of a website navigation graph, each node represents a specific web event, and each directed edge signifies that a user navigated from one page to another. The objective is to learn low-dimensional embeddings for each node such that the likelihood of observed sequences of navigation is maximized. The average log probability is given by

$$J = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p\left(n_{t+j}|n_t\right).$$

In this formula T denotes the total number of navigation events, m the window size representing a user’s navigation session, n_t the target node (current web event), and n_{t+j} a node within the sequence around n_t. In turn the p’s are given by

$$p(n_{t+j}|n_t) = \frac{\exp(v'_{n_{t+j}} \cdot v_{n_t})}{\sum_{n=1}^{N} \exp(v'_n \cdot v_{n_t})}.$$

In this equation v_n and v'_n are the “input” and “output” vectors for a node n and N is the total number of nodes in the graph.

6.5.3 Negative sampling for graph nodes

Calculating the Softmax function over all nodes can be computationally expensive, especially for large graphs. Negative Sampling is used to simplify the objective function by converting it into a binary classification problem [7]. The negative sampling objective function is given by replacing $p(v_{n_{\text{OUT}}}|v_{n_{\text{IN}}})$ of the Skip-gram model by

$$\log \sigma(v'_{n_{\text{OUT}}} \cdot v_{n_{\text{IN}}}) + \sum_{i=1}^{k} \mathbb{E}_{n_i \sim P_n(n)} \left[\log \sigma(-v'_{n_i} \cdot v_{n_{\text{IN}}})\right] .$$

Here $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function, $v'_{n_{\text{OUT}}}$ the output vector of the target node, $v_{n_{\text{IN}}}$ the input vector of the given node, k the number of negative samples, and $P_n(n)$ a noise distribution from which negative samples are drawn. Optimizing this objective corresponds to distinguishing a target word from noise through logistic regression.

Negative sampling helps use tackle the challenge of potentially millions of web navigation events. The technique substantially reduces the parameter space, thereby mitigating computational burdens inherent in such expansive event datasets. Within the context of event data, consider a specific target event and an associated context event that might precede or follow it in a user's navigation sequence. Rather than directly predicting the exact subsequent event using a softmax function over the immense catalog of events, the model is instead trained to determine whether pairs of events typically co-occur within the same navigational context. The rationale behind this sampling strategy is anchored in principles similar to those elucidated in the foundational Word2Vec publication.

6.5.4 Training the model

We trained the skip-gram model with negative sampling using the following hyperparameters: embedding dimension of 50, learning rate of 0.001, 100 epochs, and a batch size of 64. The model was trained using the ADAM optimizer: see Figures 6.7 and 6.8.

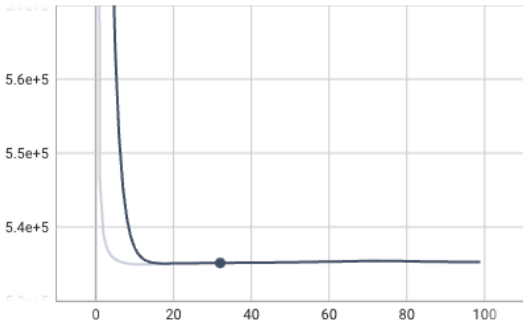


Figure 6.7: Training Loss.

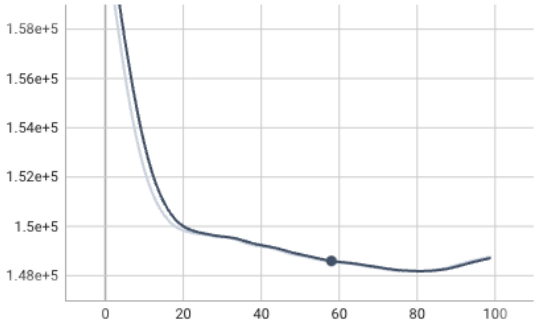


Figure 6.8: Validation Loss.

6.5.5 Exploring the events embedding

To glean a deeper understanding of the relationships between events leading to account creation, we resorted to a visual exploration of the learned embeddings. Leveraging the capabilities of TensorBoard, a popular tool for visualizing the intricacies of deep learning models, we transformed the high-dimensional event vectors using the t-SNE dimensionality reduction technique. This method is particularly adept at preserving the local structure of data in lower-dimensional spaces, making it suitable for visual inspections.

In the first visualization (refer to Figure 6.9), the entirety of the embedding space is depicted with account creation events highlighted. This overarching view provides a macroscopic understanding of

event proximities. On the other hand, to discern the nuanced relationships between specific events, we narrow our focus on a subset of the space. The second visualization zooms in on the "Mordu" events (see Figure 6.10). Notably, events close to the "Mordu" segment predominantly consist of activities associated with the "Mordu" section of the website. This observation aligns with our expectations and strengthens our belief in the model's ability to position similar navigating events near one another in the embedding space.

Figure 6.9: t-SNE visualization of the entire events embedding with account creation events highlighted.

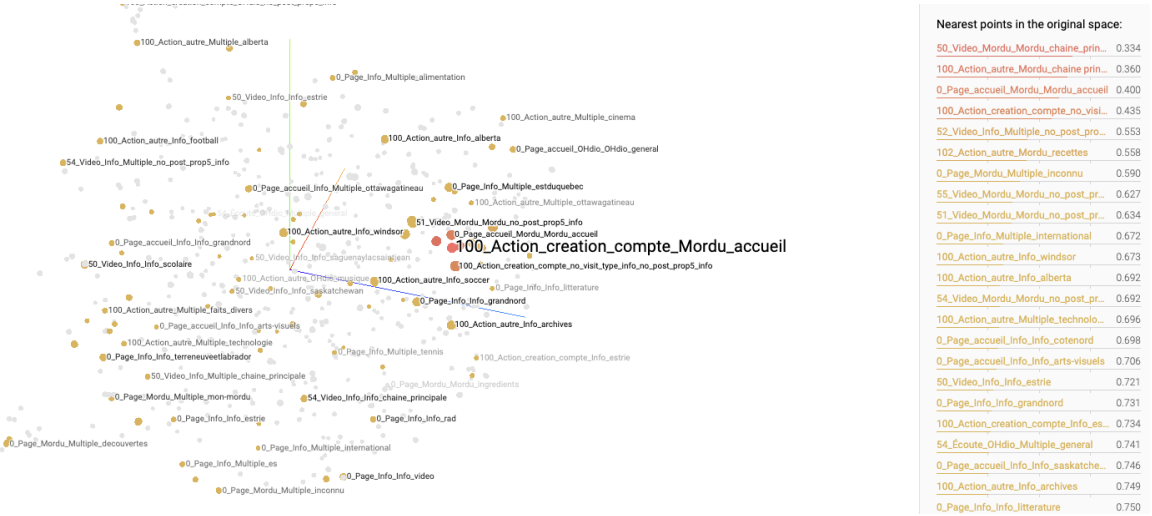


Figure 6.10: Focused t-SNE visualization centering on "Mordu" events and their vicinity.

6.5.6 Summary

In the context of website navigation:

- Each web event (node) is represented by a vector;
- These vectors are updated to maximize the likelihood of predicting the sequence of pages in a user's navigation, given a current page;
- The computation is made efficient through Negative Sampling.

By using the Skip-gram model with Negative Sampling on website navigation data, we can learn meaningful embeddings for each web event. These embeddings can capture user navigation patterns and may be used for various tasks like clustering, predicting specific events, or simply understanding user behaviour.

6.6 Conclusion

From our work, we find that (1) Sunday, Monday, and Tuesday are the days where more accounts have been created; (2) account-holders seem to be more often engaged in visits with multiple event types; (3) the probability of account creation can be modelled, suggesting: Tuesday, Wednesday, and Thursday are the best days for account creation; visitors who enable cookies are more likely to create accounts; the longer each event takes, the less likely a user is to create an account; but the more a given user visits the website, the more likely this user is to create an account; and (4) events can be embedded in a vector space and used in a predictive model for other events for each user.

As future directions, we consider using Long Short Term (LSTM) neural networks to model the sequential problem with the numerical variables, finding better features for the logistic regression, adding more feature navigation levels to increase the number of events, and identifying common trends and interpretations between the complementary models.

Acknowledgments

We would like to thank Radio-Canada and its representatives for proposing this problem and are grateful for their availability throughout the workshop.

Bibliography

[1] Radio-Canada. https://ici.radio-canada.ca/. Last accessed: 03/10/2023.

[2] Thirteenth Montreal Industrial Problem Solving Workshop. https://www.crmath.ca/en/activities/#/type/activity/id/3891. Last accessed: 03/10/2023.

[3] DeMaris, A. (1995). A tutorial in logistic regression. Journal of Marriage and the Family, pages 956–968.

[4] Likas, A., Vlassis, N., and Verbeek, J. J. (2003). The global k-means clustering algorithm. Pattern recognition, 36(2):451–461.

[5] Loftus, W. (2012). Demonstrating success: Web analytics and continuous improvement. Journal of Web Librarianship, 6(1):45–55.

[6] McDowell, W. C., Wilson, R. C., and Kile Jr, C. O. (2016). An examination of retail website design and conversion rate. Journal of Business Research, 69(11):4837–4842.

[7] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 26.