POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Quantifying Uncertainty in Systems - Two Practical Use Cases Using Machine Learning to Predict and Explain Systems Failures

GABRIELLE GAUTHIER MELANÇON
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Mathématiques appliquées

Décembre 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Quantifying Uncertainty in Systems - Two Practical Use Cases Using Machine Learning to Predict and Explain Systems Failures

présenté par **Gabrielle GAUTHIER MELANÇON**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

DEDICATION

"Some of the biggest problems facing the world
— war, hunger, poverty, and environmental degradation —
are essentially system failures.

They cannot be solved by fixing one piece in isolation from the others,
because even seemingly minor details have enormous power
to undermine the best efforts of too-narrow thinking."

- Donella Meadows [1]

ACKNOWLEDGEMENTS

Merci à mes merveilleux collègues et amis à JDA, Element AI et Polytechnique. Je n'en serais pas là aujourd'hui sans ce réseau d'aide incroyable qui m'a permis d'apprendre et me dépasser. Je suis tellement reconnaissante de l'apport de chacun d'entre vous dans l'accomplissement de ce travail. Mes remerciements sont longs et ce n'est pas pour rien.

Merci à JDA, Element AI et Michelin d'avoir accepté que je puisse publier les résultats de mes expériences et de m'avoir alloué du temps pour mes cours et la rédaction. En particulier, merci à Suresh Acharya, Martin Bernier et Alexei Nordell Markovits pour leur continuel soutien et support à l'interne afin que je puisse concilier travail et études. Merci d'avoir cru en moi et de m'avoir supportée dans ce retour aux études.

Un merci tout spécial à Philippe Grangier pour ta contribution, ta patience, tes précieux conseils et tes encouragements. Tu as été une personne clé pour moi du début à la fin. Merci aussi à Éric Prescott-Gagnon et à Emmanuel Sabourin. Nous formions une super équipe.

Un énorme merci à tout ceux à Élement AI qui m'ont aidée, que ce soit avec les innombrables discussions sur l'incertitude, les bugs latex, les embeddings, les re-lectures, ou encore les encouragements, le support et même l'hébergement. Merci Xavier Snelgrove, Orlando Marquez, Waseem Gharbieh, Iman Malik, Jeremy Barnes, Nicolas Chapados, Misha Benjamin, Marie-Claude Côté, Frédéric Branchaud, Lorne Schell, Émilie Perrault, Parmida Atighehchian, Lindsay Brin, Archy de Berker, Claudel Rheault, Jonny Linton, Éric Laufer, Ayushman Dash, Isabelle Bouchard, Tara Tressel ainsi que tous les autres.

Merci à mes amis et mes parents qui m'ont aussi soutenue dans cette aventure, et même avec l'anglais. Vos messages d'encouragement m'ont toujours fait chaud au coeur. Pour en nommer que quelques-uns, merci à Nic, Kat, Maryse, Vince, Caro... Un merci tout particulier à Kev, qui m'a encouragée et inspirée à poursuivre dans cette voix.

Pour terminer, merci aux professeurs qui ont été là pour me guider et m'enseigner. Merci à Charles Audet, pour m'avoir acceptée dans son bureau chaque semaine pour les premiers mois afin que je ré-apprenne toutes les notions mathématiques que j'avais oublié, et à en découvrir de nouvelles. Mais surtout, un énorme merci à mon directeur de recherche, Louis-Martin Rousseau, sans qui je ne me serais jamais lancée dans cette aventure. Merci d'avoir cru en mon potentiel et de m'avoir soutenue à travers toutes ces années. Les mots me manquent pour exprimer toute ma gratitude. Merci à tous!

RÉSUMÉ

Afin d'accomplir et automatiser des tâches complexes, différents modèles et composantes peuvent être organisés sous forme d'un système. Par exemple, les chaînes d'approvisionnement sont des systèmes qui permettent de transformer et transporter des matières premières en produits finis. Ces systèmes sont primordiaux lorsque la complexité d'un processus est telle qu'elle ne pourrait être gérée manuellement par un seul humain. Cependant, différentes sources d'incertitude peuvent affecter le comportement d'un système et son résultat. Cela peut mener à ce que l'on désigne comme une défaillance du système, c'est-à-dire lorsqu'un système ne produit pas un résultat désiré. Ce mémoire présente et compare deux projets où l'on utilise l'apprentissage machine afin de prédire les défaillances dans les systèmes, en modélisant ces sources d'incertitude avec une approche holistique.

Dans le premier projet, pour lequel nous avons soumis un papier au INFORMS Journal on Applied Analytics, nous cherchons à prédire les défaillances dans les chaines d'approvisionnement. Nous avons développé notre approche à JDA, un fournisseur de logiciels en chaînes d'approvisionnement, et en coopération avec Michelin, un manufacturier international de pneus. Dans le cadre du projet, nous avons défini une défaillance du système comme étant une baisse du niveau de service en dessous d'un seuil désiré. Nous avons modélisé les différents risques présents dans la chaîne d'approvisionnement en fonction de leur habilité à prédire les défaillances quelques semaines à l'avance. Nous démontrons que notre modèle d'apprentissage automatique donne de bons résultats sur un large éventail de produits. Nous soulignons également l'importance de considérer dans l'approche la façon dont les utilisateurs interagissent avec les prédictions. Pour cela, nous avons développé un outil interactif afin que les utilisateurs puissent naviguer à travers les prédictions et obtenir le contexte nécessaire afin de comprendre la source de la défaillance. Cet outil a permis de convertir les prédictions en actions concrètes afin de prévenir et éliminer les défaillances avant qu'elles ne se produisent. Dans nos résultats, nous présentons également comment l'outil a permis d'identifier des causes récurrentes de défaillances dans leur chaîne d'approvisionnement.

Dans le cadre du second projet, nous présentons nos résultats préliminaires effectués à Element AI, un fournisseur de logiciels en intelligence artificielle. Le projet consiste à quantifier la confiance associée à chaque prédiction générée par un système d'intelligence artificielle. Le système en question est composé d'une séquence de modèles, incluant deux algorithmes d'apprentissage profond et quelques heuristiques. L'objectif du système est d'extraire automatiquement de l'information contenue sur un document. Dans une approche similaire au premier projet, nous avons identifié les sources principales d'incertitude dans le système, telle de l'incertitude aléatorique et épistémique, pour ensuite les modéliser avec une approche d'apprentissage machine afin de déterminer la confiance du système dans chacune de ses prédictions. L'approche est de ce fait inverse à notre premier projet, puisque nous quantifions la possibilité que le système ne soit pas en état de défaillance. En pratique, cela revient à la même tâche. Nos résultats préliminaires suggèrent que notre approche est prometteuse.

Pour nos deux approches, nous prédisons les défaillances d'un système dans le même but, celui d'identifier les situations incertaines où un humain-dans-la-boucle (human-in-the-loop) peut intervenir afin de rectifier le résultat du système et ainsi éviter la défaillance. Cela permet de mettre à profit la créativité et la connaissance de l'humain pour les situations plus délicates, et ainsi augmenter la performance du système. Dans le contexte des chaînes d'approvisionnement, cela peut vouloir dire d'ignorer les paramètres du plan afin de déplacer des items au bon endroit ou encore augmenter manuellement la production. Dans un système d'intelligence artificielle, l'humain peut directement changer la prédiction du système en regardant par lui-même les données, dans notre cas lire le document et extraire lui même l'information. Nous soulignons aussi comment notre approche est générique et peut s'adapter à divers types de systèmes, avec différentes granularités et sources d'incertitude. Nous discutons également comment le modèle utilisé se doit d'être explicable, afin que les utilisateurs puissent identifier et potentiellement éliminer les causes récurrentes de défaillances. Pour terminer, notre approche permet aussi d'augmenter la confiance de l'utilisateur dans le système utilisé ainsi que l'automatisation qu'il permet.

ABSTRACT

Systems are widely used to execute complex tasks, such as supply chain systems, which transform and transport raw materials into finished goods. We often rely on systems to automate, at least partially, a process that would be unmanageable or not-efficient for humans to tackle manually. However, different sources of uncertainty can affect systems' outcomes and behavior, and eventually lead to system failure. The definition of failure is dependent on the context, and it represents an outcome of the system that we want to avoid. This thesis presents and compares two different tracks of work where we explored how to predict system failures, using machine learning, by modeling its sources of uncertainty holistically.

In the first project, for which we submitted a paper to the INFORMS Journal on Applied Analytics, we predict failures in a supply chain system. We developed our approach at JDA, a supply chain software provider, in co-operation with Michelin, an international tires manufacturer. In our approach, we modeled risks with features from different segments of the supply chain that could be good indicators of a future failure. We defined failure as a drop in the service level below the desired target. We showcase that our machine learning model works well on a large variety of products. Most importantly, we highlight the importance of considering in the approach how users interact with the models' predictions. We designed and built an interactive tool for users so they could navigate through the predictions, get the necessary context, and receive an explanation as to the cause of the failure. The tool allowed for predictions to become actionable, so users could act and avoid system failures. We also present in our results how planners identified recurrent issues in their supply chains.

In the second use case, we present early experiments that we have done at Element AI, an artificial intelligence software provider, on quantifying the confidence of an AI system in its prediction. The system is composed of a sequence of models, including deep learning algorithms, and aims at extracting text from documents. In a similar idea to the first project, we modeled the primary sources of uncertainty of the system, such as aleatoric and epistemic uncertainty, and used a machine learning model to measure confidence estimates in each AI system's predictions. Our early results on a given dataset suggest that our method is promising.

In both approaches, we predict system failure for a similar purpose: identifying uncertain situations where a human-in-the-loop can rectify the system's outcome to avoid failure. Indeed, by leveraging a human's creativity and knowledge in complex cases, we can boost the system's performance. In supply chains, this could mean overriding the system's parameters to

change the location of some stock or increase the production. In the AI system use case, this could mean to change the predicted text to its correct value manually. We also highlight how our approach to system failure prediction is generic and flexible to various types of systems, with different granularity, failure definition, and uncertainty sources. We further underscore how the model needs to be explainable so that users can identify and potentially eliminate recurrent causes of system failure. Lastly, failures prediction also help to build general trust in the system and the automation it is providing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

AI	Artificial Intelligence
XAI	Explainable Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
OR	Operational Research
HITL	Human-in-the-loop
OOD	Out-Of-Distribution
GBDT	Gradient Boosted Decision Tree
SVM	Support-vector Machine
SCRM	Supply Chain Risk Management
DC	Distribution Center
FE	Form Extractor
FSU	Full-Stack Uncertainty Model
CV	Cross Validation
OCR	Optical Character Recognition
API	Application Programming Interface
NLL	Negative Log-Likelihood
ECE	Expected Calibration Error
AUROC	Area Under the Receiver Operating Characteristic
PPV	Positive Predictive Value
NPV	Negative Predictive Value
GPU	Graphics Processing Unit

UI	User Interface
KPI	Key Performance Indicator

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Definitions and concepts

In this thesis, we explore the usage of Machine Learning (ML) for system failure predictions by modeling a system's primary sources of uncertainty. In this section, we first present what is a system, drawing from Meadows [1]. In her book, *Thinking in Systems*, the author discusses how to define and identify systems around us and explore why they work so well. We use some of her concepts and terminologies throughout this thesis. Subsequently, we explore the concept of uncertainty and the importance of quantifying it to predict systems' behavior. This will lay the foundation on which we will present our research on utilizing ML to predict failures in two different types of systems, supply chains, and Artificial Intelligence (AI) systems.

1.1.1 What is a system?

Meadows [1] defines a system as "an interconnected set of elements that is coherently organized in a way that achieves something". Systems surround us in our everyday lives - heating systems, trees, human bodies, ecosystems, supply chains - and even though these concepts appear quite different, they all share interesting systems properties. First, systems contain elements that communicate and interact with each other through interconnections. Elements are the parts of the system that are usually tangible. As an example, a human body has elements such as lungs, stomach, and legs. Interconnections can be physical flows, such as blood in the veins, or flows of information, such as signals in our nervous system. Secondly, every system has a purpose, which corresponds to its behavior that persists through time. We could argue that the purpose of our body is to stay alive. Thirdly, systems have this particularity where it is challenging to draw its boundaries so that the main factors influencing its behavior are considered. As an example, the human body's behavior is influenced by its surroundings, such as the ambient temperature or the level of stress occasioned by its current situation. It cannot be understood in isolation from these variables. Additionally, elements in a system can be repeatedly broken down in sub-elements, and systems can even be nested in systems. Every organ in the human body can be seen as a system. Ultimately, organs can be decomposed in cells which also belong to a system. If the system boundaries and granularity level are not carefully chosen, it can create misunderstandings or surprises when analyzing its behavior. Essential elements or interconnections should not be left out.

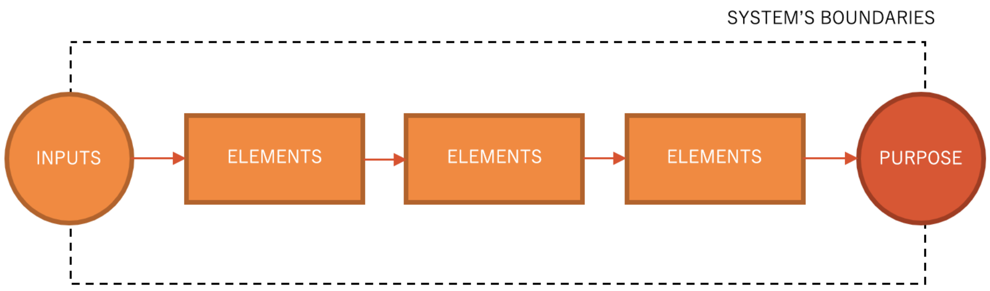


Figure 1.1 A system is composed of interconnected elements which together achieves a purpose.

As we analyze a system's behavior over time, some of its elements accumulate. Meadows [1] categorizes any material or information that builds up over time as stock - such as food (energy) in our stomach or air in our lungs. Stock levels change because of flows. Eating or burning calories are flows which increase or decrease the amount of energy in our system. Stock is a fundamental notion in systems since it allows the inputs and outputs of the system to be momentarily independent and unbalanced. Hence, they act as buffers, or shock absorbers, allowing the whole system to adapt in uncertain environments. When the amount of stock varies in a system, it can trigger a feedback loop. This last concept is key to increase systems resiliency. Feedback loops activate flows to compensate a change in stock. Our body contains numerous feedback loops that allowed us to survive without needing to consciously and continuously think about eating and breathing. Not all systems have feedback loops, but systems that can achieve their purpose in evolving conditions typically have them.

Systems are everywhere and rightly so - they work very well. Their structure allows them to be resilient and self-organized; they can adapt, learn, and evolve in variable environments. Systems are also useful and simple in their design, which allow reducing the amount of information stored in each part of the system. Each element can be self-contained and can even often be replaced without impacting the system's behavior significantly. It helps to reduce complexity for systems achieving difficult and complicated tasks. On the other end, systems' structure also brings challenges. Interactions between elements can result in unexpected outcomes. Each system sub-part has its purpose, and they do not always add up to the desired system's purpose. This complexity makes it hard to study and predict system behavior.

With such broad concepts, many concepts surrounding us can be seen as systems. Throughout this thesis, we explore more specifically systems where at least one of its elements is either an algorithm or a machine. These systems are typically used to automate a complex task. However, many ideas explored could apply to more generic cases too.

1.1.2 What is uncertainty?

Uncertainty is a state in which we cannot describe a current situation or future outcome. In the literature, uncertainty is typically attributed to either randomness of events or incomplete/unknown information. Nevertheless, some authors consider them the same, arguing that randomness is a form of unknowledge [3]. It is common to refer to those two types of uncertainty as aleatoric vs. epistemic uncertainty [4]. Aleatoric uncertainty, derived from *alea* in Latin - a rolling of dice, includes unavoidable unpredictability, or chance, where we cannot know the outcome. A flip of a coin is aleatoric. Epistemic uncertainty, derived from *epistēmē* in greek - knowledge, refers, as its name suggests, to uncertainty due to a lack of knowledge. In this case, we do not know the outcome, but we could with more information. An example would be where we flip a coin but hide its result. Taleb [3] defines epistemic arrogance as this tendency that we have as humans to underestimate what we do not know and overestimate what we know.

In the ML community, aleatoric uncertainty is usually referred to as data uncertainty and accounts for the noise in the dataset. It can be sensor noise or randomness in the data generation process, such as a dice throw. Even as more data is collected, this type of uncertainty cannot be reduced. Epistemic uncertainty, on the other end, is usually identified as model uncertainty and can be reduced by collecting more data. It includes uncertainty around the model's parameters, including the choice of model [5].

Various terms and notions gravitates around the notion of uncertainty, such as *risks*, *confidence*, *failure*, and such. While it is beyond the scope of this thesis to define these terms in details and how they relate to one another, it is important to discuss how we refer to them in the next sections. *Risks*, in some fields such as finance, is defined as the estimated variance of an expected return [6]. It can hence have a positive or negative connotation. However, in other fields such as Supply Chain Risk Management (SCRM), risks are usually defined as being a subcategory of uncertainty where the outcome is undesirable. *Confidence* can also have various interpretations, and as part of this work, we consider it loosely as being the opposite of uncertainty. Lastly, *failure* is defined as being the state in which the desired outcome is not met, usually viewed as the opposite of success. Uncertainty is a likely cause of failures.

1.2 The Problematic

In this section, we discuss how uncertainty can impact system behavior and lead to system failure. We present how uncertainty quantification can be used in systems to alleviate the impact of different sources of uncertainty.

1.2.1 Uncertainty in Systems

Systems are inherently complex and they function in uncertain and evolving environments. Inevitably, systems do not always behave as expected. Even the most resilient systems are not sheltered from failure. Different sources of uncertainty can cause system failures. The most obvious case is uncertainty in the environment surrounding the system that can affect its input and operations. It could be noise in the data or operating conditions rarely observed in the past. In addition, epistemic uncertainty can be present within the system in the form of improperly configured feedback loops, poor interactions between sub-parts of the system, or epistemic uncertainty in the elements themselves. Some components can also have inherent randomness and produce aleatoric outcomes. All these different sources of uncertainty propagate and interact with one another in complex and non-linear ways, thus affecting the system's behavior.

1.2.2 Understanding System Behavior

To understand the impact of uncertainty, one must analyze the system holistically and not only look at its sub-parts in isolation [7]. However, the challenge lies in how systems are typically built; elements are often designed and tested in isolation. Rahwan et al. [8] discuss how AI machine behavior needs to be studied at the individual, collective, and hybrid human-machine scale. The study at the individual level, that is, at the algorithm itself, is often done by the scientists and engineers who design and develop the machine. Understanding machine behavior at the collective and hybrid levels, respectively the study of the interaction between machines and the study of interactions between machines and humans, are much more difficult. Yet, they are crucial in order to benefit from AI systems while minimizing or avoiding harm.

1.2.3 Uncertainty Quantification and Explainable AI

As part of this work, we define uncertainty quantification as the task of quantifying how likely uncertainty can cause a system to fail at any given time. In AI, the field of quantifying uncertainty in ML models is often tied, or even included, in the field of Explainable Artificial

Intelligence (XAI). This area of research became popular in the last few years, due to the proliferation of AI methods deemed too opaque or hard to understand. In the AI community, there is no clear agreement on the definition of XAI [9]. For instance, the field can be referred as explainability, interpretability or even understandability. Quantifying the uncertainty of an AI system can be categorized as a form of XAI, as it gives more visibility to the underlying models and their limitations.

1.2.4 Why Uncertainty Quantification is Important

Uncertainty quantification is important for preventing system failures, identifying system improvements, building trust, and adding robustness to any downstream task.

Avoiding System Failure

First, the primary motivation behind predicting when a system is likely to fail is to prevent failure from happening. One typical way to enable this is to involve a Human-in-the-loop (HITL) in complex and rare situations, to intervene and make the system perform adequately. As humans are creative, knowledgable, and learning beings, they can leverage more information and perform more actions than the system naturally possess. By identifying situations at risk for human review, uncertainty quantification becomes a way to increase the resiliency of the system. From this perspective, the human becomes a feedback loop in the system, boosting the system's performance by intervening in uncertain cases where the system's stock is not adequate for the task at hand. In supply chains, this may be a situation where supply or product stock is low as compared to the demand. In AI systems, this may be a situation where the amount of knowledge accumulated at training time (in this case, the stock - knowledge - is intangible) is not sufficient to make a confident prediction.

System Improvement

Second, by repeatedly identifying situations where the system is failing, uncertainty quantification can indirectly help to detect the most common causes of failure and trigger their abolition. An example of this in AI would be using active learning to identify data that should be labeled in priority to increase the model's knowledge. It could also be about identifying one feedback loop which is continuously not working correctly, such as incorrect safety stock targets in supply chains. Therefore, uncertainty quantification becomes a way to explore and address common failure patterns, improving the system substantially.

Trust

Third, systems' users need consistency and reliability so that trust can be gained. Lee and See [10], in a paper on trust in automation, define misuse as failures due to users relying on automation inappropriately, and disuse as failures due to users rejecting the capabilities of automation. If uncertainty is miscommunicated to users, there is a risk in both.

Robustness of Downstream Tasks

Lastly, uncertainty quantification is also useful when the system's outcome is the input to another system. The system's uncertainty becomes a source of uncertainty for the subsequent systems. A typical example of this is when a ML model predictions feed in a Operational Research (OR) model, such as forecasts in supply chain inventory optimization models.

1.3 Research Objectives

Throughout this research, we use a ML model to predict failures in two different systems. We will discuss how our approach can be very generic and applicable to different contexts. In both use cases, the objective of the approach is to focus users' attention on situations at risk where an action can be performed to omit a failure and improve the system's performance. In our approach, the definition of failure is flexible and can be adapted to the users' context. We will present how we consider that useful alerts concern situations that are actionable, significant, and exclusive. We will also show the importance of building an explainable model, so users can have the right tools to improve the avoid the alerts, improved the underlying system, and gain trust.

One minor distinction in our approach with the two use cases lies in what the model predicts. In the first use case, the model outputs the likelihood of system failure, also referred to as uncertainty estimates. In the second use case, the model outputs the opposite, i.e., confidence estimates in the system not failing. In practice, predicting one or the other generates the same outcome.

1.3.1 Supply Chain Use Case

In the first use case, we explore a supply chain system, which purpose is to provide the right amount of products to the right location at the right time. A supply chain is a system with delays, as products usually cannot be produced and delivered instantaneously. These delays force supply chain planners to use some foresight to deliver their products in a timely and costly manner. Inventory throughout the supply chain acts as a buffer for situations such as uncertainty in the processes, demand, or supply. Because of these delays, predicting a supply chain failure in real-time does not allow for users to intervene and avoid the failure. Hence, the proposed model predicts if the supply chain will meet its demand a few weeks ahead. When the model is not confident the supply chain will succeed, we call those supply chain failures.

1.3.2 AI System Use Case

In the second use case, we explore a very different system, which is a AI system that predicts what information should be retrieved from a physical document - in our case filled text on a form. The system is composed of two Deep Learning (DL) models, one ML algorithm and a few heuristics. Quantifying the uncertainty of AI models is a growing topic in the literature. However, uncertainty quantification in a situation where different models interact together to execute a complex task has not been explored significantly. Each model has epistemic and aleatoric uncertainty. One erroneous output of a model can feed into a second model to produce unexpected outcomes. For the AI system of interest, as its purpose is to predict text, we defined a system's failure as a system's prediction that is not an exact match (at least one character of the text is wrong). Inversely to the first use case, we developed a model to predict the confidence in the system being successful. As such, the model outputs its confidence that the predicted text is an exact match.

1.4 Thesis Outline

In Section 2, the LITERATURE REVIEW, we will review the two main areas which gravitate around our two projects mentioned above. First, we will discuss literature around our first use case, uncertainty in supply chains, referred to as Supply Chain Risk Management (SCRM). Secondly, we will present the literature around uncertainty quantification in AI for our second use case.

In Section 3: SUMMARY OF THE WORK, we introduce our approach and the two use cases. We also present additional fields of research which are linked to our approach: expert systems, reliability engineering, predictive maintenance, meta-models and meta-learning. Other fields could also have been of interest, such as uncertainty quantification in traditional statistics, cybernetics, or how OR models typically handle uncertainty. These fields of study were left out as they are only adjacent or complementary to our approach.

In Section 4: ARTICLE 1: A MACHINE-LEARNING-BASED SYSTEM FOR PREDICT-

ING SERVICE LEVEL FAILURES IN SUPPLY CHAINS, we present our paper which was submitted at the INFORMS Journal on Applied Analytics on 2019, May 13^{th}, and re-submitted based on reviewer comments on 2019, November 20^{th}. This work was done with JDA Software and one of their clients, Michelin, an international tires manufacturer.

In Section 5, entitled PREDICTING CONFIDENCE ESTIMATES FOR AN ARTIFICIAL INTELLIGENCE SYSTEM, we present some initial experiments that were done at Element AI, to quantify the uncertainty of an AI system, most specifically in the context of a data entry task. This work is still ongoing, and experiments are too early to be conclusive. However, we believe that our early results are encouraging and show that the approach is promising.

Lastly, in the GENERAL DISCUSSIONS, we draw parallels between the two use cases and approaches, and we link it back to the concepts outlined in the Introduction. As the two projects have been done in different contexts and companies, the reader will notice that the vocabulary and methodology used are somewhat different. As such, this section will discuss how they relate to one another.

CHAPTER 2 LITERATURE REVIEW

2.1 Uncertainty in Supply Chains

2.1.1 Supply Chains are Complex Systems

In the last few years, supply chains have evolved to deal with a lot of added complexity, due to several factors such as increased competition between businesses, rising costs, and the need to serve customers globally [11]. Supply chains are inherently complex systems, where every subpart of the chain has multiple interactions with one another, and where risks and uncertainties make it difficult for companies to deliver their products effectively at low-cost [12]. Since uncertainty at any step of the supply chain can have an impact throughout the chain, risks mitigation need to be managed with a holistic approach, modeling the supply chain from end-to-end [6]. For more information on supply chains, the reader is referred to Stadtler and Kilger [13].

2.1.2 Supply Chain Risk Management

Dealing with uncertainties in supply chains is often referred to as Supply Chain Risk Management (SCRM), where deviations from the plan can cause disruptions in the ability of the supply chain to deliver the right amount of products at the right time. SCRM usually includes both the identification of potential risks as well as the mitigation plans to reduce their impacts and disruptions on the supply chain [14]. SCRM is thus a framework which goal is to increase the resiliency of the supply chain to risks. A few review papers provide a good overview of the SCRM literature, such as Tang [15], Ritchie and Brindley [16], Rao and Goldsby [6], Heckmann et al. [17]. For more information on SCRM, the reader is referred to Khojasteh [18].

2.1.3 Risks Identification: Sources of Uncertainty in Supply Chains

Many papers in the SCRM literature attempt to identify and classify uncertainties and risks in supply chains into different categories. Kumar et al. [12] refer to two types of risks: internal and external risks. Amongst internal risks, is included demand risk - where the forecast is not accurately representing the customer's demand, production risk - such as a machine failing at the plant, distribution risk - causing uncertainty in lead times, and lastly, supply risk - such as shortage in raw materials. On the other end, external risks include the impact of the environment on the supply chain, such as political, social, economic, and environmental factors that may cause delays or problems [12]. Other terminologies are used in other papers for the same two concepts, such as operational (internal) risks - referring to uncertainties such as uncertain demand, supply or cost, vs. disruptions (external) risks - referring to uncertainties due to human-made or natural disasters [15, 11, 19]. Jüttner et al. [20] also propose similar categories. First, the authors refer to external risks as environmental risks - uncertainty caused by the interaction of the supply chain with its environment. However, internal risks are broken down in two distinct risk sources: organizational risks and network risks. Organizational risks refer to uncertainty within the supply chain boundaries, such as machine failures. Network risks, on the other end, refer to risks associated with interactions between sub-parts of the supply chain. The authors highlight that this last type of risk is very impactful but often under-estimated. Some papers classify risks at an even more granular level, such as categorizing sources of risk in environmental, industry, organizational, problem-specific, and decision-maker related factors [6], or between operational factors, market factors, business/strategic factors, product factors and miscellaneous [21]. Heckmann et al. [17] also propose a detailed way to classify risks and explore how to measure and model them. In Khojasteh [18], they classify risks differently, where they focus on the impact of risks instead of its source. The authors categorize risks between process, supply, and demand risks, where internal and external risks can have an impact on each of those categories.

2.1.4 Risk Mitigation

Since risks are inevitable, SCRM does not only care about identifying them. It also involves planning operations in a way that reduces risks impact when they happen, while maintaining low costs. Most literature in SCRM revolves around that area, usually referred to as risk mitigation. It relies on some quantification of the risks, which can then be added to the supply chain plan to increase its robustness in the face of uncertainty. Khojasteh [18] provides an exhaustive review of papers using quantitative methods to mitigate uncertainties. Most papers use mathematical models to do so, such as stochastic programming, and a few uses game theory or statistical models. In another review paper, Tang [15], one of the most cited paper in SCRM, the author classifies papers according to how they approach risk mitigation - supply management, demand management, product management, and information management. Fahimnia et al. [22] also classify quantitative and analytical models for managing risks in supply chains. Most of these methods, such as stochastic optimization, rely on some quantification of the risks so it can be taken into account in the problem modeling [17]. Different papers approach this differently, using tools such as scenario-based modeling or deviation-based measures (variance, standard deviation) [18]. In other cases, the use of probabilities can be difficult due to a lack of historical data or if the historical data is not representative of the future. A such, some papers use possibility distributions, such as fuzzy theory [18, 23].

2.1.5 Gap in Literature: Predicting System Failures

In the literature, the field of SCRM and the field of uncertainty quantification are typically disconnected. As such, most approaches to handle uncertainty in the supply chain have been tackled by supply chain researchers integrating risks notion into the supply chain models, instead of adapting risk models from the literature to supply chains. Hence, there is a gap in the literature around approaching SCRMs with typical uncertainty approaches [24]. As presented above, most papers in the SCRM domain revolves either around identifying sources of uncertainty or around risk mitigation. Few authors have explored using typical risk analysis methods in supply chain context, such as uncertainty propagation with Bayesian networks [24, 25]. No paper explores system failure predictions in supply chains.

2.2 Uncertainty Quantification in Artificial Intelligence

2.2.1 Why Is There Uncertainty in Artificial Intelligence?

AI, such as ML or DL models, are complex systems which aim to perform a prediction task by learning complex patterns and interactions in some dataset. However, data tend to be noisy and incomplete, affecting models' ability to perform well on new data. Begoli et al. [26] discuss some challenges in DL, which explains why models tend to fail in certain contexts. First, in most DL applications, there is an absence of theory, referred to as "hard laws" such as laws of physics, which makes it impossible to ground uncertainties and bound assumptions. Second, DL models usually rely more on correlation than causation, which causes a lack of robustness when a test data point is far from the training data. Third, DL models are sensitive to noise in the data.

2.2.2 Why is Uncertainty Quantification Important in Artificial Intelligence?

In some applications, such as medical cases, the cost of an error can be fatal. Hence, uncertainty quantification is essential so that models can indicate their level of confidence in their prediction. However, even in low-stake scenarios, uncertainty quantification is crucial to build trust with the models' users [27], identify opportunities for HITL [28] and also identify erroneously classified data points and Out-Of-Distribution (OOD) instances, including distributional shift in the data [29, 27]. A related application is active learning, where uncertainty estimates are used to identify the priority of the data that should be labeled. Finally, uncertainty estimates can also help with model interpretability, especially when well-calibrated, as humans tend to have a good intuition for probabilities [30].

2.2.3 Why is Uncertainty Quantification Hard in Artificial Intelligence?

The most naive approach for uncertainty quantification is to directly use a model's output as an uncertainty estimate, such as soft-max probabilities [29]. However, different authors have shown experimentally that this does not always work well, and that AI models tend to estimate their uncertainty inaccurately. Even more so, in DL, model outputs tend not to be well-calibrated, meaning that their uncertainty score is not a good indicator of the model's performance [30, 31]. Other papers go further by showing how even the relative ranking of the uncertainty scores cannot be trusted [32, 33]. It suggests that classifiers are not the best judge of their trustworthiness [27]. For these reasons, different research avenues have been explored to perform uncertainty quantification for AI in the last few years. Some of these approaches are specific to some model architectures, such as MC Dropout with neural networks [34], while others involve using unique models designed to estimate their uncertainty, such as prior networks [35]. Some of these techniques model aleatoric uncertainty, others epistemic, and a few even both. Some frameworks, such as Bayesian Neural Networks, provides a robust mathematical approach for uncertainty estimation, but in practice are far too computationally intensive for real-world problems [36]. As a result, there is no explicit agreement amongst the scientific community as to which method works best and how to evaluate them. Below we will present a brief overview of the literature in this domain, with a focus on classification problems.

2.2.4 Approaches for Uncertainty Quantification in Artificial Intelligence

Bayesian neural networks approaches to estimate uncertainty have gained popularity in the last few years to estimate uncertainty, such as the quite popular method known as MC Dropout [34, 5, 37]. These approaches do not change the structure of the network and return a distribution over the outputs. Other approaches design their own model architecture to provide uncertainty estimates [36, 35, 38, 39], while other authors used the level of agreement of the classifier with a secondary model [27]. The techniques mentioned above tend to provide uncertainty estimates which will not preserve the initial ranking of the model's outputs. By that, we mean that the ranking of the model's outputs, such as soft-max probabilities, will not follow the ranking of the uncertainty estimates. Model calibration is a specific area of research where the focus is not on changing the ranking, but on calibrating the output, so that it becomes representative of the true correctness likelihood. Different approaches were explored throughout the years, such as histogram binning, Platt scaling, and temperature scaling [40, 41, 28, 42, 31, 30]. Finally, uncertainty quantification is also related to the field of classification with a reject option, where the model can decide not to classify a data point if it costs less than misclassifying it [43, 44, 45].

2.2.5 Dataset Shift and Out-Of-Distribution Detection

Dataset shift and OOD detection are two related concepts that uncertainty quantification approaches typically aim to address. OOD detection typically refers to the broader theme of identifying test data points that are far from the training distribution, which can cause the AI system to have unpredictable behaviors. It is difficult to assess the performance of different approaches in tackling this problem, as we cannot know the distribution of outliers. Assumptions around the nature of these OOD need to be made, which may provide over-optimistic conclusions on the viability of different approaches. Shafaei et al. [46] present an evaluation scheme to address this problem. Amodei et al. [47] identify dataset shift as being one of the five most important problems for AI safety. Dataset shift typically happens in the more specific case where a deployed AI model is exposed to a dataset with a distribution shift from the one used at training time. Some authors include it as a special case of OOD. Dataset shift usually results in a drop in the model's performance, which may be even more dangerous if the model wrongly assumes the same performance and still provides high confidence estimates. Ovadia et al. [48] state in a recent paper how dataset shift can be due to different factors, such as sample bias and non-stationarity, and presents a rigorous benchmark to compare the performance of different uncertainty methods for dataset shift detection.

2.2.6 Gap in Literature: Artificial Intelligence Systems

As seen above, uncertainty quantification is a challenging task in ML, for which there is no clear consensus on the best approaches and evaluation methods. However, what makes it even more complex in real-world applications is that models are often not working in isolation and can be part of a system of models. Models can interact with one another, sometimes sequentially or even in more complex workflows. Approaches mentioned in the literature cannot be used directly for such use cases.

CHAPTER 3 SUMMARY OF THE WORK

3.1 Our Approach

In this section, we present the Full-Stack Uncertainty Model (FSU), our approach to predict system failure, and describe how we apply it to our two use cases. The approach's objective is to focus the user's attention when the system is at risk, where an action can be performed to avoid failure. As such, our approach aims at improving the system's performance.

3.1.1 The Full-Stack Uncertainty Model

Our approach consists of using a ML classifier to predict whether a failure may occur, based on features describing the state of the system at different stages. The model trains on historical data, where $\mathbf{x}$ represents an instance of the system in action and y is a binary variable denoting whether the system met the desired purpose at that instance, as shown in Figure 3.1. The classifier hence learns how different states of the system can be indicative of failures. The different features typically represent the uncertainty that can occur at any stage of the system and cause the whole system to fail. As part of the second project, we named this model the FSU, but we use this terminology to describe our approach in both projects in this section and in the Discussion. The FSU produces an uncertainty estimate quantifying the likelihood of the system failing at a given moment. In uncertain situations, it can trigger the intervention of a HITL that can rectify the output of the system. Consequently, we consider that the FSU and the HITL become part of the system and increase its overall resiliency. Considering the user's actions as part of the system's behavior is similar to the approach of analyzing machine behavior at the hybrid level, as mentioned in the Introduction [8].

3.1.2 Introduction to the Two Use Cases

In Figure 3.2 and Figure 3.3, we represent the two use cases using a similar visualization. We display the different subparts of the system, where rectangles are its elements, circles are input and output, and arrows interconnections. The color of each shape represents different types of elements. Red boxes are steps that are not deterministic, such as physical processing units in the supply chain context. Yellow boxes generally represent some learning algorithms, such as ML models, and grey boxes are heuristics. In reality, the line between each type is blurry; the Form Aligner step in Form Extractor (FE) is a heuristic with some stochasticity, and each subpart of the supply chain contains heuristics and some learning machines.

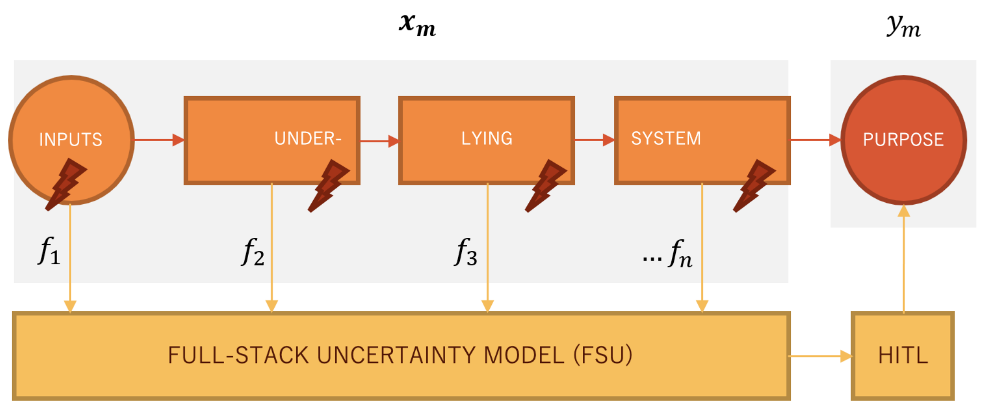


Figure 3.1 The FSU uses features from different stages of the underlying system to provide an uncertainty estimate of the whole system achieving its purpose. Features typically characterize the uncertainty that may occur in each system's elements, here represented by lightning. The uncertainty estimates, when high, can trigger the intervention of a Human-in-the-loop (HITL) which can rectify the system to avoid the failure.

In both figures, we identify the primary sources of uncertainty with a lightning in a triangle. In red is represented aleatoric uncertainty and in purple, epistemic uncertainty. The dashed rectangle identifies the boundaries of the system that we set for each FSU model. As it is difficult to draw the systems' boundaries and to explore a system at the right level of granularity, these representations are not necessarily complete and display only components which were useful for the discussion. More details on the use cases and the FSU are provided in sections 4 and 5.

3.2 Adjacent Topics

There are other fields of research that were not included in the Literature Review but are nonetheless linked to the work presented here, albeit tangentially. Below, we draw parallels between our proposed approach and some of these fields.

3.2.1 Expert Systems

Expert systems leverage experts' knowledge to solve complex problems. They are typically divided in a knowledge component and an inference engine. The former contains knowledge accumulated by experts on the problem at hand. The system then uses this information to infer on new situations, typically with if-then rules. They can be used to diagnose, control or monitor a system. Our approach can be seen as an expert system, where the experts'

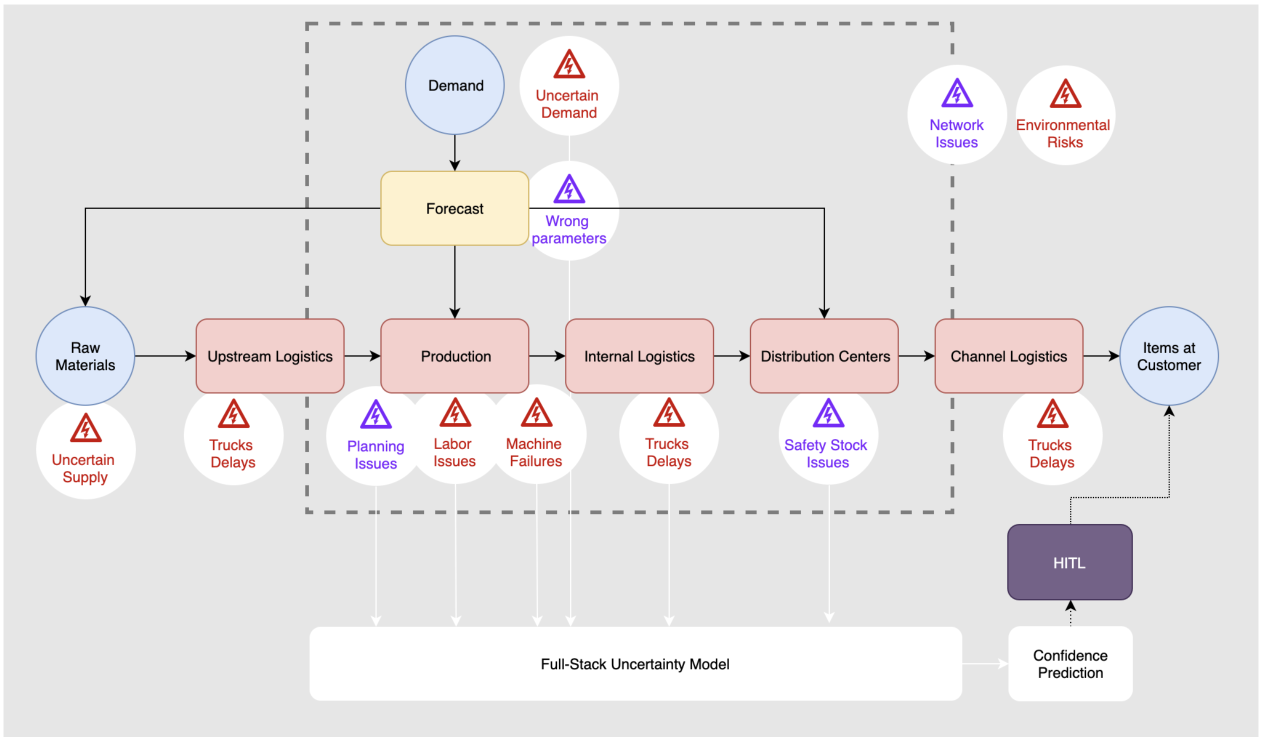


Figure 3.2 Full-Stack Uncertainty Model System for failures prediction in the supply chain.

Notes. In supply chains, uncertainty can be present in the demand and supply, in operations - such as machine failures and truck delays, in interactions amongst the different supply chain segments (network risks) or by external events such as political policies or natural disasters. Uncertainty can also come from epistemic and aleatoric uncertainty in the different models, or even in the system design.

knowledge is present in the features engineered to monitor failures. However, our approach goes further as it also allows to learn directly from the data and not just through expert knowledge.

3.2.2 Reliability Engineering

In engineering systems, usually composed of a mixture of hardware and software components, the task of predicting a system's failure is called reliability engineering. Reliability, also referred to as dependability or availability (when reliability is estimated for a specific period of time), is usually defined as being the inverse of the probability of failure. Reliability modelling tests how a system will react in the face of uncertainties and risk, before implementation. As reliability engineering is usually performed on hardware systems, an exception being software reliability, a system's failure is usually due to physical failure in one of the components, which usually cause the whole system to fail. It is hence more specific than our approach.

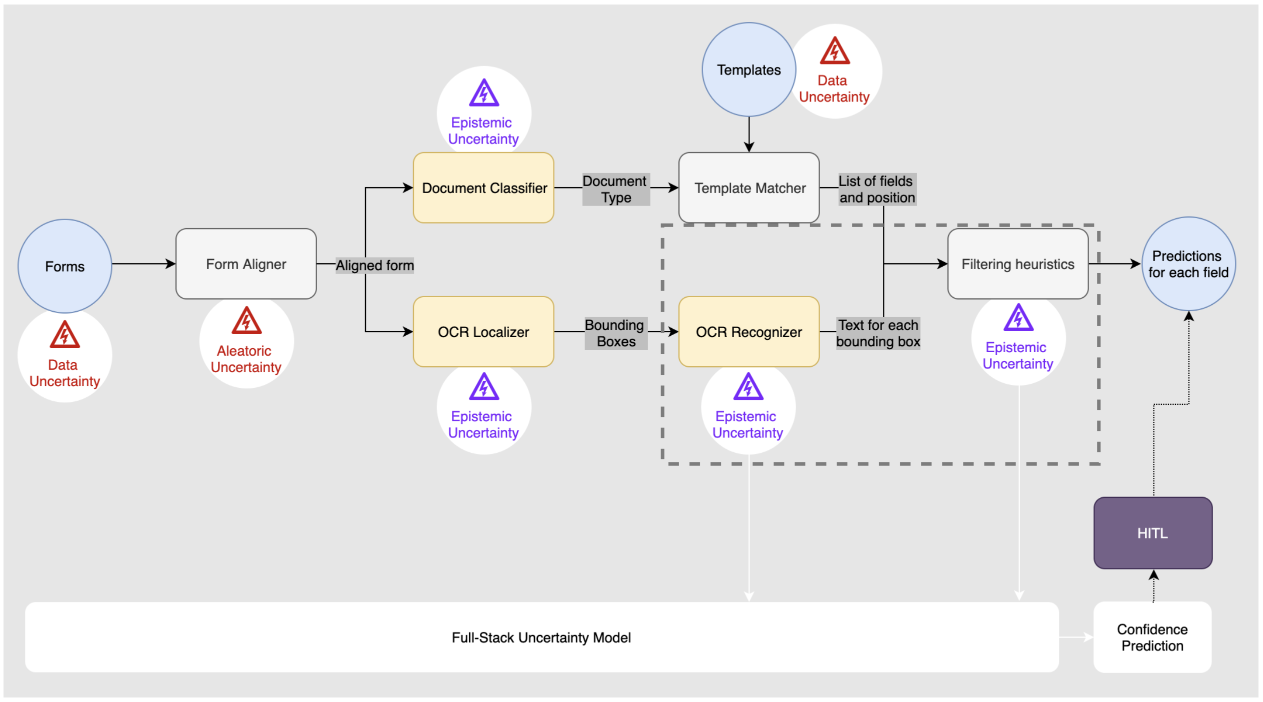


Figure 3.3 Full-Stack Uncertainty Model System for failures prediction in Form Extractor, an Artificial Intelligence system.

Notes. In FE, each model has epistemic and aleatoric uncertainty (the latter is not represented). The system's input, such as the forms and the templates, can also be noisy. Form Aligner is a stochastic heuristic; it does not always produce the same outcome.

3.2.3 Predictive Maintenance

Predictive maintenance concerns the evaluation of equipment condition to determine when maintenance should be executed in order to avoid future failures. It is used for physical machines and equipment in factories, transportation and similar fields. Similarly, our approach can be useful for detecting when systems should undergo tuning or adjustments. In our use cases, maintenance can mean retraining one ML model to adapt to a shift in the distribution of the data.

3.2.4 Meta-Models

A meta-model, or surrogate model, is typically a simplification of a model, abstracting some of its properties while representing relations between the inputs and outputs of the system. The approach of the FSU can be seen as a surrogate model, learning how the input of an underlying system relates to its output being correct or incorrect.

3.2.5 Meta-Learning

In ML, meta-learning is a field where models are trained on meta-data from other ML models. The objective is to produce a system that learns to learn and can adapt to changing environments and conditions. This is important in ML as models are built with different assumptions, such as the data distribution, which cause them to not perform well in new conditions. Meta-learning resembles our second use case, where our FSU model uses meta-data from the underlying ML models to learn when they are not well-suited to perform in the face of uncertainty.

CHAPTER 4 ARTICLE 1: A MACHINE-LEARNING-BASED SYSTEM FOR PREDICTING SERVICE LEVEL FAILURES IN SUPPLY CHAINS

4.1 Authors

Gabrielle Gauthier Melancon, Polytechnique Montreal, JDA Software
gabrielle.gauthier-melancon@polymtl.ca

Philippe Grangier, JDA Software
phi.grangier@gmail.com

Eric Prescott-Gagnon, JDA Software
ericprescottgagnon@gmail.com

Emmanuel Sabourin, JDA Software
emmanuel.sabourin@jda.com

Louis-Martin Rousseau, Polytechnique Montreal
louis-martin.rousseau@cirrelt.net

4.2 Journal

Submitted to INFORMS Journal on Applied Analytics (IJAA). See 4.14 for details.

4.3 Abstract

Despite advanced supply chain planning and execution systems, manufacturers and distributors tend to observe service levels below their targets due to different sources of uncertainty and risks. These risks, such as drastic changes in demand, machine failures, or systems not properly configured, can lead to planning or execution issues in the supply chain. It is too expensive to have planners continually track all situations at a granular level to ensure that no deviations or configuration problems occur. We present a machine learning system that predicts service level failures a few weeks in advance and alerts the planners. The system includes a user interface that explains the alerts and helps to identify failures fixes. We conducted this research in co-operation with Michelin.

Keywords: Supply chain management; manufacturing; machine learning; human-computer interface; explainable AI

4.4 Introduction

Supply chain planning typically comprises multiple optimization systems that differ in scope and planning horizon, from strategic sales and operations planning to near-real-time transportation systems. Despite advanced planning and execution systems, manufacturers and distributors tend to observe service levels below their targets due to different sources of uncertainty throughout the supply chain. SCRM is the field dedicated to identifying these risks and mitigating them.

Jüttner et al. [20] classify sources of uncertainty in supply chains between environmental, organizational, and network risks. Environmental risks refer to the impact of the environment on the supply chain, such as natural disasters and political factors. Organizational risks include uncertainty within the supply chain, such as delays in transport or production problems. Lastly, network risks refer to issues due to poor interactions between the subparts of the supply chain. Indeed, as supply chain planning requires the collaboration of different teams and systems that need to be tightly integrated, issues in systems configurations may be undetected and lead to sub-optimal plans. The authors underscore that network risks are a very impactful but often neglected source of risks. As part of SCRM, risk mitigation is an area of research that focuses on building a robust plan that can account for different sources and magnitude of uncertainty. Yet, it can be too costly to plan for the worst outcomes, and it is impossible to model uncertainty perfectly. Consequently, service level failures can occur in supply chains.

Recent advances in ML and the growing availability of data have initiated a steady stream of research combining ML and supply chains [49]. In this paper, we present a system that uses ML to raise alerts when the supply chain conditions may lead to service level failures. The alerts need to anticipate issues in time for the planners to take corrective actions but not so early that the next plan naturally accounts for them. The system focuses the attention of the planners on alerts that are *actionable* (it is possible to avoid the failure), *exclusive* (other systems did not detect the issues), and *significant* (failures concern important items for which performing the corrective action is worthwhile). The system also aims to *explain* alerts by identifying their underlying causes, so users gain confidence in the results and get the necessary context for potential fixes. We developed the system in co-operation with Michelin, an international tire manufacturer, which provided the business use case and the data.

In the remainder of this paper, we first present *Related Work*, followed by the section *Michelin Context* outlining some of Michelin's current challenges and introducing how we framed the problem. In the *Methodology* section, we outline how we modeled the problem and made predictions using ML. In the *User Interaction with the Model* section, we present the workflow and UI that we developed to make the system's predictions useful and explainable. This is followed by the *Results* section. Eventually, we give some perspectives on our work and the remaining challenge for ML to have a deep impact on supply chain management in *Perspectives*. The *Appendix* contains more details on the model and the data.

4.5 Related Work

Nguyen et al. [49] recently published a survey of big-data analytics for supply chains that classifies the studies by supply chain functions, including *demand management*, *manufacturing*, *warehousing*, and *general supply chain management* when the study encompasses multiple functions of the supply chain together. In the survey, papers falling in the latter category are either descriptive or prescriptive approaches on topics such as managing sustainability [50] or natural disaster risk management [51], but the authors do not report any predictive approaches. They highlight the increase usage of ML for specific areas of supply chain management, such as demand forecasting and machine maintenance. Our approach, which would fall in the predictive applications for the general supply chain category, thus clearly stands out in the current domain's stream of research. Additionally, to our knowledge, no software vendors were offering a disruption prediction tool ingesting data from several supply chain components simultaneously, at the time of this study.

In the SCRM literature, most papers focus on the identification of risks, such as Heckmann et al. [17], Kumar et al. [12] or on their mitigation, such as Schmitt [52], Paul et al. [53]. Still, a few papers combine both. Simchi-Levi et al. [54] propose methods to identify and mitigate risks in the automotive supply chain context at a tactical decision level. Garvey et al. [24] propose a Bayesian network framework to model risk propagation, tested on simulated data. In contrast, our approach predicts failures on a short term horizon on real live data.

For more information on supply chain planning, the reader is referred to Stadtler and Kilger [13], and to Khojasteh [18] for SCRM specifically.

4.6 Michelin Context

Michelin is an international manufacturer that produces and sells tires for a vast range of vehicles, from cars and motorcycles to tractors and aircraft. Michelin produces roughly 200M tires per year and has a commercial presence in 170 countries, reaching 13.7% of the global tire market in 2014. For the car tires segment, Michelin distinguishes two channels: one for orders placed well in advance (typically large quantities, for car manufacturers or large retailers) and one for orders placed only a few days ahead (typically for local mechanics), called *store-and-sell*.

4.6.1 Service Level Failure

In this study, we focused on the store-and-sell channel for car tires in Europe. For this channel, Michelin has a catalog of products, each with a given target delivery window. For example, a 48h delivery window means that a local garage can place an order and expect delivery within 48h. For the respective Michelin's DC, this translates into a deadline by which the items need to be available for delivery. Inability to meet this deadline is considered a supply chain failure for Michelin. Michelin tracks the performance of its supply chain with Key Performance Indicators (KPIs) aggregating orders' service level over different scopes, such as products group, regions, periods. In this study, we will focus on service levels aggregated by item and DC at a weekly frequency, and consider a service level failure as being situations where this aggregation is below Michelin's target.

4.6.2 Challenges and Opportunity

In an ideal world, Michelin's planners would continuously monitor the supply chain data and adjust its parameters when they detect situations or patterns that could lead to service level failures. However, they are usually unable to monitor the data at a granular level because of the high volume of data and the complexity of supply chains. To illustrate, for the store-and-sell channel in Europe only, Michelin has around 4k different types of items, which are produced in 10 plants and stored in 15 DCs. Additionally, the planners' responsibilities are typically siloed between the different supply chain segments. As such, the ownership of the service level's performance is shared, increasing the complexity in identifying issues and mitigating them. For these reasons, planners adopt a proactive approach for the most important items only, and resort to a corrective approach, adjusting the parameters only once the service level has fallen significantly below the target, for the vast majority of their products. This approach is reasonable because it would be too costly to monitor all items. Yet, having a system that would predict situations at risk that require planners' attention would help in improving the supply chain performance. Moreover, such a system could detect problems that would be missed by human planners.

4.6.3 Sources of Service Level Failure

Different sources of uncertainty in the supply chain cause problems which Michelin typically categorize as execution issues or planning issues. *Execution* issues refer to situations where the plan was adequate to fulfill the orders on time, but an event created a disruption leading to a failure. It can be due to environmental and organizational risks, such as delays in shipments and machine failure. Execution issues are typically challenging to foresee and prevent. By *planning* issue, we mean that the plan was inadequate in fulfilling the right amount of items on time. Planning issues can be imputable to uncertainty in demand, which the plan do not account for, or inadequate systems' configurations. By *configurations*, we refer to the systems parameters and rules that create the supply chain plans, such as safety stock targets, demand forecast settings, and master plan parameters. Additionally, planning issues can be due to poor interaction between the different subparts of the supply chain, i.e., network risks, as typically, manufacturers tend to leverage heterogeneous systems with limited integration, either from different vendors or built in-house. Typically, planning issues can be detected a few weeks in advance and as such, be predictive of future service level failure.

In previous internal studies, Michelin identified that planning and execution issues impacting the service level occurred mostly at the stage of production, in internal logistics, and at DCs. As such, and to limit the data effort, those segments were the only ones included in the model, as presented in Figure 4.1.

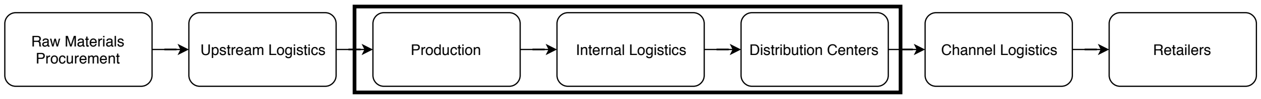


Figure 4.1 In the store-and-sell channel, the flow of material starts at raw materials procurement and ends when retailers receive the tires. As part of this study, we measured service level failures at the shipping door of the DCs. As such, we ignored channel logistics. Raw materials procurement and upstream logistics were also left out as Michelin determined these segments were rarely influencing service level failures.

4.7 Methodology

In this section, we present how we framed the problem of service level failure prediction from a ML perspective and our methodology to train such a model.

4.7.1 The task

We defined a service level failure as a situation where the service level for an item at a DC for a given week is below Michelin's service level target, resulting in a binary variable. The model predicts a service level failure when its output, often log-odds ratio in classification, is above a certain threshold. We call *alert* such situation. Given the context of Michelin's supply chain, we used a prediction horizon of 14 days, i.e., we generated predictions every week on Mondays for the week starting 14 days later for each item–DC pair.

4.7.2 Features

We engineered features from the raw data to represent the state at each segment of the supply chain included in the rectangular represented in Figure 4.1. Generally, the features compare the actual and planned values for a few periods before the moment of prediction so that they can measure uncertainty and deviations from the plan. Yet, historical data was not available for all possible factors impacting the service level, limiting us in our ability to model the situation at any given time precisely. Thus, we added features that were good proxies in estimating the uncertainty at each step of the supply chain. As an example, we did not have access to machine failures data, but we could model if the number of tires produced matched the plan, a good estimation for production's execution performance. Additionally, we included the service level of the preceding weeks as features, since they are good indicators of the global health of the supply chain. We detail the features that we used in the Appendix, as well as potential ones we would have liked to use, as data allowed.

4.7.3 Model

We selected GBDTs [55] as implemented in XGBoost [56] after some initial comparison with logistic regression, random forests, and neural networks. They offered the best performance overall, are easy and fast to train, and can handle missing values out of the box. GBDTs are an ensemble of decisions trees build iteratively where each tree's target is to correct the cumulative error made by preceding trees. The first tree generally trains on the delta between the targets and a baseline, typically the average value of the training set. Additionally, being trees based, GBDT models are usually considered explainable, as detailed in section *User Interaction with the Model.*

4.7.4 Evaluation Metrics

For unbalanced classes, such as failure prediction, performance metrics are typically area under the Receiver Operating Characteristics (ROC) or precision-recall curves. We selected the area under the precision-recall curve, which is created by plotting the precision (ratio of true positives vs. all predicted positives) against the recall (ratio of true positives vs. all positives). This choice was justified by Michelin, whose questions were centered around: "how often is your system correct when it predicts a problem" and "how many of the problems are you capturing?", which the precision-recall curve answers directly.

4.8 User Interaction with the Model

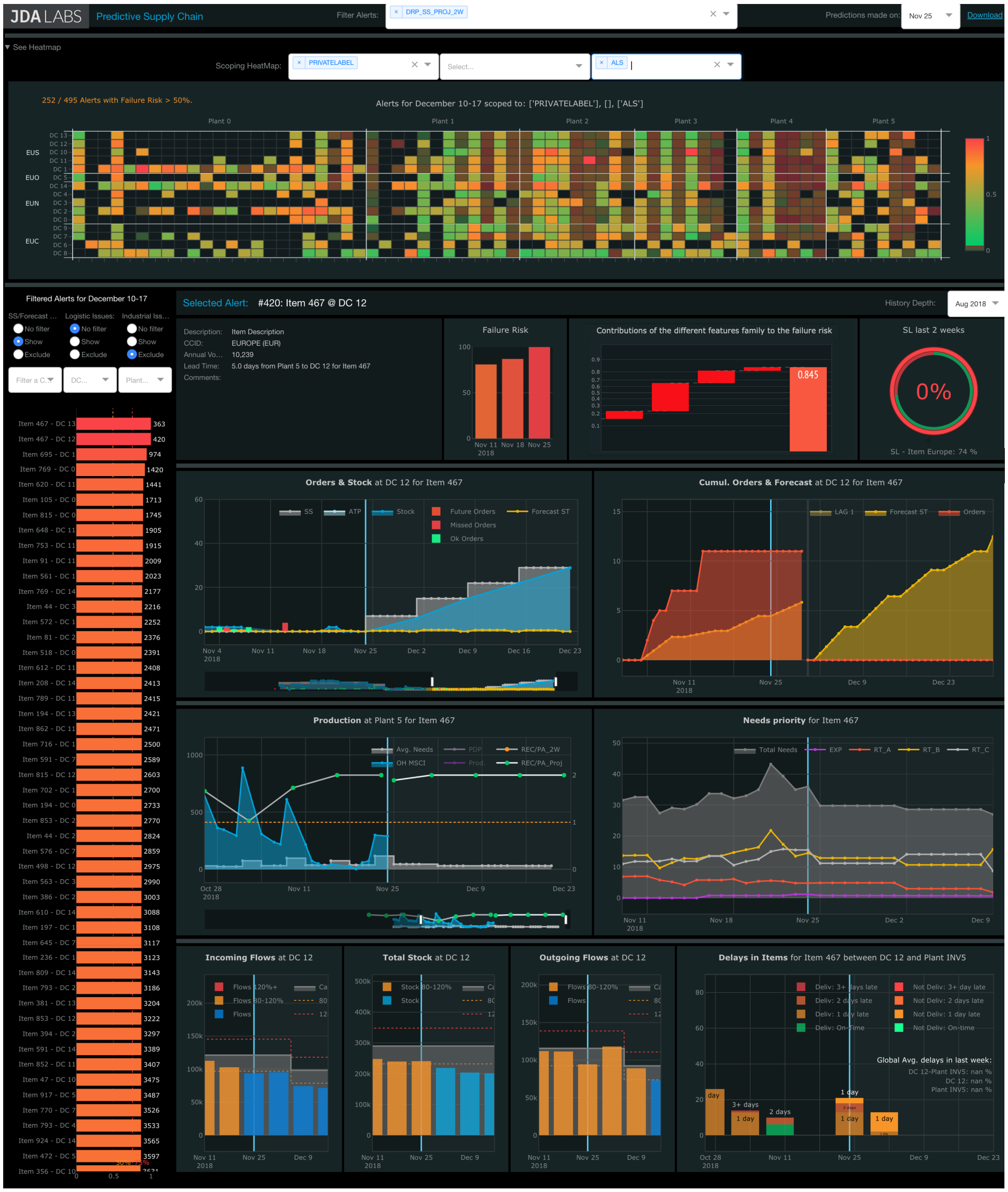
In this section, we describe important features of the workflow and supporting UI that we developed for users to interact with alerts. A classical spreadsheet approach containing the model's predictions do not answer users' needs around trust, context, and explainability. The UI, as shown in Figure 4.2, serves as a dashboard where are shown the outputs of the model, the data used (such as stock levels, production plans, logistics delays) and some additional data not relevant for the ML model, but helpful for finding the right resolution for the alert. By grouping all the information, users do not have to access multiple systems to get the necessary information and can save time to assess and resolve the case. It also increases the users' confidence in the model's results. We will present three essential features from the workflow and UI.

4.8.1 Supply Chain Health Check

First, as an entry point to explore all alerts, a heatmap near the top displays each item–DC combination, where the color (from green to red) represents the likelihood of failure. Items are on the x-axis, grouped by plant, while DCs are on the y-axis grouped by region. At a glance, managers can assess the health of the supply chain and investigate correlations.

4.8.2 Focusing on Useful Alerts

Second, due to the amount of alerts generated by the system, users need the ability to filter alerts to focus on the ones where they can have an impact. We defined the notion of a *useful* alert as being exclusive, actionable, and significant situations. By *exclusive*, we mean that the system should generate alerts that are not obvious or detected by other tools. For example, planners are usually aware of production capacity problems and quality issues. By



Filter Alerts: DRP_SS_PROJ_2W

Predictions made on: Nov 25

Scoping HeatMap: PRIVATELABEL, Select..., ALS

252 / 495 Alerts with Failure Risk > 50%.

Alerts for December 10-17 scoped to: ['PRIVATELABEL'], [], ['ALS']

Filtered Alerts for December 10-17

SS/Forecast ...: Show

Logistic Issues: No filter

Industrial Iss...: Exclude

Selected Alert: #420: Item 467 @ DC 12

History Depth: Aug 2018

Description: Item Description
CCID: EUROPE (EUR)
Annual Vo...: 10,239
Lead Time: 5.0 days from Plant 5 to DC 12 for Item 467
Comments:

Global Avg. delays in last week:
DC 12-Plant INV5: nan %
DC 12: nan %
Plant INV5: nan %

Figure 4.2 We developed an interactive UI which can be used to have an overview of all alerts and to explore each prediction individually, providing context and explanations to identify potential failure resolutions.

Notes. The UI was developed using Dash by Plotly. It connects to the model output and additional data sources.

actionable, we mean that alerts should identify situations where the planners can attempt to avoid the failure event. Potential actions include changing the forecast, adjusting the safety-stock levels, and adding transportation options. The prediction horizon should be long enough to ensure a minimum number of available actions. For example, anticipating a stock-out for the next day when the replenishment lead time is one week is not helpful. By *significant*, we mean that alerts need to concern items and locations that are important (e.g., in terms of volume or strategy) so that the corrective action is worth the effort and cost. Nevertheless, alerts that are not exclusive, actionable, or significant may still allow supply chain planners to forewarn customers of delays and highlight recurrent problems that could be alleviated by structural changes in planning processes. In the UI, we enable users to filter out non-useful alerts through dropdown filters (in the white box at the top).

4.8.3 Model Explainability

Lastly, in the waterfall graph in the top third of the screen (we present a zoomed version on Figure 4.3), we provide some explainability around the predictions. In general, explaining results is increasingly recognized in ML to help users gain confidence in the system. On top of that, identifying which supply chain conditions lead to a failure prediction can help the planners in finding the appropriate resolution. One way to explain a model's output is to identify the most important features. Additive feature attribution methods [57] are approaches that assign a contributing value to each feature for each prediction, such that the sum of all contributions is equal to the model's output. Through game theory, we can consider this value attribution problem as a cooperative game, by viewing all players as the different features. The solution to this problem is named Shapley values [58]. The intuition behind those values is to compare the prediction with and without each feature. Unfortunately, computing Shapley values has an exponential complexity, and as such, they typically need to be approximated. We used a recently proposed method called Tree SHAP [2], which estimates Shapley values for decision trees in pseudo-polynomial time. As Shapley values are additive, we can display them in a waterfall graph such as in Figure 4.3.

4.9 Results

In this section, we present the experiments we conducted and discuss the results we obtained both from a statistical and business perspectives.

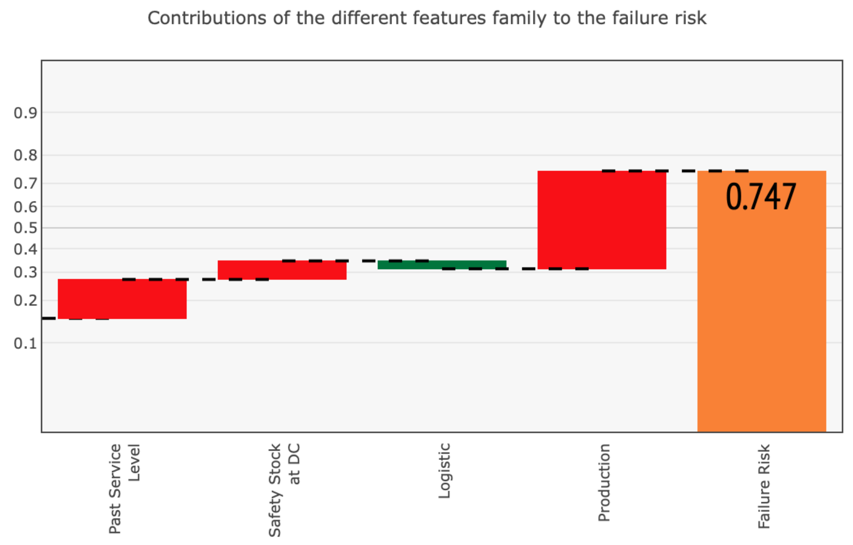


Figure 4.3 In the context of GBDTs, Shapley values sum to the difference between the prediction (failure risk) and the model's baseline, in our case, the average service level in the training set (y intercept). In this example, the rightmost bar displays the overall prediction of 0.747. Through a waterfall graph, the other bars illustrate the sum of the contributions (Shapley values as estimated by Tree SHAP [2]) for each feature family. The Table 4.3 in the Appendix indicates the mapping between features and features' family that we considered. Contributions can be positive (green) and decrease the failure risk or negative (red) and increase it. Note that Shapley values share the same units as the model's output, here log-odds ratio, and as such, they represent log-odds contributions. The graph uses a non-linear (logit) y-axis so that the contributions of the bars can be linearly comparable.

Notes. In this example, the main underlying cause is a production problem. The service level in the last few weeks and features representing the safety stock at the DC are also factors which increase the failure risk. Favorable logistic conditions have slightly lowered it.

4.9.1 Experiments Overview

The project was carried out in four phases. In the first, we gathered data and developed the model, by focusing on about a hundred 17″ summer tires that Michelin identified as adequate representatives of the general situation in their supply chain. Once we validated the performance of the model, we moved to the second phase, where we performed live tests for ten weeks. This phase's focus was on users' adoption of the system, as we developed a workflow and UI to convert predictions into actionable alerts that could bring business value. Users experimented dynamically with the tool for a few weeks, during which they identified pertinent issues in their supply chain. As planners found the tool useful, we moved to a third phase, where we tested if our model could generalize to an extended range of products. As such, we tested the 17″ summer tires model on the complete range of Michelin car tires in Europe and obtained satisfactory results. Thus, we pursued with a fourth phase where planners used the system dynamically once more, for six weeks, and performed corrective actions at scale to measure the impact on the service level.

4.9.2 First phase - Initial Performance

For the first phase, targeting 17″ summer tires, we had more than 43k data points covering 14 months and describing the supply chain conditions of 95 items, produced at 10 plants, and stored in 16 DCs. All tires were mono-sourced, i.e., produced in one plant at a time. These orders represented over 23k customers. We used the first twelve months as the training set and performed nested Cross Validation (CV) and Grid Search for hyper-parameters' tuning. In the *Appendix*, we list the selected hyper-parameters and the CV parameters. We used the remaining two months and a half as a test set. Figure 4.4 shows the precision-recall curve of the model. The mean service level is around 87.5%. As such, a random classifier would correspond to a horizontal line at $y = 0.125$, as pictured with the horizontal dotted line. Our curves are significantly above this, which indicates the predictive value of the proposed model.

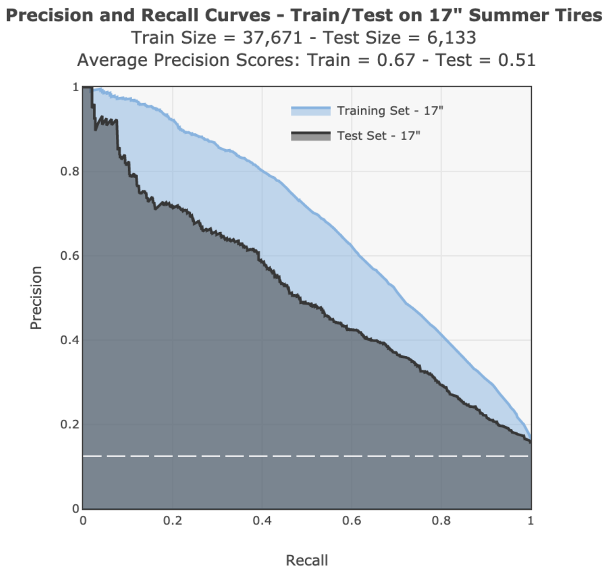


Figure 4.4 Precision-recall curves for train and test on 17" Summer Tires. The gray area represents the training performance on the first twelve months of data for 17" summer tires, more precisely from 2016-09-01 to 2017-09-20. The blue area is the test performance on the following two months and a half of data for the same tires, from 2017-09-21 to 2017-12-12. The horizontal dotted line displays the mean service level in the dataset. The model reaches good performance.

4.9.3 Second phase - Business Validation with Users

In the second phase, we developed a UI so planners could interact with the model's predictions. They used the system dynamically for a few weeks, and we iterated on the UI so it could support a manageable workflow where planners could quickly identify *useful* alerts and have an impact. As such, we verified the model's performance on the subset of useful alerts only. As shown in Figure 4.5, the performance is slightly lower than for all the alerts, and in particular, the system does not reach a high precision level. Since users filtered out easy-to-find problems captured by other systems, i.e., not exclusive alerts, we expected a drop in performance. Nonetheless, results were still satisfactory.

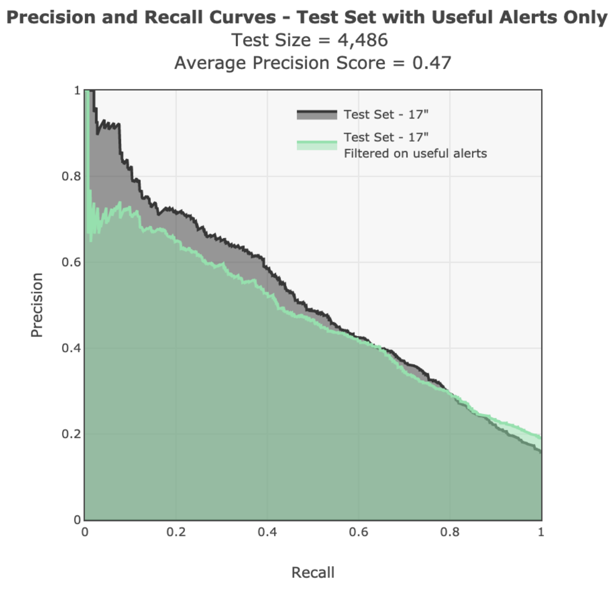


Figure 4.5 Precision-recall curve on useful alerts only. The blue area represents the test set with 17" tires as shown in Figure 4.4. The green area displays the performance on a new test set of 17" summer tires, filtered down to useful alerts only. However, for this analysis, only exclusive and actionable alerts were considered as alerts concerning tires of all significance were included (big runners, medium runners and long tail products).

While using the system, planners were able to identify recurrent issues in the supply chain imperceptible in high-level metrics. Below, we present three such situations identified during the second phase.

First, through multiple similar alerts generated by the system, Michelin detected a problem in the computation of the safety stock for their small-volume tires, affecting around 25k units on that week. To temporarily fix the problem, the planners manually changed the safety-stock targets of the item–DC combinations for which the system predicted service level failures. After a few weeks, to everyone's surprise, planners discovered that the safety-stock overwriting process was faulty as well, so the manual changes had no impact. A month after discovering these issues, the number of items affected by the problem was reduced to around 2.8k tires, a decrease of 89%.

Second, the system detected multiple situations where the underlying cause was a forecasting issue, highlighting that the forecasting algorithm was not sufficiently dynamic. The system identified individual cases of under-forecasting and raised awareness of these issues, motivating further work at Michelin on forecast improvements.

Third, the system detected multiple situations where an item soon-to-be discontinued was phased out too aggressively, given the customer demand. The planners reached out to the appropriate team to address this issue.

Overall, these issues result from supply chain's systems that are either wrongly configured, no more adapted to the current supply chain dynamics, or not interacting well with one another. By raising Michelin's awareness on these issues so they could fix the problems at their source, our alerting system already had a positive impact beyond the tires in the scope of this phase.

4.9.4 Third phase - Full Scope of Tires in Europe

For the third phase, we had access to the data for most of the car tires sold by Michelin in Europe (around 4k items), ranging from 13″ to 22″ and beyond. The data spread about 11 months and represented about 500k points. Because we had access to less than a year of data, we decided, together with Michelin, not to retrain the initial 17″ model and test it directly on the extended scope. Figure 4.6 shows that the extended performance is similar to its initial one on 17″ tires, illustrating that the model generalizes well.

4.9.5 Fourth phase - Dynamic Phase on Full Scope

The focus of phase four was on testing the system dynamically on the full scope of tires and performing corrective actions at scale to measure the impact on the service level. To that end, during the last three weeks of the phase, planners implemented an automatic process to take corrective actions based on the alert's underlying cause. More specifically, the process would modify the safety stock target for item–DC combinations that were concerned by an alert raised by the *safety stock at DC* features. In total, planners acted on 23% of the store-and-sell volume based on the ML predictions. In Table 4.1, we present the detailed results observed on the impacted tires and compare them to tires on which planners did not

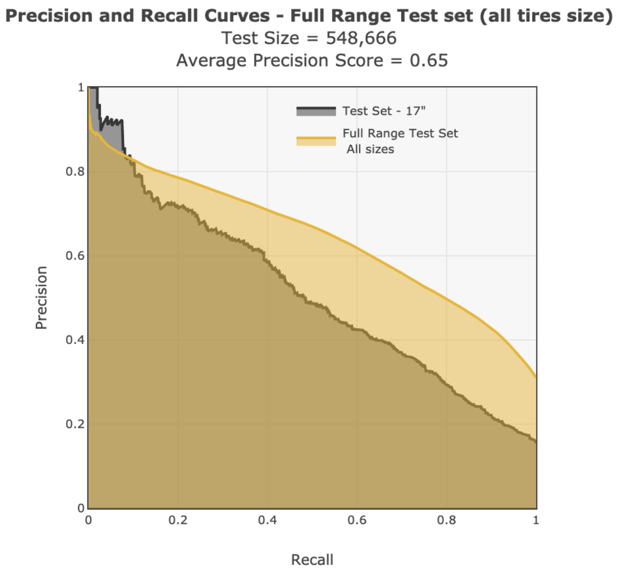


Figure 4.6 Precision-recall curves on a test set containing the full range of tires. The blue area is the initial test set as shown on Figure 4.4. The yellow area displays the performance on all sizes of tires (around 4k items for 11 months of data). The performance is similar, indicating that the model generalizes well to new items.

intervene. We witnessed a gain of 10 percentage points (p.p.) in the service level. To put the gain in context, we also provide numbers for the same period on the variation of demand and days of coverage, i.e., coverage of the stock as compared to the demand. Admittedly, it could be easy to observe an increase in the service level only due to a decrease in the demand, or at the expense of an increase in the total inventory and storage cost. As detailed in the table, this was not the case. First, the decrease in demand was steeper on tires not affected by any corrective action, demonstrating that the gain in service level was not only due to the negative trend in demand. If it were the case, tires not impacted by any action would have seen a natural increase in their service level as well. Secondly, the days of coverage increased only by a small margin, to a level that Michelin considered as both stock and cost-efficient, given the decrease in demand. On top of these results, it was also stated by Michelin that improving the service level also leads to lowering the orders cancellation rate, thus increasing sales and revenue.

Table 4.1 Changes observed over three weeks of corrective actions in service level (in percentage points), demand (%), and days of coverage. Values were measured before and after the prediction horizon, hence three weeks apart. The columns Action display aggregated results on all items impacted by a corrective action based on a ML alert. The columns No Action display aggregated results for the remaining items, thus for which the supply chain plan remained as is. The total is a weighted average according to the volume of tires in each category.

Changes in ...	Service Level (p.p.)		Demand (%)		Coverage (Days)	
	Action	No Action	Action	No Action	Action	No Action
Big Runners	+4	-4	0%	-3%	+1	+6
Medium Runners	+14	+3	-7%	-10%	+2	+9
Long Tail	+14	+5	-2%	-21%	+4	+15
Total	+10	0	-2%	-10%	+2	+8

Notes. ***Volume of Tires.*** *Big runners represent typically around* 6% *of the tires and* 60% *in terms of sales volume, medium runners, respectively* 12% *and* 20% *and finally long tails,* 82% *and* 20%.
Demand and Service Level. *The tests were done during weeks where the demand of tires tend to naturally decrease. However, for tires impacted by the corrective actions, the demand only reduced of 2%, when the service level gained 10 p.p., as opposed to remaining tires which demand reduced of 10%, with a stable service level. This shows that the gain in service level is not only due to the negative trend in the demand.*
Days of Coverage. *For tires affected by actions, days of coverage increased of 2 days only, as compared to an increase of 8 days for not-impacted tires. This could be due in part to the decrease in demand. However, the increase of 2 days of coverage for a decrease of 2% in demand and a gain of 10 p.p. in service level was very reasonable and cost efficient for Michelin.*

4.10 Perspectives

4.10.1 Available Data

It is not yet standard practice, to our knowledge, to archive all of the supply chain data at the finest granularity. For example, at the beginning of our case study, Michelin did not archive the forecast at the item–DC level, nor the *Available To Promise*, a value derived from the fulfillment plan indicating how many items at a DC on a given day have not been allocated yet. This lack of historical led us to discard some data that we believed could have improved the model's predictions. Before ML can significantly enter this space, archiving must become the default policy for supply chain data. Our initiative influenced Michelin's perspective on the importance of archiving data at the most granular level, and in response, they implemented a data lake during the project.

4.10.2 Managerial Insights

The ML approach for failure predictions brings new opportunities for planners to intervene in the supply chain. As such, Michelin needs to adapt its current process so planners can perform manual corrections easily. Today, the available corrective actions for planners are still limited, the main one being adjusting the safety stock at the DC. For example, because Michelin does not perform its forecasting process at the most granular level, it is yet impossible for planners to change the individual forecast of a specific instance. Adjusting safety stock is the most practical lever planners have to re-allocate tires or increase production. Adding new corrective actions could allow planners to react to deviations and other issues more readily.

Supply chain management is complex, and planners typically own only one segment, such as safety-stock specialists, forecasting experts, and logistics planners. As such, it is challenging to gain a product or customer view of the supply chain. At Michelin, our system was the first one to offer a quick holistic view of the supply chain conditions affecting one particular item. It provided an opportunity for different specialists to collaborate and discuss global supply chain mechanisms, as well as specific issues, broadening their understanding of the supply chain. Indeed, the tool instigated regular meetings between planners, who would not interact as frequently otherwise. Our experiments suggest that the tool could also help with the training of new employees since it gives just the right level of information about every stage of the supply chain affecting the service level, and the available levers to react to different disruptions.

4.10.3 Long Term Viability

As manufacturers deploy failure prediction systems at scale, the supply chain performance will tend to improve. The system will catch systemic issues and help refine supply chain planning tools, such as making better forecasts. From a statistical perspective, as fewer failures will occur, the system will face a shift in the distribution of the data. As such, the model will need continuous re-training. We believe that designing such a re-training loop will be one of the main challenges for the long term viability of failure prediction systems.

4.11 Conclusion

We have developed a system that uses ML to predict service level failures in a supply chain. Early on in this project, it has been clear that a good performing algorithm is usually not sufficient to ensure users' adoption. It needs to be paired with a system that builds confidence and understanding of the model. Our results and the adoption by the planners show the potential of ML systems to complement existing systems for supply chain management. We believe that this type of approach can be applied to more complex supply networks and other areas such as production planning.

As this system gets used in production over an extensive period, it will probably trend towards identifying less structural changes for more fine-grained issues. A natural extension would be to perform corrective actions based on the predicted failures automatically so that the supply chain becomes a *self-learning* entity, dealing with deviations in *autopilot* mode. As the availability of data improves, such initiatives will lead the way to a new era in supply chain management.

4.12 Acknowledgement

We wish to thank Emmanuel Cadet of Michelin for his constant support throughout this project: it led to valuable learning on both sides. Merci beaucoup!

4.13 Appendix: Data and Parameters

4.13.1 Data Format

For each m historical instances, a set of n features (f_n) is computed at the item, DC, and week level. The complete list of features can be found in the next section, Features Set. To indicate failures, aggregated service levels are computed with the mean of the corresponding orders weighted according to the product quantities. These aggregated service levels are then compared to the global service level target and converted into a binary variable, y. If the service level is below the target, it is considered as a failure (1), else a 0. In Table 4.2 is shown the data format. Note that the product, location, and week columns were not used explicitly as features, hence the model can be used with new location and products. The columns in grey were hence dropped before running the model.

4.13.2 Features Set

Below we detail typical features that could be useful for the task of predicting service level failure. In Table 4.3, we list the specific features that we used as part of this project, and in Table 4.4, statistical measures for each feature.

First, *production* problems can be good indicators of future failures since the lead time between the plant and the DCs delays their impact, allowing us to foresee potential issues

Table 4.2 Data format used in the model.

	Product	Location	Week	f_1 f_2 ... f_n		Failure
x_1	AXP	9272	$W1$		y_1	1
x_2	BFR	875	$W1$		y_2	0
x_3	AXP	9272	$W2$		y_3	0
...	...	...	...		...	...
x_m	GFT	7654	$W700$		y_m	1

Note. Grey columns were dropped out of the model, so the system can produce alerts for new products and locations.

and delays. Relevant features include (1) production plan vs. actual production; (2) inventory on-hand compared to latest forecast; and (3) percentage of production capacity achieved.

Second, between the plant and the DC or between the DC and the ultimate consumer, *logistics* problems may occur in the transportation network or the loading and unloading, delaying the shipment of the items. Although essential to understand past failures, transportation disruptions are less likely to predict future ones. For example, a delay may be weather-related and likely to be resolved within the time horizon. Relevant features include (1) average logistics delays in last period and (2) percentage of logistics capacity achieved.

Lastly, *forecasting* deviations and *stock* problems at the DC may indicate that the plan is no longer meeting the demand. Relevant features include (1) cumulative forecasts of the last few periods compared to the customer orders, indicating over- or under-forecasting and (2) inventory on-hand vs. safety-stock target.

In Table 4.3, we detail the complete list of features used in the final model. Each feature type corresponds to either an item, DC, plant, or combination of those, as noted in column *Aggregation*. We computed each feature type for different time steps, from the three weeks preceding the instance to the three weeks after (for some projections), as detailed in column *Time*. The column *Nb* indicates the number of features as a result of the different time steps for each feature type. Lastly, the column *Underlying Cause* indicates the mapping between feature types and the features family. In particular, these mappings are used to produce the cumulative contributions graph in Figure 4.3, as well as to categorize the alerts by cause in the UI to accelerate their resolution. In total, our model used 35 features.

In Table 4.4, we display the distribution for each feature on the training set, thus on 17" summer tires for the first 12 months of data (2016-09-01 to 2017-09-20). We show the mean and variance, important percentiles, as well at the percentage of NaN and infinite values.

Table 4.3 Features set used in the final model, according to data availability.

Feature Types	Aggregation	Time	Nb	Underlying Cause
(1) Production Plan vs Actual Production	Item-Plant	t0	1	Production
(2) Inventory and Production Plan vs Needs	Item-Plant	t0:t+2	9	Production
(3) Inventory vs Safety Stock	Item-Plant	t-2:t+2	5	Production
(4) Average logistic delays	Item-Plant-DC	t-3:t0	4	Logistic
(5) Total Stock (all items) vs Capacity	DC	t0	1	Logistic
(6) Inventory vs Safety Stock	Item-DC	t0:t3	4	Safety Stock at DC
(7) Projected Safety Stock vs current Safety Stock	Item-DC	t1:t3	3	Safety Stock at DC
(8) Past service level (Volume tires)	Item-DC	t-3:t0	4	Unknown
(9) Past service level (Count orders)	Item-DC	t-3:t0	4	Unknown

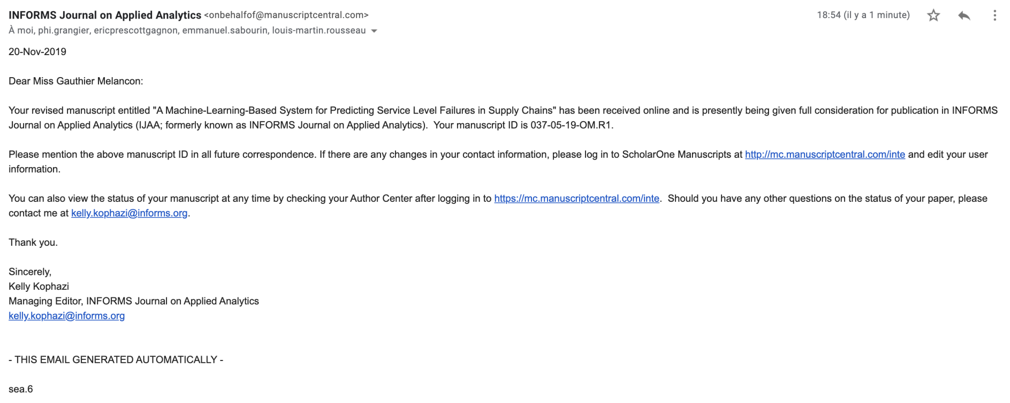
Notes.
For feature type (2), 3 different needs projections were used for each time step, resulting in 3x3 features in total.
In the first column, "vs" represents the relative difference between both values. As such, these features are ratios.

4.13.3 Algorithm

We selected the following hyper-parameters for the GBDT using XGBoost: n_estimators=75, learning_rate=0.1 and max_depth=5. We kept the other hyper-parameters at their default value.

For CV parameters used in the first phase when building the model, we used three splits. In Table 4.5, we detail the dates that we used to create the splits.

4.14 Acknowledgement of Receipt



INFORMS Journal on Applied Analytics <onbehalfof@manuscriptcentral.com> 18:54 (il y a 1 minute)
À moi, phi.grangier, ericprescottgagnon, emmanuel.sabourin, louis-martin.rousseau

20-Nov-2019

Dear Miss Gauthier Melancon:

Your revised manuscript entitled "A Machine-Learning-Based System for Predicting Service Level Failures in Supply Chains" has been received online and is presently being given full consideration for publication in INFORMS Journal on Applied Analytics (IJAA; formerly known as INFORMS Journal on Applied Analytics). Your manuscript ID is 037-05-19-OM.R1.

Please mention the above manuscript ID in all future correspondence. If there are any changes in your contact information, please log in to ScholarOne Manuscripts at http://mc.manuscriptcentral.com/inte and edit your user information.

You can also view the status of your manuscript at any time by checking your Author Center after logging in to https://mc.manuscriptcentral.com/inte. Should you have any other questions on the status of your paper, please contact me at kelly.kophazi@informs.org.

Thank you.

Sincerely,
Kelly Kophazi
Managing Editor, INFORMS Journal on Applied Analytics
kelly.kophazi@informs.org

- THIS EMAIL GENERATED AUTOMATICALLY -

sea.6

Figure 4.7 Email confirming the article was submitted to IJAA, and revised on 2019, November 20th.

Table 4.4 Distribution of features used in the model, computed for the training data.

				Percentiles					%	
Type	Time	Mean	Var	Min	25	50	75	Max	nan	inf
1	t0	-0.6	9.5	-324.0	-0.1	0.0	0.3	4.0	0.5	0.0
2	$t0_1$	463.1	2013.5	0.0	2.3	8.7	28.0	9999.0	0.3	0.0
2	$t0_2$	348.9	1788.6	0.0	1.8	4.5	11.6	9999.0	0.3	0.0
2	$t0_3$	320.8	1738.7	0.0	1.6	3.1	6.8	9999.0	0.3	0.0
2	$t1_1$	479.7	2050.9	0.0	2.3	8.7	27.8	9999.0	0.4	0.0
2	$t1_2$	356.2	1808.2	0.0	1.8	4.5	11.6	9999.0	0.4	0.0
2	$t1_3$	329.8	1763.5	0.0	1.6	3.1	6.9	9999.0	0.4	0.0
2	$t2_1$	486.2	2064.1	0.0	2.3	8.8	27.3	9999.0	8.4	0.0
2	$t2_2$	373.1	1851.8	0.0	1.8	4.6	11.7	9999.0	8.4	0.0
2	$t2_3$	344.6	1798.2	0.0	1.6	3.1	7.1	9999.0	8.4	0.0
3	t-2	116.4	526.6	0.0	5.8	27.2	74.2	31680.0	0.3	0.0
3	t-1	110.9	524.6	0.0	5.6	26.6	72.0	31802.0	0.3	0.0
3	t0	109.2	519.2	0.0	5.4	25.7	70.6	31802.0	0.3	0.0
3	t1	107.6	486.3	0.0	5.4	25.2	69.8	31802.0	0.4	0.0
3	t2	119.0	820.7	0.0	5.3	24.2	66.7	31680.0	8.4	0.0
4	t-3	-1.4	2.9	-22.0	-2.8	-0.9	0.0	28.0	0.3	0.0
4	t-2	-1.4	3.0	-22.0	-2.9	-0.9	0.0	28.0	0.3	0.0
4	t-1	-1.4	3.0	-22.0	-2.9	-0.9	0.0	28.0	0.3	0.0
4	t0	-1.4	3.0	-22.0	-2.8	-0.9	0.0	28.0	0.3	0.0
5	t0	0.8	0.1	0.3	0.7	0.8	0.9	1.2	0.3	0.0
6	t0	1.3	8.4	-471.4	0.3	0.9	1.5	772.0	0.0	0.1
6	t1	1.2	7.7	-471.4	0.3	0.9	1.5	772.0	0.0	0.1
6	t2	1.4	7.0	-463.6	0.3	0.8	1.6	386.0	0.0	0.1
6	t3	1.5	7.2	-458.8	0.3	0.8	1.6	386.0	0.0	0.1
7	t1	1.0	0.2	0.0	1.0	1.0	1.1	3.0	0.1	0.0
7	t2	1.1	0.4	0.0	1.0	1.0	1.1	8.0	0.1	0.0
7	t3	1.1	0.5	0.0	1.0	1.0	1.2	10.0	0.1	0.0
8	t-3	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.4	0.0
8	t-2	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.4	0.0
8	t-1	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.4	0.0
8	t0	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.3	0.0
9	t-3	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.4	0.0
9	t-2	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.4	0.0
9	t-1	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.4	0.0
9	t0	0.9	0.3	0.0	1.0	1.0	1.0	1.0	0.3	0.0

Note. The columns Type and Time map to Table 4.3 columns. The two last columns quantify the % of NaN and infinite values for that feature.

Table 4.5 Cross-Validation splits used to train and test the model.

Train Splits			Test Splits		
Start Date	End Date	Nb of rows	Start Date	End Date	Nb of rows
2016-09-01	2017-04-06	21,396	2017-04-07	2017-05-31	6,019
2016-09-01	2017-05-31	27,415	2017-06-01	2017-07-31	5,920
2016-09-01	2017-07-31	33,335	2017-08-01	2017-09-20	4,336
2016-09-01	2017-09-20	37,671	2017-09-21	2017-12-12	6,133

Note. The test set from the last split is what we called our test set for the Phase 1.

CHAPTER 5 PREDICTING CONFIDENCE ESTIMATES FOR AN ARTIFICIAL INTELLIGENCE SYSTEM

5.1 Problem Description

In this section, we present Artificial Intelligence (AI) systems and introduce a specific example that was built to extract information from physical documents in a data entry process. We finish by discussing sources of uncertainty in the system and why it is useful to quantify them.

5.1.1 Artificial Intelligence Systems

AI models gained in popularity over the last decade and are showing some promising results in various particular tasks, from image recognition to playing chess. Typically, scientists optimize AI models for a given dataset according to a defined metric, separately from the context in which the model will operate. Indeed, when AI models are put in production to solve concrete business needs, they are rarely used in isolation. For example, self-driving cars require a complex system where multiple models and agents interact to achieve various tasks. As such, a AI model, when analyzed individually, may behave very differently than when examined in interaction with others. In this recent paper in Nature, Rahwan et al. [8] claims that AI systems need to be studied at different scale so they can be rightly understood: the individual level, meaning the algorithm itself, the collective level, where multiple models interact together, and finally the hybrid level, where we also consider the interactions with the humans.

In this thesis, we consider a AI system as being a system where at least one of its elements is a AI model, which englobes Machine Learning (ML) and Deep Learning (DL) algorithms. Typically, this model can interact with a diverse set of components, such as other AI models, OR algorithms, heuristics, physical processing units, and humans.

5.1.2 Current Data Entry Process

Data entry clerks generally extract information from documents through a manual process. They first read each document and then retrieve the required information in some computer systems. The task is tedious and repetitive. A AI system can improve clerks' experience and speed by automatically retrieving the information on the document. This way, they can shift their focus from manual entry to validation tasks and tackle the more challenging documents only.

Extracting information from a document is a trivial task to humans. Nevertheless, it does revolve on multiple capabilities. We use our vision to recognize that something is a document with information on it. We may use our memory to identify the document if we have seen similar examples in the past. We then locate the text and read it. Based on the meaning of the text, we finally use our judgment to determine what information is essential. Thus, although we have never seen a form before, we can quickly infer the name of the person who completed it.

5.1.3 Using Artificial Intelligence to Extract Information: Form Extractor

For a AI system to perform the same task, it requires different specialized models. In our work, we experimented with a system designed to extract fields from a scanned or photographed completed form, for which we have a predefined template. The system is called Form Extractor (FE) and consists of the following steps. The first model aligns the form as well as resizes it if needed. Secondly, a document classifier identifies what stored template matches the form if any. The fact that FE can only retrieved information for a set of known templates is a limitation of the system. Templates contain the information of the required fields to extract and their location in the document. In parallel to the document classifier, the third and fourth steps respectively localize the text on the form and recognize it, using Optical Character Recognition (OCR) models. Based on the location of the fields stored in the template, FE extracts the correct information for each field. Some post-processing can then be applied, if needed, such as splitting or merging text into different fields.

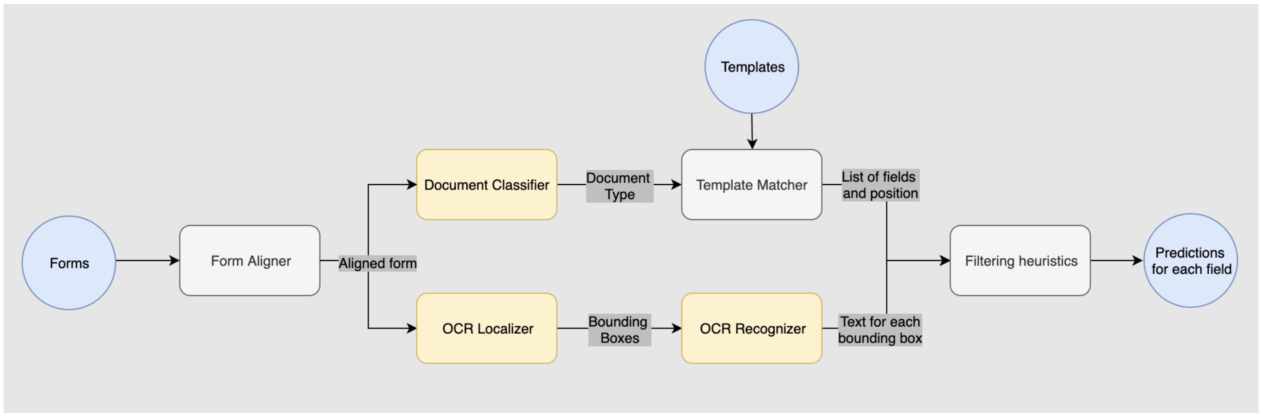


Figure 5.1 Different components of the Form Extractor system.

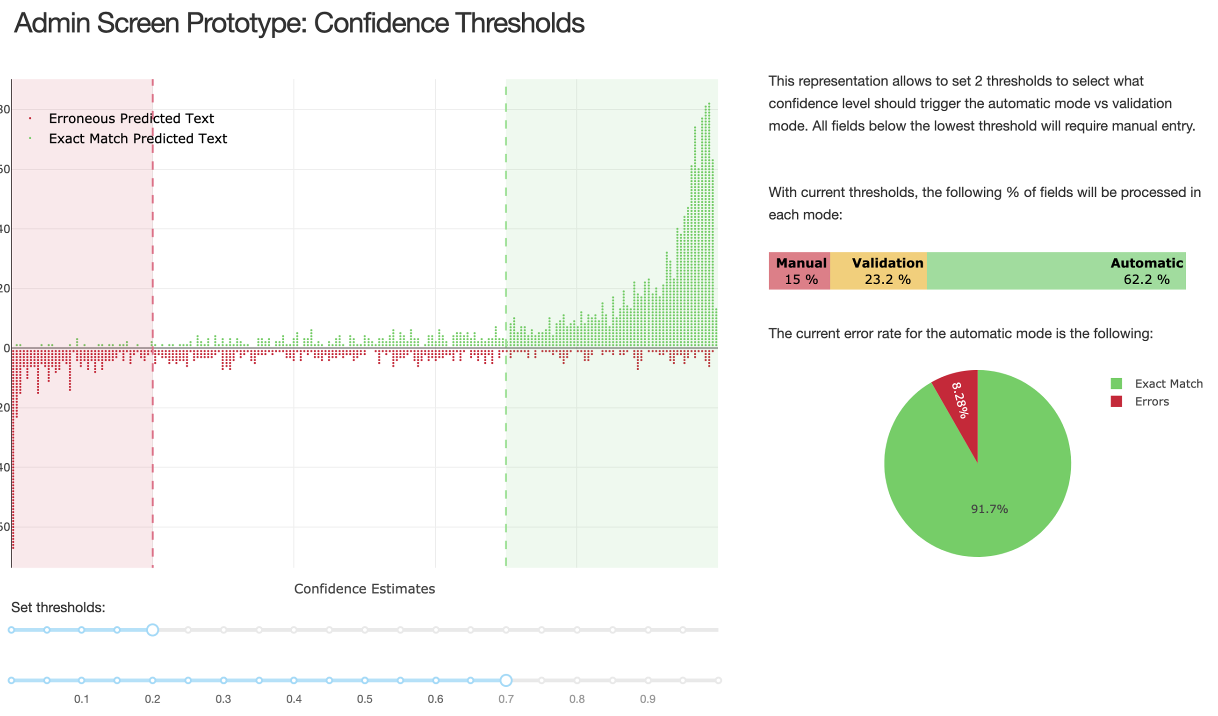
5.1.4 Uncertainty Quantification in Form Extractor

In the FE system, different sources of uncertainty exist. First, the data can be noisy; images can be blurry or contain irrelevant scribbles overlaid on top. Additionally, the data can be mislabeled, meaning that information that needs to be retrieved is not well identified. Secondly, each model in the system has its own aleatoric and epistemic uncertainty, since they were each trained individually on noisy and incomplete datasets. These sources of uncertainty affect the whole system's ability to retrieve the information present on a form correctly. Their cumulative effects are non-linear on the system's overall uncertainty, as their interaction can amplify or reduce their impact.

As discussed in the Introduction, quantifying uncertainty in systems is essential to build the user's trust and improve the system's performance. However, one of the most significant opportunities is the identification of uncertain predictions which require the attention of a Human-in-the-loop (HITL). As part of this project, we define confidence estimates as a measure of the likelihood of a system being successful (instead of predicting system failure). Based on the system's confidence, different workflows for a HITL can be triggered, as explored in Figure 5.2. An automatic mode could be in place for predictions with high confidence, where no user is involved or where only a few random predictions are checked to verify the performance of the system. Lower confidence predictions could enable two different workflows, one for medium confidence cases where users validate each prediction of the model, and another where no prediction would be made, thus requiring the user to retrieve the text manually. The rationale behind this last workflow is that it is easier for users to manually perform the task, as they would typically do in their current workflow, without having to delete the erroneous prediction beforehand. We could imagine other scenarios and workflows than the ones proposed here. Moreover, the configuration of the confidence thresholds triggering the different workflows is not straightforward. The optimal settings depend on the cost associated with errors in each mode vs. the benefits that the workflow yields. For instance, the automatic mode may not be expensive since it requires minimal intervention. However, errors still need to be expected even for high confidence cases, such as with adversarial examples. Thus, there is a necessary trade-off that needs to be accounted for when configuring those thresholds and workflows.

5.2 Methodology

In this section, we present how we used a ML model to generate confidence estimates for every system's predictions. We detail the model's objective, the type of data that was available



Admin Screen Prototype: Confidence Thresholds

This representation allows to set 2 thresholds to select what confidence level should trigger the automatic mode vs validation mode. All fields below the lowest threshold will require manual entry.

With current thresholds, the following % of fields will be processed in each mode:

Manual	Validation	Automatic
15 %	23.2 %	62.2 %

The current error rate for the automatic mode is the following:

Set thresholds:

Figure 5.2 Example of screen to set different workflows based on the system's confidence.

Notes. *The typical user for such a screen would be a system's administrator that needs to configure the thresholds based on some business logic. The thresholds would enable different workflows for the data entry clerks. The main plot displays the system's performance on a representative dataset. Each dot represents a prediction, where the x-axis indicates their corresponding confidence estimate. The y-axis displays the amount of prediction at each confidence level. The two data series, in reflection on the x-axis, represent respectively data points which were extracted correctly by FE, in green and the opposite in red, i.e., FE system failures. If confidence estimates are reliable, green dots are expected to be as much as possible on the right, and red ones on the left. The three different areas created by the two thresholds map to the different workflows: manual, validation, and automatic. In the right panel, the expected consequences of the selected thresholds are shown, such as the number of predictions falling in each mode, and the number of estimated errors in the automatic mode.*

to perform the task, and the features we engineered. We also briefly mention additional features that could have been useful in other contexts or if further data was available. We end this section by presenting the selected algorithm and the methodology that we followed to evaluate the performance of the model.

5.2.1 Overview of the Approach: the Full-Stack Uncertainty Model

Typical methods for quantifying uncertainty in AI models, such as the ones presented in the LITERATURE REVIEW, cannot be applied directly to a AI system composed of multiple models. They are usually designed for individual models or specific architectures. As such, we approached this problem with a similar idea than the one presented in the previous chapter on supply chains. The approach consists of building a classifier that looks at features from every stage of the system to predict whether the system is correct. As part of this project, we named this model the Full-Stack Uncertainty Model (FSU). This model returns the confidence associated with each prediction of an underlying AI system composed of a stack of models.

5.2.2 Model's Task

The goal of the FSU is to predict how likely the underlying system is predicting the right outcome. The definition of a right outcome can vary based on the context. For instance, the user may be indifferent if the edit distance of the predicted text is within two (two characters off). In our experiments, we consider that users are only interested in exact match predictions (edit distance of zero), thus that anything else is a system failure. For the FSU to be able to learn what influences the system to fail, it needs examples of the system's successes and failures. Hence, we benchmarked datasets on the system and captured if the predicted text is an exact match or not. These binary values become the FSU's target (y). As such, the FSU is a classifier that learns how likely a given prediction is an exact match, and as such, outputs confidence estimates for the underlying AI system.

5.2.3 Available Data

While benchmarking a dataset, we can collect different types of information throughout the system, which in turn can be used to derive features for the FSU. First, a type of available information is the input data used by the underlying models, such as the images of the forms. Second, another type of information available is the predictions made by any model that is part of the system, such as bounding boxes from OCR Localizer, as well as the final cumulative prediction of the system, in our case, the predicted text. Lastly, any metric representing the state of each model at the moment of prediction can also be collected (such as the estimated uncertainty of a model's prediction, if available).

In practice, it is not easy to have access to all this information in productized environments. AI systems are complex and composed of multiple components, and as such, well-defined Application Programming Interfaces (APIs) need to be in place for each model so they can easily interact with one another and be maintainable. However, this reduces the overall flexibility of the system, as it is difficult to extract further information than what the APIs provides. In our experiments, we solely had access to the following data to create features for the FSU: the field name, the global prediction (predicted text), and its label (the information that needs to be retrieved - targeted text), from which we can infer if the prediction is an exact match. Additionally, one model of FE, OCR Recognizer, outputted its soft-max probabilities along with its prediction and was readily available in the model's API. In Table 5.1 we present fictive examples of the available data. In the next sub-section, we present the engineered features computed from this data.

Table 5.1 Data available to train the Full-Stack Uncertainty Model model

	Predicted Text	Targeted Text	Field Name	OCR Prob.		Exact Match
x_1	Samuel	Samuel	Name	0.99	y_1	1
x_2	HOE 1V4	H0E 1V4	Postal Code	0.02	y_2	0
x_3	Jonn	John	Name	0.98	y_3	0
...	...	...	...	...	...	...
x_m	21	21	Birth Day	0.97	y_m	1

5.2.4 Features Engineering

The FSU approach allows us to perform features engineering to improve the confidence estimates based on intuitions or evidence as to what causes the AI system to fail. As such, to uncover a useful set of features, we can perform an error analysis of the underlying system and discover what its typical mistakes are. Below we present the set of features that we built as a result of an error analysis performed on a dataset combining three forms, presented later in sub-section 5.3.1.

Uncertainty of OCR Recognizer

The most frequent source of errors in FE is due to OCR Recognizer. It is understandable since it is the hardest task of the system. For that reason, before the FSU approach, FE used the soft-max probabilities of OCR Recognizer as a proxy for the confidence of the complete system. It explains why they were readily available in FE's API, as mentioned above. As such, we used OCR recognizer confidence estimates as a baseline to evaluate the performance of the FSU Model, which we will cover later.

OCR recognizer is a DL model. As discussed in the LITERATURE REVIEW, DL model predictions tend to be poorly calibrated, and even their relative ranking cannot always be trusted. It was the case with OCR Recognizer at the beginning of our experiments; the model's outputs, soft-max probabilities, were either very close to 0 or 1, as pictured in Table 5.1, hence not good representatives of the model's confidence. Some experiments were made to improve them with methods such as temperature scaling, KL Divergence, and MC

Dropout. As a feature for the FSU, we used confidence estimates generated by a modified version of OCR Recognizer using a temperature scaler. As such, the confidence measures were better distributed than those shown in Table 5.1.

Output Format Validation

In the FE system, no underlying model is aware of what format is valid for each field. OCR Recognizer is trained independently on an extensive OCR dataset and knows nothing of the particular fields for which it is extracting text. Hence, in the situation where the OCR model would hesitate between different potential predictions for a given text, such as 31 or 81 for a date field, some notion of what a date is could help improve its prediction. Adding this knowledge to the OCR model is not an easy task and could be tackled with constrained learning. As this was not in place at the time of our experiments, we computed features that could describe the format of the predicted text, such as the percentage of alphabetic characters in the string, so that the FSU can learn what the usual format for each field is. In Table 5.2 are shown examples of such features. The complete list is detailed in Appendix A. For the FSU to learn what output is valid for each field, we also added the field name as a feature, using a one-hot encoding. However, this reduced the ability of the FSU to generalize, which we addressed with pre-trained subword embedding, as next discussed.

Table 5.2 Examples of output format validation features

	Predicted Text	Field Name	%_alpha	%_uppercase	number_value
x_1	Samuel	Name	100	17	NaN
x_2	HOE 1V4	Postal Code	57	100	NaN
x_3	Jonn	Name	100	25	NaN
...	...	...	...	...	...
x_m	21	Birth Day	0	NaN	21

Notes. *The grey column is not a feature. %_alpha represents the amount of alphabetic characters in the predicted text, %_uppercase represents the amount of uppercase characters and number_value represents the actual number that was predicted, if applicable.*

Pre-trained Subword Embeddings

Adding a one-hot encoding representation of the field name as a feature brought one challenge: the FSU model could not generalize to forms with new field names at inference time. Nonetheless, since field names tend to be composed of words and abbreviations, which can indicate their content, we can use a richer representation of the field name than a sparse one-hot encoding. As an example, fields representing *days* tend to contain the word *day*, same for *names* and *postal codes.* For this reason, we leveraged a pre-trained subword embedding model from fastText [59] to generate dense vector representation for field names. Subwords are all the substrings held in a word, with a defined minimal and maximal number of characters (between 3 and 6 for fastText). Since this pre-trained model uses subwords instead of words, it can represent many tokens not present in the embedding training dataset (Wikipedia and Common Crawl). As such, the model can generate a vector for new field names. By providing semantic information on the fields to the FSU model, we made it possible to use the model on new forms and fields, assuming new field names come from a similar distribution. We will discuss such limitations in sub-section 5.4.

Using pre-trained subword embeddings also offered another benefit. When training the FSU, we did not have access to a big dataset. Thus, we did not have access to many examples per field. As fastText embeddings were pre-trained on massive datasets that can roughly capture the semantics of the English language, we were able to enrich the dataset by representing the similarity in different field names. For instance, if multiple birthday fields are on a form, the field names would typically be *birth_day_1*, *birth_day_2* and *birth_day_3*. With a one-hot encoding representation, the FSU cannot leverage the examples of one field to learn on the other ones. It is particularly crucial for some fields which are typically empty on forms. For example, *birth_day_3* could only be present when a person has more than two children. Hence, very few instances of these fields exist in the training set, and examples from *birth_day_1* and *birth_day_2* should be leveraged. Pre-trained subword embeddings allowed us to do that. Even with the atypical formats of field names, which contain a lot of abbreviations and symbols such as "_", we were pleased to observe empirically that the cosine similarity was generally high for similar field names.

5.2.5 Additional Potential Features

In our experiments, we were limited by the data which was available in the FE's API. Additionally, when performing the error analysis on the FE system to identify useful features, some error types rarely occurred on our dataset. As such, engineering features for these situations were not likely to offer a large gain in performance. As an example, it was quite rare that the form aligner step would fail, thus adding a feature that would measure the uncertainty of this step would not have improved the performance of the FSU significantly. Nevertheless, we present below two broad types of features that could generally be helpful on different datasets.

Features describing the system's input

Some features can characterize the input of the system, so the FSU can learn how the data impacts systems' failures. These features can measure if the data has the expected format or quality. As examples, features can detect if the image has the right size, is blurry, or is Out-Of-Distribution (OOD). Additionally, if information on the data generation process is available, such as meta-data, it can also serve as good candidates for the FSU. As examples, for photographed forms, it could be information about who took the picture. In our experiments, the data was quite clean. As such, we did not need to engineer features on the system's input. However, a different dataset could require such features.

Features describing models' states and outputs

We can also use different features that describe either the internal state of the models, their prediction, or their uncertainty estimates. In our current experiments, only OCR Recognizer provides an estimation of its uncertainty. If the other models would also provide such estimates, the FSU could learn how they interact. Other features could also describe internal states of the models, such as the values of one model's layer. Additionally, predictions from the model, or a representation of them, could also be features for the FSU. In our example, we could have used the bounding boxes from OCR Localizer as features, so the FSU learns when the position of the text is odd.

5.2.6 Algorithm

Different algorithms could be good candidates to model the FSU. Common classifiers are logistic regressions, Support-vector Machines (SVMs), random forests, GBDT and neural networks. We could also explore more complex models. For example, an interesting avenue could be to model the FSU with a Bayesian approach, where we account for the uncertainty of the FSU itself. However, since we did not have a big dataset for our experiments, we used GBDT [55], as they are simple to implement and can easily handle features with *NaN*. We also had previous experience with them, as presented in Chapter 4. We used the XGBoost implementation [56]. Another benefit of using this model is its ability to be explainable using techniques such as SHAP [57, 2], as also demonstrated in Chapter 4.

5.2.7 Evaluation Methodology

Metrics

We selected different metrics to evaluate the performance of the FSU. First, we selected metrics typically used to evaluate the quality of uncertainty estimates in classifiers, the Area Under the Receiver Operating Characteristic (AUROC), and Negative Log-Likelihood (NLL).

In contrast to FE's objective - which is to predict text - the explicit goal of the FSU is to measure how confident we are in those predictions. Therefore, the calibration of FSU outputs is crucial, and it needs to be consistent across different fields and forms. By calibration, we mean that predictions falling in a given bin of confidence estimates, such as between 80% and 90% confidence, should have a matching accuracy, in this case, 85%. For that reason, we added as a metric the Expected Calibration Error (ECE), computed for ten bins, which measures precisely that.

Baseline

Establishing a comparative baseline for the FSU is not straightforward since FE does not output any confidence estimates by default. However, as stated before, in practice, FE uses OCR recognizer confidence estimates as a proxy for the confidence of the overall system. Consequently, we used these as a baseline to compare our FSU results.

5.3 Results

We present the results of our experiments in this section. Our work is still early, and more research needs to be done to explore how the approach can generalize in various conditions. We expose some of these limitations at the end of the section.

5.3.1 Experiments Details

For our experiments, we had access to a few different datasets which were hand-labeled in-house by a team of data annotators. Table 5.3 presents characteristics of the datasets. We benchmarked these three different types forms on FE and used the results to build a dataset to train and test the FSU. Each form has its own set of fields.

We performed Cross Validation (CV) with five splits to assess that the performance is constant across different train and test splits. The test set has the same distribution as the training set in the number of examples from each field. We did not perform any exhaus-

Table 5.3 Available datasets to train and test the Full-Stack Uncertainty Model

Form ID	Nb of forms	Nb of fields per form	Avg. nb of filled fields per form	Total nb of fields	Total nb of filled fields
Form A	235	147	55	33,665	13,118
Form B	238	66	18	4,443	4,443
Form C	247	137	22	5,607	5,598
Total	720	350	97	43,715	23,159

tive hyper-parameters tuning, as our results were already beating the baseline. Additionally, we removed from the dataset null predictions (text is an empty string), as we have no features that can help in these cases; all features are *NaN*. It caused unnecessary noise in our performance metrics.

We performed a few different experiments, with different features and settings. In the next section, we focus our analysis on our most conclusive experiment, using the set of features described earlier in sub-section Features Engineering and detailed in Appendix A. We also present aggregated results for our experiment using a one-hot encoding representation for field names, instead of pre-trained subword embeddings. We then compare our method to the mentioned baseline.

5.3.2 Model Performance

In Table 5.4, we show aggregated performance metrics. We present the results for both train and test for each of the four selected metrics. The last two columns, *FE Accuracy*, display the performance of the underlying model, FE, on each subset, to put other metrics in perspective. We can observe that the accuracy changes from a CV split to another, thus impacting the performance of the FSU. Generally, as the test set accuracy drops, the FSU tend to perform better as measured by the AUROC, but not according to NLL. More investigation would be needed to understand why this is the case. The ECE metric shows that generally, the FSU outputs are well-calibrated. More research needs to be done to see if this holds across different fields.

In the *Avg.* row, we can see that the FSU model surpasses the performance of the baseline by a fair margin (*Base.* row). As a reference, we also display the aggregated performance over the same CV splits when using one-hot encoding. (*N-emb.* row). We can observe that it is performing slightly worse, thus showing that semantic features improved the results, on top of making the model more generic.

In Figure 5.3 we present the predictions of the FSU on the 5^{th} CV split. The representation

Table 5.4 Performance of the Full-Stack Uncertainty Model on the combined dataset using 5 splits of Cross Validation

	AUROC		NLL		ECE		FE Accuracy	
	[0,**1**]		[**0**, ∞[		[0,**1**]		[0,**1**]	
Splits	Train	Test	Train	Test	Train	Test	Train	Test
CV 1	0.92	0.85	0.27	0.22	0.04	0.03	0.80	0.90
CV 2	0.92	0.87	0.27	0.24	0.03	0.03	0.80	0.88
CV 3	0.92	0.89	0.26	0.30	0.03	0.02	0.82	0.81
CV 4	0.91	0.90	0.25	0.33	0.03	0.03	0.84	0.75
CV 5	0.91	0.91	0.25	0.31	0.03	0.03	0.83	0.76
Avg.	0.92	**0.89**	0.26	**0.28**	0.03	**0.03**	0.82	0.82
N-emb.	0.90	0.87	0.29	0.30	0.04	0.03	0.82	0.82
Base.	0.63	0.61	0.44	0.52	0.11	0.08	0.84	0.77

Notes. *In the header, below each metric, we display the corresponding range of values, and in bold, the optimal value. The* ***Avg.*** *row shows the average performance across the different CV splits for our main experiment. The* ***N-emb.*** *row shows the results when running the FSU without the pre-trained subword embeddings, thus using a one-hot encoding for field names. The* ***Base.*** *row shows the performance of the baseline, OCR recognizer confidence estimates, on the* 5^{th} *CV split.*

is the same as the one presented in Figure 5.2. We can observe that the model generally generates high confidence estimates for fields that are extracted correctly: few green dots are on the left section of the graph. However, there is a fair amount of red data points in the right portion of the graph. Those data points are the most problematic ones, as they represent cases where the FSU is very confident in the FE predictions, but the FE is wrong. Nevertheless, when looking empirically at some of these data points, we noticed that a fair proportion of them was due to labeling mistakes, hence not errors caused by the system. As anecdotal examples, the top error is for a year field, which label has five digits instead of four, and in the two following errors, there is an extra space at the end of the label. We could not easily evaluate what proportion of errors were due to labeling, as we would need to look at each form to spot these errors manually.

5.3.3 Explainability of the Results

We used the SHAP library [57, 2] to explain the FSU predictions. We present the SHAP summary plot in Figure 5.4, which displays the most influential features across the dataset on the y-axis. The x-axis represents the SHAP value of that feature for each data point in the dataset. The SHAP value measures the impact of a feature on the model output (here confidence estimates) for a given prediction. The pink vs. blue color-coding represents for each data point if the feature value is high (pink) or low (blue).

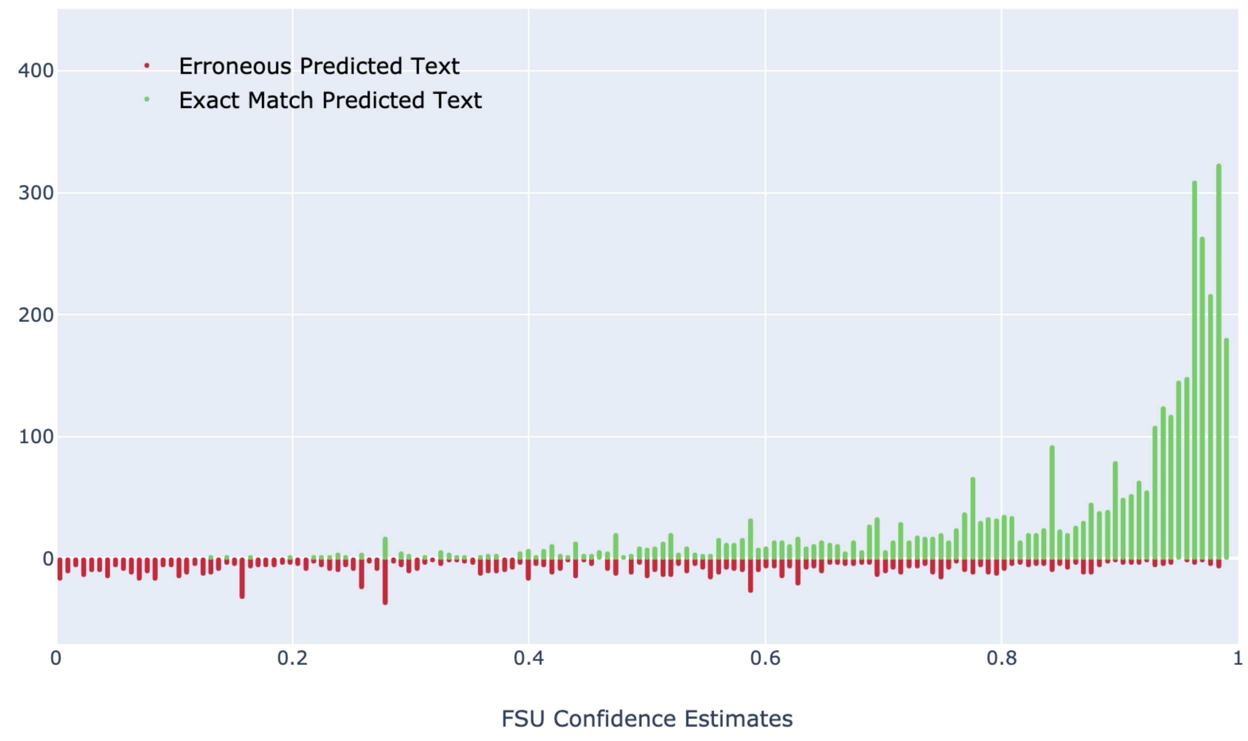


Figure 5.3 Confidence estimates predicted by the Full-Stack Uncertainty Model on the 5^{th} Cross Validation split

The most powerful feature throughout the dataset is the confidence of OCR Recognizer (*conf*). For data points with high OCR confidence values (pink dots), theFSU confidence estimates tend to be high, and vice versa. The second most important feature is the length of the predicted text (*output_length*). The longer the predicted text is, the less confident the FSU is in the FE prediction. One way to explain this could be that as the length of the text increases, there are more chances for the OCR Recognizer to make a mistake. It could also be that the system outputs longer predictions when it is not localizing the right text. The third, fourth, and fifth features all characterize the format of the predicted text, either in terms of % of alphabetic characters, % of uppercase, or the actual predicted value, if numerical. We can notice that the correlation between the pink/blue dots and positive/negative SHAP values is not as strong as with the *conf* feature. It could be because it is hard to quantify positively or negatively the impact of these features in isolation. For example, a *perc_alpha* of 0 for a name is odd, but not for a date. It would be interesting to dive further in this analysis to understand detailed correlations that the FSU learned.

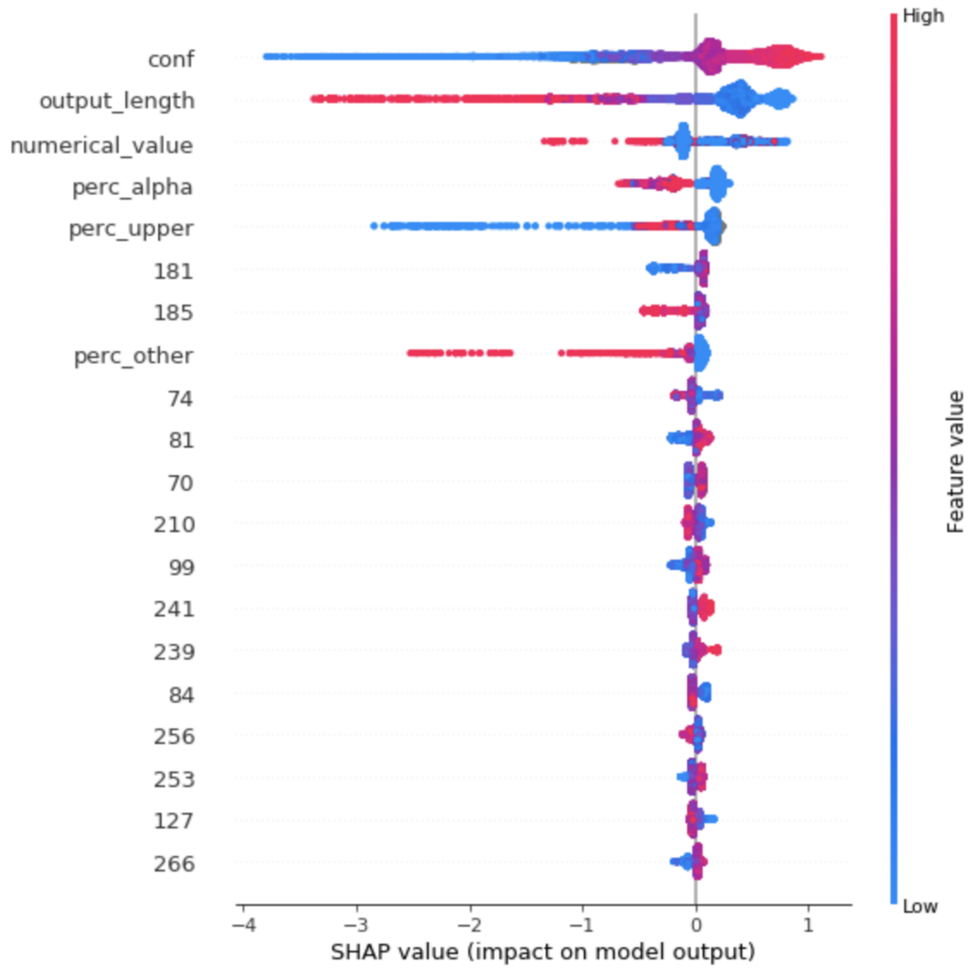


Figure 5.4 SHAP summary plot displaying the most influential features and their impact on the model output

Notes. *Features named with a number are the pre-trained subword embedding features.*

5.4 Limitations and Future Work

Although our experiments show promising results and surpass the baseline, many limitations exist in our current approach. Amongst other things, the results should be analyzed and understood further, and we should investigate performance metrics per fields. Below, we present a list of limitations and ideas that our experiments would benefit from.

5.4.1 Training Data

First, we were limited in the amount of data available to conduct our experiments. The difficulty lies in the labeling of the forms, which requires manual work. Acquiring a bigger dataset would surely help the FSU to be more robust but would also allow testing of more complex features and models. Additionally, the dataset should contain a more diverse set of forms, to test that the FSU generalizes well to different types of fields.

The dataset is also unbalanced, as they are fewer instances of system failure than success. With the size of our dataset, it was especially a challenge for the FSU. The fact that there are only a few examples for each field, and sometimes very few errors, limits theFSU in its ability to learn specific format for certain fields, such as postal codes. In earlier experiments, we trained the FSU on a smaller set of fields, and it learned almost perfectly the format of postal codes (length of 6, the right % of alpha and numerical, all uppercase). However, in our latest experiments on the dataset combining three form types, it did not do as well for such particular fields. Having access to a more significant training set could allow us to have multiple models trained on different sets of fields, where each model would learn more fine-grained formatting rules for each field.

5.4.2 Features

As mentioned in sub-section 5.2.5, we were limited by the FE's API to conduct our experiments in a timely manner. Having access to additional features would have helped us gain knowledge as to what set of features can be useful.

Another limitation is the usage of the field name as a feature. When using a one-hot encoding representation of the field name, this prevents us from using the model with new forms and fields. It also is a concern since we do not have many data points per field. On the other end, one limitation with the semantic vector representation is that it relies on some consistency in the naming of the field names, which is not necessarily enforced in the labeling process today. For instance, if two field names that are very different share much semantic information, this could confuse the FSU. It also assumes that field names from different forms will have similar naming conventions.

Lastly, the features that we have engineered so far can only help us with non-empty FE predictions. It would be essential to add features that can help in identifying the confidence of the system for a null prediction. It could be the number of black pixels close to the field's location, for example.

5.4.3 Algorithm

So far, we have done little experiments in terms of choice of models and hyper-parameters tuning. The challenge resided in the data aspect, and as we achieved adequate performance, we did not prioritize this work.

5.4.4 Out-Of-Distribution Detection

Since the goal of the FSU is to quantify how the system reacts in the face of uncertainty, it would be crucial to test the approach on Out-Of-Distribution (OOD) data points. We could test different types of OOD instances, such as running FE on images that are not forms, or on forms for which we have no stored template. Additionally, it could be on exceptional field values, such as the name of a person with 30 characters long. It will be crucial to assess if the FSU predicts low confidence for such outliers. In the same idea, it would be important to track how the FSU reacts to some distributional shift in the data or other shifts we might expect to see in production. Different methods could be envisioned to improve the performance of the FSU for that, such as training the model with some OOD data instances or modeling the FSU with a Bayesian approach.

CHAPTER 6 GENERAL DISCUSSIONS

6.1 Comparison of Both Systems

In this section, we compare the two systems present in our use cases. We highlight their differences and how these impact our approach. This section underscores how our approach can be generic and applicable to different contexts.

6.1.1 Different Granularity

Because supply chains are complex, its representation is much less granular than FE in its modelization in the FSU. There are important sub-systems dynamics with are not analyzed, such as how each element has its own set of models and heuristics, with aleatoric and epistemic uncertainty too. As an example, we could have considered the production individually as a system and use the FSU to predict plant failures.

In the AI system, FE, although some of its elements are quite complex, such as the DL models, the general structure of the system is simpler. Nevertheless, some level of granularity is still abstracted, such as the training and testing procedure of each model. Understanding these steps is crucial when analyzing the system's behavior. Nevertheless, some features of the FSU can account for that, such as metrics measuring the uncertainty of each model. An example of that is the confidence estimates from OCR recognizer, which can help detect when the data at inference time is far from the training data.

6.1.2 Physical Processing Units and Time

A fundamental difference between the nature of both systems is the context in which they operate. The constraints of the physical world bind the supply chain use case - its elements are mostly tangible processing units, such as plants and DCs. Since they function in our physical world, randomness can affect their outcome. This notion is less present in the AI system, as even the output of the AI system is intangible; it essentially outputs knowledge. Yet, some physical constraints also affect the system, such as the Graphics Processing Units (GPUs) needed to execute it.

Another related difference in both use cases is the time aspect. As supply chains contain tangible elements and processes, there are some inevitable delays. As it is impossible to produce items instantaneously, planners need to perform many projections. The likelihood of system failure is thus a time series problem, where the situation in a previous timestamp can impact the next ones. As such, when modeling features for the FSU, it is essential to include information from previous timestamps. Additionally, when planners act to avoid a system failure, inherent delays exist, which require the FSU to predict failures a few weeks in advance. In FE, it is not the case. Rectifying an erroneous predicted text can be done quickly. Each prediction is also independent of one another.

6.1.3 Stock and Feedback Loops

Following the previous idea, another distinction between both systems is in the nature of the stock and feedback loops.

In supply chains, there is tangible stock at various stage to compensate for different sources of uncertainty, such as stocks of raw supply and finished goods at the plant and stock at the DCs. In AI systems, the notion of a stock is less straightforward, as it is not tangible. Since the output of the AI system is a prediction about a specific situation, we consider the stock as being the knowledge accumulated at training time by each model. For example, if a model has seen a lot of noisy data at training time, it can react better to such cases at inference time.

Feedback loops are a key concept in systems as they keep the stock at an adequate level. In supply chains, one great example is the safety stock target. Once a stock drops below a certain point, which is usually a function of the item's demand, an order is automatically triggered to replenish the stock. In a AI system, a feedback loop would be a process that monitors the amount of knowledge, so it does not drop below a certain level. This process could automatically trigger the retraining of the models. Monitoring performance is critical for AI systems in production to ensure they can be resilient to evolving and noisy environments. The FSU uncertainty estimates could be used for that purpose if they can detect a shift in the data distribution. This approach could also benefit from active learning. As the FSU identifies uncertain data points, they can be verified and labeled by a HITL. Thus, this becomes an automatic way to gather additional data and knowledge on these edge cases.

6.2 Features to Predict Failures in Systems

In both our use cases, we engineered features that intuitively seemed like good predictors of system failure. In hindsight, we found some common themes in what our features measure: stock, epistemic uncertainty, aleatoric uncertainty, and validation features. We present our learnings in this section.

6.2.1 Stock

Stock can compensate for some uncertainty. As such, it is important to include it in FSU's features. In supply chains, when we observe a drastic increase in the demand, it may not cause a system's failure if enough stock was already sitting at the relevant DC. In our model, we included the stock both at the plant and the DC. Not surprisingly, the stock at the DC was one of the most important indicators of a future failure. In AI systems, modeling the stock could correspond to measuring the epistemic uncertainty, hence the lack of knowledge of each model of the system. In our experiments, we used uncertainty estimates from OCR Recognizer.

6.2.2 Epistemic Uncertainty

When trying to identify epistemic uncertainty in a system, it boils down to question what is unknown to the system. Assuming no randomness, if the system knew about everything, then it would likely never fail. Epistemic uncertainty can be present in different places and at different levels. In our experiments, we found three typical reasons why a system can lack knowledge. First, it can be due to a lack of general knowledge or common sense, such as not knowing about basic laws of physics or things that cannot exist. Second, the lack of knowledge can be due to elements that were left out of the system's boundaries. Lastly, each element may have its uncertainty, and subparts of the system may not interact well. It could be due to noisy and incomplete datasets, to poor model configurations, or even generally to bad system design, such as defective feedback loops.

Lack of general knowledge

As humans, we tend to be good at performing various diverse tasks, even when they are unfamiliar to us. It is partially due to general knowledge that we acquired through experiments and education, which allows us to judge of a new situation and use our common sense to make an appropriate decision. Systems and machines generally do not have access to such knowledge, as they are usually designed for a specific task and only know about the information we give them. Consequently, a HITL can have a significant impact.

In our second use case, we engineered many features to account for that lack of general knowledge. In the way the system is designed, FE is not aware of what format is valid for each field. When being asked for a particular field such as *birth_day*, the system does not know that the output should be a numerical value between 1 and 31. To account for some of that, we built the *output format validation* features. Additionally, the system cannot detect inconsistency between results of different fields on the same form. For instance, if the *birth_month* is February, the *birth_day* should be below 29. In our case, we did not engineer any features for that last example.

System's boundaries

The lack of knowledge of a system can also be due to the choice of the system's boundaries. As discussed in the Introduction, systems typically do not have defined limits; they are usually created artificially to simplify the analysis of a system's behavior.

In the supply chain project, we excluded many elements of the system due to a lack of historical data. As such, we did not include forecasts, raw materials procurement, and available-to-promise quantities. If any of these elements is a frequent cause of failure, excluding it from the system makes it difficult to understand the system's behavior. The same applies to external factors, such as environmental risks or even weather, which could impact the demand, employees no-show, and transportation.

In the second use case, we also ignored some elements for data limitation reasons, such as OCR Localizer and Form Aligner. Additionally, we did not include the data generation process, i.e., the forms. This process contains different steps which can generate noisy data. For example, the data generation process could start with an empty form (although we could go further than that), which is then filled by a human. There is some randomness as we never write the same way. The filled form can then be processed by some agent, which may add noise to the form by adding scribbles on sections that are unimportant, for instance (we have seen examples of that in the data). The filled form is then either scanned or photographed, which can also cause some randomness due to the manual process. Before we can feed the data to a model, it also needs to be labeled by data annotators, which can also make mistakes. Even though we cannot eliminate all these sources of noise, additional data could help in understanding some of the data variance. For instance, it could be general metadata on the picture, such as the exposition, the camera model, or even the name of the photographer.

Poorly configured models and inadequate system design

Poorly designed systems or models, such as models with the wrong parameters, can also be a cause of system failure. In ML, this can be due to models that were trained independently on incomplete and noisy datasets, thus which parameters are not rightly optimized. It can also be due to a faulty training procedure leading to an over-fitted model. The choice of the model may even not be appropriate for the task at hand. Lastly, models may not perform adequately when in interactions with other components. Modeling this type of uncertainty boils down to the features presented in the stock paragraph above. The most obvious ones are uncertainty estimates computed for each element of the system, as we have done with OCR Recognizer. However, not all models can estimate their uncertainty. Therefore, we can also feed directly models parameters or outputs to the FSU so it can learn when not to trust the model's parameters.

Inappropriate feedback loops are one example of poor system design. When they work properly, they prevent failures without human intervention. In our project with Michelin, we detected a problematic feedback loop, which caused repetitive issues unknown to planners. For a few products, the target was set erroneously to zero. The FSU model learned that this was indicative of a failure. When supply chain planners investigated some of the examples, they realized that the process of computing safety stock targets was not working correctly, affecting lots of items.

6.2.3 Aleatoric Uncertainty

The environment in which the system operates usually have aleatoric uncertainty that can bring noise in the input of the system and failures in its operations. Some elements can also have inherent randomness.

In the supply chain example, as the model predicts system failure a few weeks ahead, it can be difficult to find features measuring aleatoric uncertainty that can be predictive of future failures. However, due to the lead time between the different supply chain subparts, upstream operations sometimes impact the service level with a delay. For instance, machine failures which impact the current production level would not influence the current customers' delivery, as the corresponding items are already produced and stored at the right DC. This delay give planners some leverage to react and provide useful features for the FSU. Another particularity with aleatoric events is that data rarely exists on them, such as data on machine failures, or employees strike. Hence, in our experiments, we used features that would be useful proxies to identify those execution problems. As an example, we measured the number of items produced as compared to the production plan, which measures the ability of the plant to execute despite random events.

In FE, the two primary sources of randomness is the form aligner step and the data generation process, as we discussed in the systems boundaries sub-section. We did not use any features to model these sources of aleatoric uncertainty as it was not a frequent cause of errors. However, features could explicitly characterize the noise in the data, or a metric could evaluate the quality of the form aligner step.

6.2.4 Validation Features

Some features can be powerful to compensate for the lack of data in some sources of uncertainty. These features are usually indicators that the system is working as expected. We already mentioned an example when discussing features that detect execution problems at the plant. Another example, in FE, is the output validation features set that verify if the predicted text has the right format. Although not explicitly, this feature accounts for the uncertainty of all previous steps, as it assesses that the overall prediction makes sense. A similar idea is present in the supply chain model, where a feature quantify if the stock is above the safety stock at each DC. This feature also captures some uncertainty of the previous steps - if raw material procurement or production would have failed, this feature could capture it.

6.3 Users Workflow

In both projects, we needed to develop a new workflow to enable users to interact with the failures prediction. Because of fundamental differences in the nature of both underlying systems, the two workflows ended up being quite different, impacting modeling decisions.

6.3.1 Explainable AI

In the first project, with supply chains, we developed an elaborated UI for users to interact with the alerts. The motivation was that it is complex for supply chain planners to understand why a failure may happen, and even more difficult to understand how to avoid it. They hence need a lot of context and explainability, to identify the right root cause and pose the right action. In the second project, the users do not need as much context as to why the predicted text may not be accurate. Explanations such as "I do not think this is a postal code because there is only four characters" does not particularly help the user in correcting the prediction. The fix is always the same; they manually override the prediction if it is erroneous. We could still imagine a workflow where the cause of the failure could automatically trigger a correction to the predicted text. As an example, if the explanation is "There is a numerical character in the name of the person", automatic recommendations could propose to "remove numerical character" or "replace by this letter", where we propose the closest match to the numerical value.

Besides, explainability was important in both use cases to identify common patterns in a system failure that, if eliminated, can help improve the system's design and performance. With supply chains, the wrongly configured safety stock target is an excellent example of that. In FE, it can bring visibility to scientists as to the most critical system's improvement opportunities. For instance, if most errors are due to formatting errors, it could trigger work to add a heuristic to the system that validates that the output makes sense for every prediction.

6.3.2 Useful Alerts

In the supply chain project, we also defined the notion of useful alerts, which corresponds to alerts that are actionable, exclusive, and significant. This notion was crucial for their context since supply chain planners do not have much time to look at the alerts generated by the system. It was crucial to bring their focus on alerts for which they could make a difference. These notions were not as crucial in FE, as discussed below. First, not all situations in supply chains are actionable - sometimes delays do not allow for any flexibility. With FE, all failures prediction are actionable, since the action is to rectify the predicted text. Secondly, the concept of exclusivity is important in supply chains as our model was conjointly working with other systems with their alerts. It was thus crucial to only show alerts which were new to planners. In FE, it was not the case. Lastly, alerts need to be of significance. This notion was important in both systems. For FE, not all fields are as important on a document. For instance, it may be critical to output accurately the name of the person or its social number, and not so much its age. In this case, alerts that concern those critical fields are of more significance, and users should deal with them in priority. A similar idea existed in the supply chain use case, where not all products and customers are as important.

6.3.3 Thresholds and Performance Metrics

In the second use case, as shown in Figure 5.2, we covered some potential workflows that thresholds based on different confidence estimates could enable, namely manual, validation and automatic mode. This capability was key in the AI system as users have time to investigate all uncertain predictions made by the system. This was not the case in supply chains due to time constraints. Indeed, supply chain planners investigated alerts ranked in order of probability of failure, going down the list of alerts as much as possible in the time they had. As such, it was pointless to define a threshold over which planners should investigate alerts. This difference in the two use cases' workflows impacted the choice of performance metrics. In the supply chains use case, as planners do not have the bandwidth to go through all uncertain situations, users care more about precision, also called Positive Predictive Value (PPV), i.e. how relevant are the alerts, then recall, i.e. if all issues are detected. In the case of the AI system, the focus is different. Because the system is in automatic mode above a certain threshold, it is important that when the FSU predicts there is no issue, that this is effectively the case. If we consider an issue as the positive class, this would be equivalent to the Negative Predictive Value (NPV).

CHAPTER 7 CONCLUSION AND RECOMMENDATIONS

7.1 Summary of Works

In this thesis, we highlight the importance of predicting system failure through two concrete use cases; supply chains and AI systems. Through our experiments, we underscore the importance of modeling a system and its sources of uncertainty holistically to allow us to understand systems' behavior. We use a ML model to understand correlations between systems failures and the state of the underlying system. We called our approach the Full-Stack Uncertainty Model (FSU).

In the first use case, we predict system failure in the context of Michelin's supply chain, an international tires manufacturer. We showcase that our approach yields good performance on different datasets, and we discuss the importance of providing context and explainability for users to build trust and to allow them to prevent failures. As such, we designed and developed a workflow, enabled through a UI, that allows supply chain planners to get a summary of all alerts at a glance, to navigate through them and filter not useful ones, and to visualize all the relevant data to carry out the right corrective action. In that context, we defined useful alerts as being actionable, meaning that levers still exist to prevent the failure, exclusive, i.e., not already known to planners, and significant, meaning that it affects items and locations worth investigating. By drawing their attention to uncertain cases, our approach enabled planners to avoid multiple failures, thus leveraging a HITL approach to boost the supply chains' performance. Nonetheless, our most important contribution is the identification of recurrent issues in the subparts of the supply chain. Amongst other problems, we identified an issue in the safety stock algorithm, which caused the system to fail on numerous occasions. Lastly, our alerting system also provided value by breaking silos between different parts of the supply chain, by providing a holistic view of the situation, instead of a narrow view as dictated by the different planners' ownership.

In the second use case, we present similar experiments on a AI system, FE, designed to extract fields information from a scanned form. The system is composed of different models, including DL algorithms and heuristics. We showcase promising preliminary results on providing confidence estimates for FE predictions using our FSU approach. We also discuss how these confidence values can trigger the intervention of a HITL to boost the system's performance, as in the first use case.

In the discussion, we underscore how our approach to system failure prediction is generic and flexible. It can work with systems studied at different granularity, accommodate different failure definitions, and use a wide variety of features characterizing different types and sources of uncertainty. The failure predictions can also adapt to different needs, such as identifying HITL opportunities to improve the underlying system's predictions and uncovering recurrent issues in the system. As such, the model needs to be explainable, so that failure root causes can be identified and tackled.

7.2 Limitations

In both projects, we conducted our research on systems used in production. Yet, the FSU experiments were done as proofs-of-concept, manually outside the productized environment. As such, they are different considerations that we did not explore when we put such a model in production.

First, the FSU performance is tightly tied to the underlying system and its performance. We did not investigate how the FSU would react in the face of a drift in the data distribution or the system's performance. In an ideal world, we would like the FSU to detect such situations by providing lower confidence estimates over time. Additionally, the performance drift in the underlying system can be due to the very own usage of the FSU approach; as the model is used to predict and prevent failures, fewer errors occur, making the FSU less and less valuable. It is unclear if the method can adapt to discover more fine-grained source of failures.

Secondly, the features engineering task for the FSU is manual and requires some intuition on what can cause a system to fail or a detailed error analysis of the system's performance under different conditions. This manual step can be hard to scale to production environments.

Lastly, related to the two previous ideas, as features engineering is dependent of the underlying system, it can bring challenges and overhead on the whole's system maintenance. As elements in the systems are improved and potentially replaced, it affects the FSU performance and can influence the set of available features. For instance, if we update the OCR Recognizer model with a more performant algorithm, but that is unable to produce uncertainty estimates, then the FSU may lose one of its most essential features. We already experienced some of these challenges while working in our second use case, as two teams were working simultaneously on the FSU and on improving the underlying model, FE.

7.3 Future Research

In terms of future work, tackling some of the limitations mentioned above would be key. For instance, it would be necessary to simulate how the FSU model reacts to a shift in the data distribution and to outliers. It would also be crucial to investigate if the features engineering step can be automated or at least formalized so that when we model new systems, it can be straightforward and quick to implement our approach.

Another opportunity lies in the definition of a system failure. In both of our approaches, we represented them as a binary variable, where the system is failing or not. However, in practice, not all failures carry the same impact. In supply chains, if failures are for big orders or major customers, it is primordial for the FSU to detect such cases. Today, we accounted for that in verifying our performance on the subset of useful alerts, but it would be more robust to incorporate that as part of the optimization of the model. With FE, we could explore a similar idea by modeling the edit distance of the prediction instead of the exact match. It could involve changing the FSU approach to a regression model, or multi-class classification, instead of a binary one.

It would also be compelling to test our approach on non-sequential systems, such as systems with loops and branches, where the input to the system can go to different steps or multiple times over the same ones.

Finally, as more data is collected, it would be appealing to test if we can model the FSU with more sophisticated techniques that can account for the uncertainty of the system itself or its dynamic nature. For instance, we could use a Bayesian approach where we simultaneously learn the epistemic uncertainty of the model and discount the confidence estimates accordingly. We could also use reinforcement learning, where the actions of the HITL impact its universe.

REFERENCES

[1] D. H. Meadows, *Thinking in systems: A primer.* chelsea green publishing, 2008.

[2] S. M. Lundberg, G. G. Erion, and S.-I. Lee, “Consistent individualized feature attribution for tree ensembles,” *arXiv preprint arXiv:1802.03888*, 2018.

[3] N. N. Taleb, *The black swan: The impact of the highly improbable.* Random house, 2007, vol. 2.

[4] D. J. Spiegelhalter and H. Riesch, “Don’t know, can’t know: embracing deeper uncertainties when analysing risks,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 369, no. 1956, pp. 4730–4750, 2011.

[5] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[6] S. Rao and T. J. Goldsby, “Supply chain risks: a review and typology,” *The International Journal of Logistics Management*, vol. 20, no. 1, pp. 97–123, 2009.

[7] M. Batty and P. M. Torrens, “Modelling and prediction in a complex world,” *Futures*, vol. 37, no. 7, pp. 745–766, 2005.

[8] I. Rahwan *et al.*, “Machine behaviour,” *Nature*, vol. 568, no. 7753, p. 477, 2019.

[9] Z. C. Lipton, “The mythos of model interpretability,” *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[10] J. D. Lee and K. A. See, “Trust in automation: Designing for appropriate reliance,” *Human factors*, vol. 46, no. 1, pp. 50–80, 2004.

[11] Y. Daultani *et al.*, “A supply chain network equilibrium model for operational and opportunism risk mitigation,” *International Journal of Production Research*, vol. 53, no. 18, pp. 5685–5715, 2015.

[12] S. K. Kumar, M. Tiwari, and R. F. Babiceanu, “Minimisation of supply chain cost with embedded risk using computational intelligence approaches,” *International Journal of Production Research*, vol. 48, no. 13, pp. 3717–3739, 2010.

[13] H. Stadtler and C. Kilger, *Supply chain management and advanced planning.* Springer, 2002, vol. 4.

[14] Z. Khojasteh-Ghamari and T. Irohara, "Supply chain risk management: A comprehensive review," in *Supply Chain Risk Management.* Springer, 2018, pp. 3–22.

[15] C. S. Tang, "Perspectives in supply chain risk management," *International journal of production economics*, vol. 103, no. 2, pp. 451–488, 2006.

[16] B. Ritchie and C. Brindley, "Supply chain risk management and performance: a guiding framework for future development," *International Journal of Operations & Production Management*, vol. 27, no. 3, pp. 303–322, 2007.

[17] I. Heckmann, T. Comes, and S. Nickel, "A critical review on supply chain risk–definition, measure and modeling," *Omega*, vol. 52, pp. 119–132, 2015.

[18] Y. Khojasteh, *Supply chain risk management.* Springer, 2017.

[19] J. Chen, A. S. Sohal, and D. I. Prajogo, "Supply chain operational risk mitigation: a collaborative approach," *International Journal of Production Research*, vol. 51, no. 7, pp. 2186–2199, 2013.

[20] U. Jüttner, H. Peck, and M. Christopher, "Supply chain risk management: outlining an agenda for future research," *International Journal of Logistics: Research and Applications*, vol. 6, no. 4, pp. 197–210, 2003.

[21] P. Singhal, G. Agarwal, and M. L. Mittal, "Supply chain risk management: review, classification and future research directions," *International Journal of Business Science & Applied Management (IJBSAM)*, vol. 6, no. 3, pp. 15–42, 2011.

[22] B. Fahimnia *et al.*, "Quantitative models for managing supply chain risks: A review," *European Journal of Operational Research*, vol. 247, no. 1, pp. 1–15, 2015.

[23] M. S. Pishvaee, M. Rabbani, and S. A. Torabi, "A robust optimization approach to closed-loop supply chain network design under uncertainty," *Applied Mathematical Modelling*, vol. 35, no. 2, pp. 637–649, 2011.

[24] M. D. Garvey, S. Carnovale, and S. Yeniyurt, "An analytical framework for supply network risk propagation: A bayesian network approach," *European Journal of Operational Research*, vol. 243, no. 2, pp. 618–627, 2015.

[25] R. R. Pai *et al.*, "Methods toward supply chain risk analysis," in *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, vol. 5. IEEE, 2003, pp. 4560–4565.

[26] E. Begoli, T. Bhattacharya, and D. Kusnezov, "The need for uncertainty quantification in machine-assisted medical decision making," *Nature Machine Intelligence*, vol. 1, no. 1, p. 20, 2019.

[27] H. Jiang *et al.*, "To trust or not to trust a classifier," in *Advances in neural information processing systems*, 2018, pp. 5541–5552.

[28] X. Jiang *et al.*, "Calibrating predictive model estimates to support personalized medicine," *Journal of the American Medical Informatics Association*, vol. 19, no. 2, pp. 263–274, 2011.

[29] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv preprint arXiv:1610.02136*, 2016.

[30] C. Guo *et al.*, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1321–1330.

[31] V. Kuleshov and P. S. Liang, "Calibrated structured prediction," in *Advances in Neural Information Processing Systems*, 2015, pp. 3474–3482.

[32] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[33] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 427–436.

[34] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[35] A. Malinin and M. Gales, "Predictive uncertainty estimation via prior networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 7047–7058.

[36] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

[37] A. Mobiny *et al.*, “Dropconnect is effective in modeling uncertainty of bayesian deep networks,” *arXiv preprint arXiv:1906.04569*, 2019.

[38] N. Papernot and P. McDaniel, “Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning,” *arXiv preprint arXiv:1803.04765*, 2018.

[39] T. DeVries and G. W. Taylor, “Learning confidence for out-of-distribution detection in neural networks,” *arXiv preprint arXiv:1802.04865*, 2018.

[40] J. Platt *et al.*, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999.

[41] B. Zadrozny and C. Elkan, “Transforming classifier scores into accurate multiclass probability estimates,” in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2002, pp. 694–699.

[42] A. Niculescu-Mizil and R. Caruana, “Predicting good probabilities with supervised learning,” in *Proceedings of the 22nd international conference on Machine learning.* ACM, 2005, pp. 625–632.

[43] P. L. Bartlett and M. H. Wegkamp, “Classification with a reject option using a hinge loss,” *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1823–1840, 2008.

[44] M. Yuan and M. Wegkamp, “Classification methods with reject option based on convex risk minimization,” *Journal of Machine Learning Research*, vol. 11, no. Jan, pp. 111–130, 2010.

[45] C. Cortes, G. DeSalvo, and M. Mohri, “Learning with rejection,” in *International Conference on Algorithmic Learning Theory.* Springer, 2016, pp. 67–82.

[46] A. Shafaei, M. Schmidt, and J. J. Little, “Does your model know the digit 6 is not a cat? a less biased evaluation of" outlier" detectors,” *arXiv preprint arXiv:1809.04729*, 2018.

[47] D. Amodei *et al.*, “Concrete problems in ai safety,” *arXiv preprint arXiv:1606.06565*, 2016.

[48] Y. Ovadia *et al.*, “Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift,” *arXiv preprint arXiv:1906.02530*, 2019.

[49] T. Nguyen *et al.*, “Big data analytics in supply chain management: A state-of-the-art literature review,” *Computers & Operations Research*, vol. 98, pp. 254–264, 2018.

[50] T. Papadopoulos *et al.*, “The role of big data in explaining disaster resilience in supply chains for sustainability,” *Journal of Cleaner Production*, vol. 142, pp. 1108–1118, 2017.

[51] J. B. S. Ong *et al.*, “Understanding natural disasters as risks in supply chain management through web data analysis,” *International Journal of Computer and Communication Engineering*, vol. 4, no. 2, p. 126, 2015.

[52] A. J. Schmitt, “Strategies for customer service level protection under multi-echelon supply chain disruption risk,” *Transportation Research Part B: Methodological*, vol. 45, no. 8, pp. 1266–1283, 2011.

[53] S. K. Paul, R. Sarker, and D. Essam, “A quantitative model for disruption mitigation in a supply chain,” *European Journal of Operational Research*, vol. 257, no. 3, pp. 881–895, 2017.

[54] D. Simchi-Levi *et al.*, “Identifying risks and mitigating disruptions in the automotive supply chain,” *Interfaces*, vol. 45, no. 5, pp. 375–390, 2015.

[55] J. H. Friedman, “Greedy function approximation: a gradient boosting machine,” *Annals of statistics*, pp. 1189–1232, 2001.

[56] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* ACM, 2016, pp. 785–794.

[57] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems*, 2017, pp. 4765–4774.

[58] L. S. Shapley, “A value for n-person games,” *Contributions to the Theory of Games*, vol. 2, no. 28, pp. 307–317, 1953.

[59] E. Grave *et al.*, “Learning word vectors for 157 languages,” in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

APPENDIX A FEATURES SET OF THE FULL-STACK UNCERTAINTY MODEL

In Figure A.1, we show the set of features that we used in our model, for which we show results in Section 5. The first column displays the feature name and the second column details its source in the available data. The third column counts the number of columns that each feature account for in the training data. Lastly, we present a short description of what each feature represents.

Table A.1 Features set used in the Full-Stack Uncertainty Model model

Feature Types	Source	Nb	Description
(1) OCR Recognizer Confidence Estimates	OCR Recognizer	1	Softmax probabilities of OCR Recognizer
(2) Output Length	Predicted Text	1	Nb of characters in the prediction
(3) % Alphabetic	Predicted Text	1	Percentage of alphabetic characters in the prediction
(4) % Numerical	Predicted Text	1	Percentage of numerical characters in the prediction
(5) % Other (not alphanumeric)	Predicted Text	1	Percentage of characters which are not alphanumerical
(6) % Upper Case	Predicted Text	1	Percentage of alphabetic characters which are in upper case
(7) Numerical Output	Predicted Text	1	When the prediction is numerical, actual numerical value
(8) First letter	Predicted Text	89	First character of the prediction (one-hot encoding)
(9) Is title	Predicted Text	1	Binary feature if the prediction starts with an uppercase
(10) Field Name Embedding	Field Name	300	Pre-trained subword embedding representation of the field name