Vladimir Balayan

Bachelor of Science

Human-Interpretable Explanations for Black-Box Machine Learning Models: An Application to Fraud Detection

Dissertation submitted in partial fulfillment
of the requirements for the degree of

Master of Science in
Analysis and Engineering of Big Data

Adviser: Pedro Saleiro, Senior Research Manager, Feedzai
Co-adviser: Ludwig Krippahl, Assistant Professor, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Examination Committee

Chair: Prof. Doutor Pedro Manuel Corrêa Calvente de Barahona, Full Professor, FCT-NOVA
Rapporteur: Prof. Doutor Bruno Emanuel da Graça Martins, Assistant Professor, IST

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

January, 2020

Acknowledgements

I would like to first pay my special regards to my thesis adviser Pedro Saleiro for his consistent support and guidance during the running of this work. His patience and wisdom were essential in the elaboration of this thesis and all the knowledge acquired. To my co-advisor Ludwig Krippahl for all the availability, valuable feedback and words of encouragement.

To my work colleagues for the excellent daily meetings that charged me with energy for the rest of the day. I would particularly like to single out Catarina Belém for all constructive feedback and the availability to help at any time.

To Feedzai, particularly Pedro Bizarro, for all the support given to this project. Also the experts who were involved in this research work who showed the availability and excellent disposition to carry out the experiments.

To my family for all the emotional support and great motivation. Essentially to my parents and little brother, Oleg, Lu, and David for all the love and continuous encouragement. You are always there for me.

To my friends for the motivation and all happy distractions in moments of rest, and to my girlfriend, Patrícia, for all the patience and support during the elaboration of this work.

I would like to recognize the invaluable assistance that you all provided during this work. Thank you.

"Never Explain Anything",
H.P. Lovecraft

Abstract

Machine Learning (ML) has been increasingly used to aid humans making high-stakes decisions in a wide range of areas, from public policy to criminal justice, education, healthcare, or financial services. However, it is very hard for humans to grasp the rationale behind every ML model's prediction, hindering trust in the system. The field of Explainable Artificial Intelligence (XAI) emerged to tackle this problem, aiming to research and develop methods to make those "black-boxes" more interpretable, but there is still no major breakthrough. Additionally, the most popular explanation methods — LIME and SHAP — produce very low-level feature attribution explanations, being of limited usefulness to personas without any ML knowledge.

This work was developed at Feedzai, a fintech company that uses ML to prevent financial crime. One of the main Feedzai products is a case management application used by fraud analysts to review suspicious financial transactions flagged by the ML models. Fraud analysts are domain experts trained to look for suspicious evidence in transactions but they do not have ML knowledge, and consequently, current XAI methods do not suit their information needs. To address this, we present JOEL, a neural network-based framework to jointly learn a decision-making task and associated domain knowledge explanations. JOEL is tailored to human-in-the-loop domain experts that lack deep technical ML knowledge, providing high-level insights about the model's predictions that very much resemble the experts' own reasoning. Moreover, by collecting the domain feedback from a pool of certified experts (human teaching), we promote seamless and better quality explanations. Lastly, we resort to semantic mappings between legacy expert systems and domain taxonomies to automatically annotate a bootstrap training set, overcoming the absence of concept-based human annotations. We validate JOEL empirically on a real-world fraud detection dataset, at Feedzai. We show that JOEL can generalize the explanations from the bootstrap dataset. Furthermore, obtained results indicate that human teaching is able to further improve the explanations prediction quality.

Keywords: Machine Learning, Explainable AI, Domain Knowledge Explanations, Self-Explainable Methods, Human-AI Cooperative systems

Resumo

A Aprendizagem de Máquina (AM) tem sido cada vez mais utilizada para ajudar os humanos a tomar decisões de alto risco numa vasta gama de áreas, desde política até à justiça criminal, educação, saúde e serviços financeiros. Porém, é muito difícil para os humanos perceber a razão da decisão do modelo de AM, prejudicando assim a confiança no sistema. O campo da Inteligência Artificial Explicável (IAE) surgiu para enfrentar este problema, visando desenvolver métodos para tornar as "caixas-pretas" mais interpretáveis, embora ainda sem grande avanço. Além disso, os métodos de explicação mais populares — *LIME* and *SHAP* — produzem explicações de muito baixo nível, sendo de utilidade limitada para pessoas sem conhecimento de AM.

Este trabalho foi desenvolvido na Feedzai, a *fintech* que usa a AM para prevenir crimes financeiros. Um dos produtos da Feedzai é uma aplicação de gestão de casos, usada por analistas de fraude. Estes são especialistas no domínio treinados para procurar evidências suspeitas em transações financeiras, contudo não tendo o conhecimento em AM, os métodos de IAE atuais não satisfazem as suas necessidades de informação. Para resolver isso, apresentamos JOEL, a *framework* baseada em rede neuronal para aprender conjuntamente a tarefa de tomada de decisão e as explicações associadas. A JOEL é orientada a especialistas de domínio que não têm conhecimento técnico profundo de AM, fornecendo informações de alto nível sobre as previsões do modelo, que muito se assemelham ao raciocínio dos próprios especialistas. Ademais, ao recolher o *feedback* de especialistas certificados (ensino humano), promovemos explicações contínuas e de melhor qualidade. Por último, recorremos a mapeamentos semânticos entre sistemas legados e taxonomias de domínio para anotar automaticamente um conjunto de dados, superando a ausência de anotações humanas baseadas em conceitos. Validamos a JOEL empiricamente em um conjunto de dados de detecção de fraude do mundo real, na Feedzai. Mostramos que a JOEL pode generalizar as explicações aprendidas no conjunto de dados inicial e que o ensino humano é capaz de melhorar a qualidade da previsão das explicações.

Palavras-chave: Aprendizagem de Máquina, IA Explicável, Explicações de conhecimento de domínio, Métodos Auto-Explicáveis, Sistemas Humano-IA Cooperativos

Contents

List of Figures

List of Tables

Acronyms

AI	Artificial Intelligence
AM	Aprendizagem de Máquina
AUC	Area Under the Curve
BCE	Binary Cross Entropy
BPMLL	Back-Propagation Multi-label Learning
CNN	Convolutional Neural Networks
DNN	Deep Neural Networks
FFNN	Feed Foward Neural Network
FPR	False Positive Rate
IAE	Inteligência Artificial Explicável
JOEL	Jointly learned cOncept-based ExpLanations
ML	Machine Learning
NN	Neural Network
TPR	True Positive Rate
UI	User Interface
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Motivation

The use of ML to inform high-stakes decision-making is becoming standard in many different sensitive domains, such as healthcare [17], financial services [23], or criminal justice [15]. At the same time, the continuously growing availability of data, paired with the commoditization of cloud computing, enabled training larger, highly complex ML models. Hence, state-of-the-art ML models are perceived as "black-boxes" due to their opaqueness to human stakeholders. This paradigm contributes to a mistrust in Artificial Intelligence (AI) and reduces the human capacity of detecting both technical and ethical risks in these systems (*e.g.*, [75, 28, 4]).

As a consequence, the field of XAI is becoming very popular in the ML research community. It stands for techniques that try to improve the interpretability of complex models to mitigate the issues stated above [36]. However, there is still a lack of consensus on definitions, evaluations, and taxonomies [53]. Moreover, most proposed methods are tailored for data scientists [75, 55, 67, 76, 98], producing low-level explanations (*e.g.* feature-attribution explainers [1]) that are very difficult to grasp to the humans-in-the-loop [28, 46, 58]. Usually, these humans are domain experts, responsible of making decisions based on the model's predictions, but do not have any ML skills.

1.2 Problem Statement

This work was developed at Feedzai, a fintech company that uses ML to prevent financial crime. Feedzai's clients, such as banks or payment processors, have their own risk

[1] These build explanations as lists of input features and associated feature scores, *i.e.*, feature's contribution to a given ML prediction

management teams (mostly fraud analysts) that are responsible for declining or accepting suspicious transactions. The ML models score every transaction, and the system forwards all transactions flagged as high risk to the case management User Interface (UI), to be reviewed by the fraud analysts. When declining or accepting suspicious transactions, fraud analysts need to provide a justification. Figure 1.1 describes the general flow, starting with the transaction that arrives at the Feedzai platform.

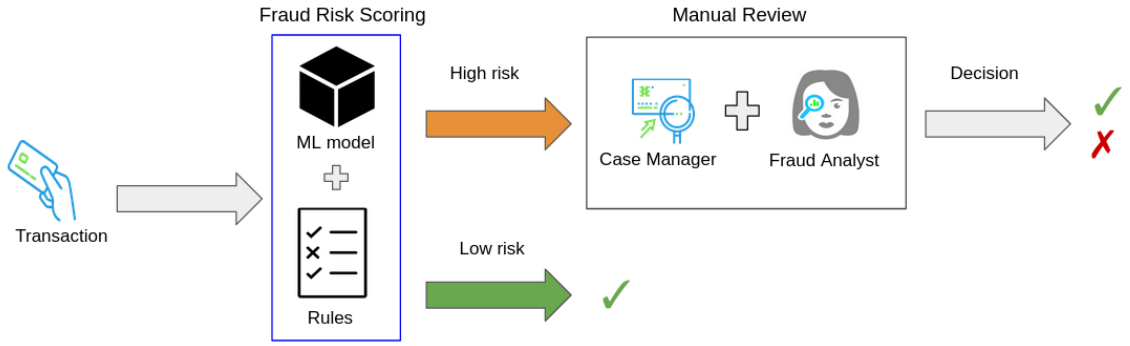


Figure 1.1: Case management flow of payment transactions through Feedzai's platform.

Besides the transaction fraud score, the analysts have access to contextual information, such as previous transactions for each linked entity (*e.g.*, credit card), and often they have limited time to make a decision. In practice, the model score alone is not sufficient to aid the analyst in her review nor to debug the model itself, hindering trust, analysts' efficiency and, ultimately, the system robustness. Although being domain experts, fraud analysts do not have data science skills, and current state-of-the-art in XAI does not suit their information needs. They would have to understand the explanation very quickly, and feature-importance or case-based explanations (*i.e.*, similar transactions with similar scores) are hard to digest in a short period of time.

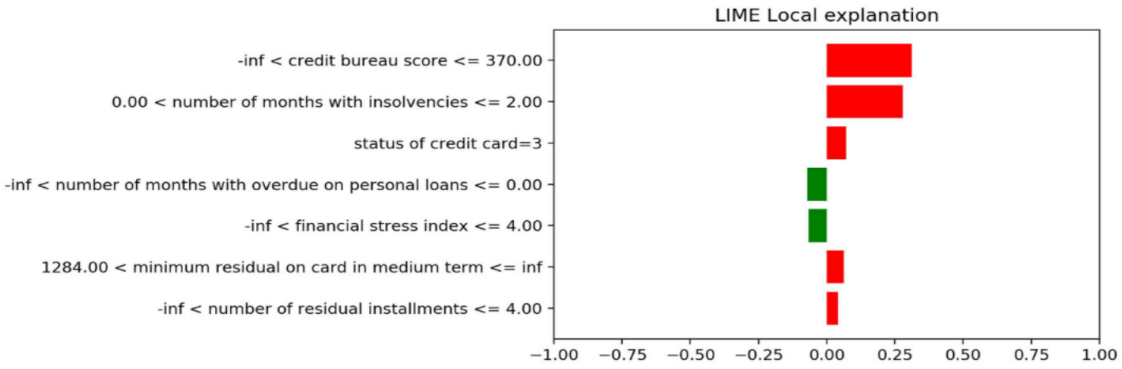


Figure 1.2: Example of LIME explanations in a Credit Risk problem [91].

Figure 1.2 shows the output of the current state-of-the-art explanation method LIME

[75]. The explanations comprise feature names along with its value and contribution to the final prediction. For Data Scientists, this may make sense since they are the ones who build the model and understand how the features are being generated. For Fraud Analysts, it is too low-level, technical, and may confuse them.

1.3 Objectives

The main goal of this work is to develop a self-explainable fraud detection ML model that, despite being opaque, provides high-level explanations using fraud domain knowledge. We aim to leverage a fraud taxonomy to create explanations using fraud concepts, and therefore, easily understandable by fraud analysts. The concepts represent a more abstract and generic idea of a given phenomena, unlike the ML model features that are usually used as explanations. An example of an ideal explanation in this setting would be the following:

"There are **multiple shipping addresses** associated with this credit card in the last hour, so it seems a **reshipping** pattern. It must be **fraud**!"

We also want to leverage the human-in-the-loop feedback to continuously improving the predictive performance and also the explainability of the self-explainable NN. Ultimately, we aim to create a Human-AI cooperative learning framework in which the AI helps the human expert to improve her efficiency through concept-based explanations, while the expert helps the AI to get better at predicting and explaining itself through her decisions and justifications.

1.4 Contributions and Foundations

In this dissertation we developed a XAI framework, dubbed *Jointly-learned cOncept-based ExpLanations* (JOEL), a self-explainable NN that jointly learns semantic concepts and a decision associated with a predictive task. We opt for encoding this ML interpretability in the architecture of a NN model. This gives us the flexibility to incorporate domain knowledge and associate semantic concepts to the decision-making task [46, 28, 58, 64]. Moreover, these high-level concepts may encode external information that, while not present in the data itself, may be very useful.

Our framework is particularly tailored for humans-in-the-loop that are extremely skilful in a specific domain but lack deep technical ML knowledge. The human-AI close proximity facilitates human teaching, where the model learns how to improve itself based on the teacher's (expert's) feedback.

Finally, the creation of explanation methods based on domain knowledge poses difficulties, especially, in the data collection [58]. This reflects in the low availability of

datasets. To address the impracticalities of human-labeling[2], we use a semi-supervised approach based on mappings between existing legacy rules system and domain taxonomies.

Thus, the main contributions of this work are the following:

- We create a self-explainable neural network framework called JOEL to jointly learn decision (class) labels and associated domain concepts.
- We propose a human teaching process that uses the human experts in the loop to teach the explanation model how to improve explanations.
- We use a semi-supervised approach leveraging legacy expert rule systems to bootstrap the training of the self-explainable NN.
- We create a real-world Fraud Taxonomy of concepts, together with fraud experts, that categorizes suspicious behaviors associated with fraud detection in payment transactions.
- We create a real-world manually concept-annotated dataset consisting of approximately 1500 data points. The annotations include both fraud label and semantic fraud domain concepts provided by domain experts.

Part of the material of this thesis was accepted at the **NeurIPS 2020 workshop HAMLETS** (Human And Machine in-the-Loop Evaluation and Learning Strategies)[3] in the following paper:

- V. Balayan, P.Saleiro, C. Belém, L. Krippahl, P.Bizarro, "Teaching the Machine to Explain Itself using Domain Knowledge"(2020).

1.5 Outline

The structure of this work is the following. In Chapter, 2 we review the literature, showing the main concepts and definitions of XAI. We detail the explanation requirements for different personas, and provide insights about the explanation methods that produce different types of explanations, outlining their problems. In the same chapter, we review the multi-label learning methods to serve as the inspiration for the proposed explanation method. Finally, we summarize the main takeaways of the literature review.

In Chapter 3, we describe in detail our proposed solution, reviewing the rational of the decisions that were taken. In Chapter 4, we detail the experimental setup of performed experiments in real-world fraud detection setting. In Chapter 5 we present, analyse, and discuss the obtained results. Finally, in Chapter 6, we summarize our work, drawing the main conclusions and outlining the future work.

[2]Manually creating these datasets carries abnormal costs and time efforts.
[3]HAMLETS official page: https://hamlets-workshop.github.io/

Chapter 2

State of the Art

This chapter consists in two distinct parts. In the first part, we present the current state-of-the art in the XAI field. The second part elaborates on Multi-Label methods used in the literature.

2.1 Explainable AI

In this section we explore the literature about XAI. We begin by defining interpretability and outlining the different requirements for good explanations. Based on these requirements, we describe the different taxonomies. We finalize by surveying the current state-of-the-art explanation methods and techniques, discussing their limitations.

2.1.1 Definitions, Requirements and Taxonomies

When considering the literature in XAI field, the terms that are usually confused or used ambiguously in literature are: *interpretability*, *explainability*, and *transparency* [53, 7]. Also, it is common to see the term *black-box* that refers to a complex ML model which predictions and the internal structure are not understandable by humans. On the other hand, a *White-Box*, *Glass-Box* or *Transparent model* is usually referring to a model that outcome is comprehensible by the individual [7].

Most authors consider that *interpretability* is the ability to explain or present the outcome of the ML Model in understandable terms to a human [29, 25, 75, 76, 49]. It is a common claim that transparent models, such as Linear models, decision trees, or rule-based methods are always interpretable (*e.g.* [79]). For instance, having a very deep tree, the overall decision process is not easy to grasp, meaning that although it is a transparent model, it still not interpretable [46]. Also, interpretability was defined as the ability of describing the internals of the ML model to a person in a suitable way [29].

Similarly, *explainability* is related to the explanations and the processes to produce them in understandable way to humans, with the intent to clarify or detail the reasoning behind the decision [7, 29]. Moreover, it can be considered as the interface between the ML model and the end-user [34]. That means that the explainability is "closely related to the concept of interpretability: systems are interpretable if their operations can be understood by a human, either through introspection or through a produced explanation" [13].

In this work, we will assume that interpretability and explainability refer to the same underlying concept - explaining the ML model decisions. Also, we argue that interpretability must be always considered regarding the end-user. We will discuss different end-users requirements in section 2.1.1.1.

2.1.1.1 Explanation Requirements

The choice of a "good" explanation and its quality evaluation must be user-centered [48]. Some authors [60] categorize the end-users into three general groups, defined as *ML experts*, who design ML algorithms, *data experts*, who use ML for analysis or decision making but are not considered ML experts, and *AI novices* who use AI without technical knowledge of ML. Then, the authors identify which kind of explanation is needed for each group.

In other work [86], the authors go deeper in the categorization of the end-users, defining several roles for different users according to their interaction with the ML system, claiming that a given user can take different roles. Table 2.1 summarizes the proposal of classification of the agents that interacts with ML system.

Table 2.1: Different roles, description and the goals of interpretability in ML ecosystem proposed in [86]

Role	**Role Description**	**Interpretability Goal**
Creators	create the ML system	improve the system understanding
Operators	interact directly with the ML system and report results to Creators	provide useful and trustworthy information
Executors	make decisios	make better decisions
Decision-subjects	those who are affected by the decisions made	understand why a certain decision was made
Data-subjects	training-data providers	know how their data is used
Examiners	auditors	audit ease

From this categorization, it is possible to identify different interpretability needs for each role and end-goal. Recent work [77, 61] summarizes the different explanation requirements that aim to satisfy such concerns in four pillars:

- **Fidelity**: how well the explanation approximate the prediction of black-box box [75].
- **Stability**: how similar the explanations for identical instances are, when fixing the ML model [77].
- **Comprehensibility or Human-Interpretability**: how understandable the explanation is to the humans [43].
- **Diversity**: how diverse the explanations are, i.e., whether its components are not redundant (e.g. if the explanation present three features which are correlated or mean the same, it's not a diverse explanation) [44]
- **Usefulness**: how much the explanation helps to optimize the goal of the end-user (e.g. Data Scientists want the explanations to help them to debug better the models).

We decided to add the **Usefulness** to the list of requirements because the explanation may be stable, diverse, and human-interpretable but still does not optimize the end-user goals.

Therefore, due to a large number of explanation requirements, the evaluation and assessment of the quality of the explanations is not a trivial task. Thus, the definition of the target persona and its requirements must be considered as the one of the first and most important steps to develop a explanation method.

In a credit card fraud detection setting, there are at least three different personas: data scientist, executors (or fraud analysts), and decision subjects. In the case of an executor, one of the main requirements is Human-Interpretability because explanations must be easy to understand by non-technical personas. Moreover, Usefulness must also be considered, since the explanations aim to optimize the overall performance of the executor and useless explanations are less likely to do so.

2.1.1.2 A Taxonomy of Explanations

Research works on XAI propose different categorization of the explanations and the techniques that are used to produce them. In this section, we summarize and describe the taxonomy proposed by different authors and also to discuss different criteria for taxonmies that are relevant to our work.

Interpretability Building Criteria According to author [45], interpretability can be built in different stages: before, during, or after the ML model. Table 2.2 summarizes the proposed classification.

Pre-Model methods can show insightful information even before training any model (e.g. for a given sample, show the features that are more correlated with a label of interest). **In-Model** methods are frequently called intrinsically interpretable or transparent, i.e., the domain knowledge constraints are applied to those methods with the goal to reach

Table 2.2: Interpretable methods classification according to [45].

Stage	**Methods problems examples**	**Method example**
Pre-Model or Model-free (before)	data issues (e.g. skewness)	Exploratory data analysis [51] techniques
In-Model (during)	the complexity of tree/rules	Linear Models, Decision Trees, Rule-based Learners [7]
Post-Model (after)	faithfulness to original model/perturbation problems	LIME [75], Integrated Gradients [85]

interpretability [53, 79, 61]. However, in the recent work, the authors argue that black-boxes also fit in this category, defining them as "self-explainable" (or "self-explaining") methods, i.e., a method that despite being a black-box, it is able to provide the explanations for its decisions, *i.e.*, it incorporates the interpretability architecturally (imposed through regularization [58] or by jointly learn the explanations [40, 71]) [26]. Finally, the **Post-Model** methods or **Post-hoc** explainable methods are applied to an already trained ML model. These can be further separated in **Model-Specific** or **Model-Agnostic** methods. The former is specific to certain ML model types. The Model-Specific methods limit the user to use only a certain class of ML models. On the other hand, Model-agnostic methods treat any model as black-box, ignoring its internal structure. This is a key advantage of such methods. However, the explanations produced by those methods could give a misleading perception of the ML model by providing inaccurate explanations or insufficient information to understand the decision [61, 79, 7].

It is noticeable that most of the recent work in XAI focus mostly on Post-hoc methods (*e.g.* [75, 55, 67, 76, 98]). However, as it will be analysed in section 2.1.2, these methods suffer from a few problems, including inconsistency on the explanation generation process [3, 50, 83]. In this work, our focus will be In-Model interpretability, more precisely on self-explainable methods, since there is more control on how we can introduce the interpretability. Also, we believe that by jointly learning the decision task and associated explanations leads to more robust, authentic (or faithful, meaning that the explanation is the "true" reason for the prediction [58]) [1] explanations, and more stable [58, 26].

Interpretability Scope Another criterion that can be used to classify different explanation methods is by its scope. The explanation methods with **Global** scope try to explain the aggregated knowledge of the model, by focusing at the global patterns learned from training data. On the other hand, to understand the prediction for some specific instance, the **Local** behavior of the ML model is explored, trying to find the reason behind the decision [61].

[1] In this context, we refer to authenticity as being a true reason for a given ML outcome.

Our focus is on local explanations, since the Fraud Analysts, which are our target personas, will analyse several events individually and decide if they are fraudulent or legitimate, meaning that the explanations must justify the model's decision for the exact instance they are reviewing.

2.1.1.3 Data Type

Explanation algorithms use different techniques to generate explanations. Some of them are designed to work with specific data types (*e.g.* image data, or tabular). Table 2.3 summarizes the ideas.

Table 2.3: Classification of the explanation methods by the type of the data used.

Category	Data type	Example
Data agnostic	Any	[75, 55, 81, 67, 98]
Data specific	Image	[28, 100]
	Text	[6, 68, 37]
	Tabular	[17]

Eexplanation methods that work for any type of data are considered **Data agnostic**. On the other hand, methods that use techniques that require specific data types are called **Data Specific**. For instance, methods that visualize features learned by hidden units in Convolutional Neural Networks (CNN) are tailored for images.

This work is developed on fraud detection domain, where tabular data is predominant. No other data type will be used.

Explanation Output Recent work in XAI proposes several taxonomies for the explanation methods' outputs [8, 7, 61]. Since they are not standardized, we propose alternative taxonomy that takes the main ideas from other works, summarizing this categorization in table 2.4.

Even though Molnar [61] proposes to further separate feature-based category in smaller ones (e.g., feature contribution, feature visualization, etc...), we decided to group them in a single category. In general, **Feature contribution** explanations presents a feature and its contribution to the outcome of the model. The difference between Feature contribution and **Feature visualization** is the presentation of the explanation, *i.e.* although methods such as Partial-Dependency Plot use the feature as explanations, it only makes sense if visualized. Finally, a feature combination in "if-then" conditions is defined as a **Rule** type, expressing the conditions that lead to the model's decision. Features used on the explanations are usually low-level and are difficult to grasp, even for a technical persona, such as Data Scientist [46], since the feature names or its values could be meaningless. The example of feature contribution explanation is shown in figure 2.1.

An **Example-based** output provides examples of instances as an explanation, like similar training examples that explain the instance of interest. Besides that, we group a

Table 2.4: Classification of the explanation methods by its output type.

Output Type	Output Sub-type	Example
Feature-based	Feature contribution	Permutation Feature Importance [61], LIME [75], SHAP [55]
	Feature visualization	Partial Dependency Plot (e.g. *pdp* package for R language [32])
	Rule	Anchors [76]
Example-based	Counterfactual (or Contrastive)	Foil trees [92], CEM [24]
	Similar example	MAPLE [67], LEAFAGE [1]
Model internals	-	Variable-wise Hidden States [37]
Concept-based	Internal semantics	ACE [28]
	External semantics	TCAV [46]

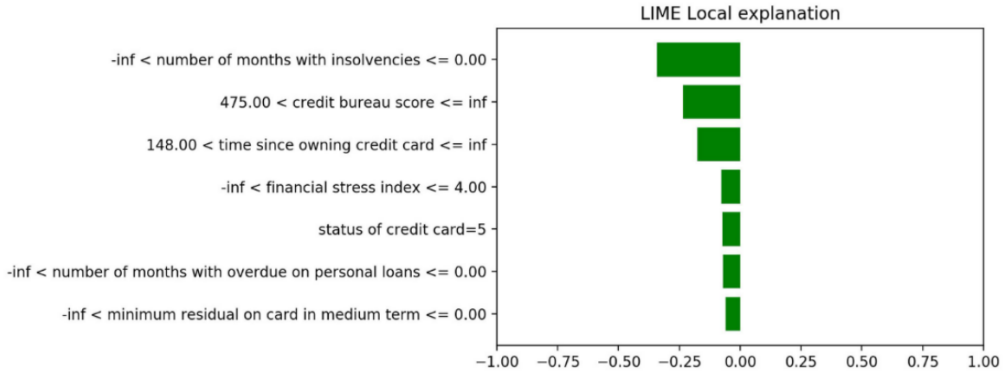


Figure 2.1: Example of feature contribution explanation. In this example, an algorithm LIME [75] produces explanations for Credit Risk [91].

Contrastive and **Counterfactual** sub-types since they show the example that is similar to instance of the interest but has another label. It basically answers the question "What are the minimal changes in the instance to change its class?". Figure 2.2 shows an example of such type of explanations.

Model internals reveal some internal components of the ML model in a human-understandable way, giving insights about how the decision was made. For instance, in use-cases with images, it is possible to visualize what the hidden units learned in the CNNs architecture do, hence revealing the internal component of the model's structure [11].

Finally, the **Concept-based** output can be divided in **Internal** and **External** Semantics. This concept-based output takes the advantage of target user's domain knowledge

Prediction: High

Most similar houses with value Low

Living Area	Year Built	Overall Quality(1-10)	Bathroom Amount	Bedroom Amount
135 m² (1456 ft²)	1978	6	2	3
137 m² (1479 ft²)	1976	6	2	3
133 m² (1441 ft²)	1978	6	2	3
135 m² (1456 ft²)	1976	6	2	3
113 m² (1218 ft²)	2009	6	2	2

Most similar houses with value High

Living Area	Year Built	Overall Quality(1-10)	Bathroom Amount	Bedroom Amount
171 m² (1850 ft²)	1994	7	2	3
194 m² (2093 ft²)	1986	7	2	3
181 m² (1950 ft²)	1997	7	2	3
194 m² (2097 ft²)	1993	7	2	3
149 m² (1614 ft²)	2005	7	2	3

Figure 2.2: Example-based explanation produced by LEAFAGE [1] for house price prediction. It shows similar and counterfactual examples as the explanations.

and uses that information to provide more high-level explanations, matching the user's mental representation. The concepts can be extracted directly from the input data, which we define as Internal Semantics or by linking the domain knowledge information from the external origin, defined as External Semantics. The main difference between those is that for the latter, domain experts must be involved in the process of information source creation. Then, the concept-based explanation method will use this pre-defined source of information to enforce the explanations. Alternatively, for the internal semantics, the methods automatically extract the concepts (e.g. the authors in [28] use image segmentation algorithm to extract parts of the image and use them as concepts in the explanation). Figure 2.3 shows some examples of this type of explanation output.

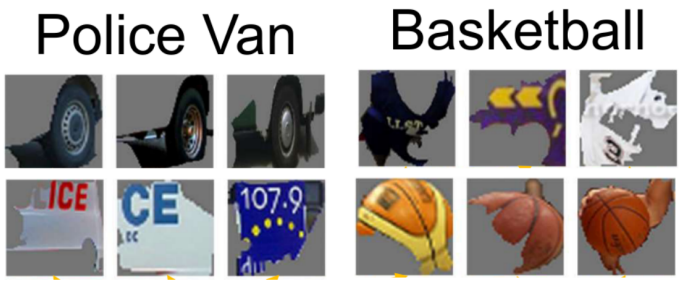


Figure 2.3: An example of high-level concept-based explanations for class "Police Van" (left) and "Basketball" (right) produced by ACE [28].

To best to our knowledge, there is no much work in concept-based explanation methods. We believe that the main reason for that is the complexity of creation concept-based annotated dataset. Also the definition of the external semantics is not straightforward, implying existence of the domain expertise to be able to define a taxonomy of concepts.

In this work, we will focus on the concept-based explanations, since our target personas are Fraud Analysts that, despite being non-technical, are the domain experts. By

enriching the explanations with domain fraud concepts, Analysts could match their mental representation of fraud patterns with the explanations, making the model's output more interpretable for them and in turn enhance their performance on daily tasks.

2.1.2 Methods and Techniques

Different explanation methods were proposed in recent years. Those methods can be divided into different categories, depending on the used technique. For instance, the technique called **Model Simplification** stands for building new simpler, and intrinsically interpretable models. These simpler models attempt to map the relationship between the predictions of the other more complex ML model and the inputs [75, 35, 92]. The resulting simpler models are usually called **Surrogate models**. The most well know [2] surrogate-based model simplification techniques is LIME, which stands for Local Interpretable Model-agnostic Explanations. LIME builds a linear model to locally approximate the more complex ML model, based on the perturbed samples around the point of interest [75]. LIME's output is feature-based, since each feature is assigned a coefficient (or contribution) of a linear model. Also, this method is local, Post-hoc, and both Model- and Data-agnostic. In short, LIME produces explanations by creating the perturbed dataset around the point of interest. It uses the complex model (i.e., black-box) to obtain the scores for the perturbations. Then, it weights each perturbation by computing a distance metric (e.g., euclidean distance) to the point of interest. Afterwards, LIME applies a feature selection algorithm to discern the most important ones. Finally, it trains a weighted linear model using the perturbed dataset with the predicted scores, and it returns the coefficients of the linear model as the final explanations. The coefficients (or "explanations" as authors refer) may be negative or positive. For instance, in Fraud Detection classification problems, the event is fraudulent (class 1) or legitimate (class 0). Thus, the negative coefficient in this case means that the feature decreases the prediction score, i.e., pulls the score towards negative class (legitimate). Otherwise, the features with positive coefficient increase the score towards positive (fraud). Figure 2.1 shows an example of negative contributions in Credit Risk use-case.

Although LIME is considered state-of-the-art of explainable methods, it has some problems when applied to tabular data, namely the perturbation process that is not straight-forward. For instance, for categorical features, a random value is chosen from all possible categories. This process presumes that features are independent and ignore the possibility of a correlation between them. The perturbed data can be unrealistic (e.g., feature *age=7* and *civil-status=married*) or as the final result it can produce redundant explanations, i.e., return correlated features which are related to the same phenomena. Also, the stability of LIME explanations is often discussed in the research community,

[2] Over 2000 citations in Google Scholar (https://scholar.google.com/scholar?cluster=16724302923321127943&hl=en&as_sdt=0,5)

showing that similar inputs, with minimal modification, can lead to completely different results [3, 2].

Different variants of the LIME were proposed in recent years, such as Deterministic LIME (DLIME) [98] that instead of using the perturbed sample, the authors propose to apply the hierarchical clustering to the training set and then, when the instance to explain comes, the label of its cluster is predicted by k-Nearest Neighbours (k-NN) algorithm. The problem with this approach is finding the optimal k (number of neighbors to consider) parameter for the k-NN. Another example of a LIME variant is k-LIME [38], where k-Means clustering algorithm is applied to create k groups and then for each cluster, a local Generalized Linear Model (GLM) is trained. Many other variants have been proposed to tackle the sampling issues associated with LIME. Note that all LIME-based implementations resort to low-level explanations based on feature contributions and are, therefore, difficult to grasp.

Still, considering the feature-based output methods, LIME authors present a rule-based explanation algorithm called Anchors [76]. This algorithm produce rule-based explanations and also introduce the notion of coverage of the anchor, claiming that the produced rule also applies to some portion of the unseen instances, besides the instance of the interest. Like LIME, Anchors also resorts to perturbations around the instance. However, it uses them to produce the candidate set of explanations rules from which it selects the ones with higher coverage. In other words, Anchors selects the rules that explain the largest number of neighbours. Even though the explanations produced by Anchors are more human-understandable (because they are in the form of rules), the produced predicates [3] can still be difficult to grasp for a non-technical persona. Another drawback of this approach is the poor validation of the hyper-parameters used to produce the explanations. Moreover, similarly to other perturbation-based explanation methods, Anchors can produce meaningless perturbation instances that can lead to unreliable explanations.

Another variant of the Model Simplification is called **Knowledge Distillation** [54]. The idea behind this technique was introduced by [16] for model compression. The end goal is to approximate the functionality of the complex model by building a much simpler and faster model with similar performance. This is, data is passed through a complex model to get the scores probabilities for those and use it for training smaller models. Then, the authors [9] introduce the "matching logits [4]" method that uses the scores produced by the model before the softmax activation (usually used in NN to normalize the scores between 0 and 1) instead of training with probabilities, claiming that this method improves the training of the smaller model. In the XAI field, the authors in [54] use the Knowledge Distillation to improve the interpretability of the complex ML models by using the black-box to produce logits and then use them for training a decision tree that is self-interpretable. While the Knowledge distillation technique does not exhibit

[3] terms that forms the rule
[4] logarithm of predicted probabilities

the limitations associated to perturbation-based methods, it still presents a few problems. Firstly, its outputs are feature-based. This means that even though the model is simpler, it is still based on the same non-interpretable features of the more complex model and, therefore, explanations are still difficult to grasp and interpret by non-technical personas. Also, even after distilling the complex model, the resulted simpler one may have a lot of parameters [52] to reach approximately the same performance as the complex model.

Another technique used by post-hoc explanation methods is **Backpropagation pass** that uses the backpropagation algorithm to find the importance of the input features passing the signal from an output through the ML model [81]. One of the most promising methods on this group is DeepLIFT [81] that compares the activation of a neuron to the reference activation (some representative example) and assigns the score according to the difference. The authors claim that this method overcomes the problem of the previously proposed methods on this category, such as Layerwise Relevance Propagation [14], Integrated Gradients [69], Grad-CAM [80] and other variants that suffer from gradient saturation [39]. One of the main limitations of this group of explanation methods is being Model-Specific since they are only applicable to black-boxes that use the backpropagation algorithm. Also, some explanations methods in the backpropagation pass category can be data-specific. In [82], the authors resort to saliency maps to provide more visual explanations for the learned features of the CNN.

Moreover, some methods use **Architecture Modification** that stands for modification of the design of the ML model to improve the interpretability. One example of a modification can be adding new components or simplifying existing ones. For instance, an attention mechanism was used to improve the interpretability of the NN, highlighting the most relevant parts of the ML model, for instance, Recurrent NN hidden state [70, 20]. Note, however, that attention mechanisms have been shown to identify irrelevant parts [96] to use in the explanations. This technique is widely used in image and text domains [10, 56, 19], due to its visualization and validation ease. However, the same does not verify for tabular data, where the output of such methods becomes much more challenging.

None of the previous explanation methods uses external or internal semantics. Instead, they merely resort to the input features and, therefore, present low-level explanations. The usage of semantic in XAI is becoming more popular in recent years [5]. In [46], the authors propose a global concept-based explanation method that uses external semantics. This method is called "Testing with Concept Activation Vectors" (TCAV) and it quantifies the degree to which a user-defined concept is important to a classification result (for example, how sensitive a prediction of "bird" is to the presence of concepts "feathers"). As the input, the user defines a set of different high-level concepts of interest along with random concepts and the image to test how sensitive it is to the concept. Then, the linear classifier is trained on some activation layer's output, given the random and

[5] In 2019, there was the first workshop dedicated to the Semantic Explainability (http://www.semantic-explainability.com/)

user-defined concepts. Finally, the Concept Activation Vector is defined as an orthogonal vector to the decision boundary produced by the linear classifier that gives how the input image is sensitive to the given concept. One of the main limitations of this explanation method is being Model-specific, i.e., its purpose is to explain the ML models that use the activation functions to represent the data in high-dimensional space such as NN. Also, although this method was first proposed for the image domain [46, 22] and text-domain [41], we did not find any application of this method to the tabular data domain, which is the use-case for this work. Finally, the way that this explanation algorithm was designed forces the end-user to define the concepts to be tested. This method does not reveal the concepts that are important to the system but more to validate if the concepts that user want to test are present [28].

Another example of a concept-based explanation method is called "Automated Concept-Based Explanation" (ACE) [28]. This method uses internal semantics, meaning that it extracts the concepts from the input data without any external source. Since ACE is a global explanation method, it explains an entire class instead of a specific instance. So, firstly ACE segments all the provided images, passes each segment through the black-box to get the final layer's activations (the logits) and apply a clustering algorithm (the authors used the euclidean distance as a similarity measure) to form the concept examples for the given class. Also, the outlier segments are removed. Finally, by using the TCAV [46], previously presented, most relevant concepts are returned, i.e., the ones with the highest importance. Basically, ACE extends the TCAV method, providing the way to extract the concepts instead of forcing the user to define them a *priori*. Although this explanation algorithm overcomes some of the TCAV's problems, it is still Model-specific and data dependent. The authors present an algorithm specifically oriented to the image domain since the main component of the concept creation is oriented to the image segmentation methods. Also, this method assumes that all the concepts can be extracted by segmenting the image.

In another perspective, authors in [58] include the interpretability straight into the ML model architecture. They use autoencoders to generate the basis concepts rather than using low-level features.

In another vein, authors in [64] propose the use of domain knowledge to produce more comprehensive explanations in medical diagnosis tasks. Authors use ideas from LIME [75], this is, learn a local interpretable classifier that mimics the "black-box" model's output (also known as surrogate model), perturbing the data around the instance to explain. Unlike the LIME's random perturbation process, the authors of *Doctor XAI* propose the usage of domain ontology to generate more meaningful perturbations. In medical domain, which is the case studied in this work, each data point can be associated with one or mode clinical codes. Thus, authors decided to use multi-label approach, exploring the multi-label decision tree [65]. The output from proposed method is a rule-based explanation that contains codes from medical ontology.

All in all, most concept-based methods in the literature are specific to computer

vision. Indeed, to the best of our knowledge, there is currently no research work in what it comes to high-level concept-based explanations for decision-making tasks in tabular domain, *i.e.*, there is no explanation method that satisfy our target persona (Fraud Analyst) requirements: being **local**, **concept-based** and could be applied for **tabular** domain.

2.2 Multi-label Approaches

Multi-label learning concerns the problems where a single instance is assigned multiple labels simultaneously. This opposes multi-class and binary classification settings where each instance is assigned a single class at a time. Figure 2.4 shows the visual distinction between these three types of classification formulations.

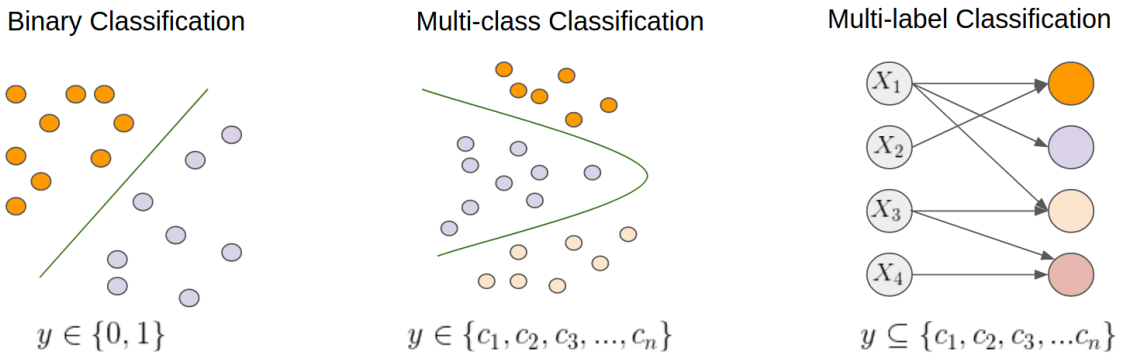


Figure 2.4: Difference between types of classification problems. X_i stands for the feature vectors, c_i represent different classes which are also represented by different colors. Each colored circle represents the instance that belong to some class c_i

Represented in figure 2.5, a multi-label setting consists of two main components: the data and the algorithm. There are two main approaches related to that components. The first one is the **Problem Transformation** that transforms the multi-label problem into one or more binary classification, regression, or ranking problems [18]. After transforming the multi-label into single-label data, use single-label ML algorithms to address the simplified problem. Unlike the previous approach, **Algorithm Adaptation** methods affect the algorithm itself extending them to handle multiple labels (e.g., extending k-nearest neighbors to predict multiple labels [105]).

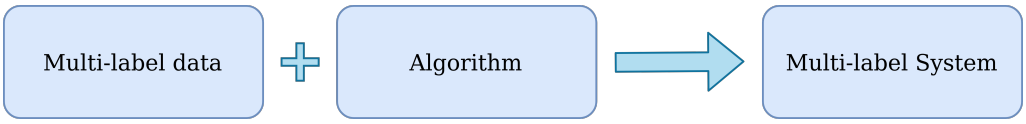


Figure 2.5: Multi-label System main components.

Given that the work developed concerns binary classification tasks , the literature review will merely focus on this setting.

2.2.1 Problem Transformation

As previously stated, the main idea of Problem Transformation is to transform the multi-label data in such way that traditional ML algorithms could handle it, without any modification.

One of the simplest group of techniques is called **Simple Transformations** [84]. These methods concern the application of transformations to the data, like selecting, copying, or ignoring specific instances. One example of such transformation is the **select**, where for each multi-labelled instance it will pick a single label according to the desired criterion, e.g., select-max picks the label with highest frequency, select-min picks the one with lowest frequency, and select-random picks randomly. Other possible simple transformation is **copy**, where each multi-label set is transformed in single-label instance, creating one instance per label, treating the result as multi-class classification. The variant where each created single-label instance has weight term ($\frac{1}{|Y|}$, where |Y| is the length of the label set of a specific instance) is called **copy-weight**. Finally, the **ignore** transformation discards all the multi-labelled instances and keeps the single ones. Although the idea behind these methods is simple, the loss of the information is a big issue. As an example, consider the credit card fraud setting discussed earlier, where each transaction is associated with multiple fraudulent patterns. Applying the ignore transformation to this dataset implies that we are only keeping the instances for which there is only a single fraudulent pattern associated. Naturally, this violates our problem statement and, as far as we know, there is no evidence that these methods have good performance in multi-label learning problems.

Other and most common method in the Problem Transformation is called **Binary Relevance**, where one independent binary classifier (C) is trained for each label. Figure 2.6 (left) shows one example of such method. While predicting, each instance can be assigned to multiple labels. One of the main issues with this approach is that it ignores labels correlations [103]. To tackle this problem, binary classifiers may use a **chaining structure**, where each consecutive model uses a previously predicted label along with the original input [74]. Figure 2.6 (middle) illustrates this type of structure. However, the order by which each label is predicted might lead to significantly different results, since each label influences the prediction of the proceeding label. To address this, the same authors propose an ensemble of classifier chains, where instead of only one chain, several are trained, using voting procedure to select the predicted labels [74]. Still, this method presents a few limitations related to the poor performance of the first classifiers of each chain, i.e., the errors of each previous classifier will be propagated for the consecutive ones [103]. In contrast, instead of having a chain structure, [31] propose to use a stacking structure, as showed in figure 2.6 (right). In this approach, some base-level binary classifiers provide the predictions to the next layer of classifiers, called meta-level classifiers. The figure 2.6 summarizes the main Binary-relevance-based techniques. Other approaches that tries to solve the labels correlation problem are based on **Bayesian network structures** that explores the correlation through the directed acyclic graph (DAG) structures [104, 5],

instead of stacking or chaining the labels.

Other technique is **Label Powerset** that transforms multi-label problem in multi-class approach, treating each unique combinaton of labels as a new class [87, 89]. The main drawback of this method is the computational complexity, since the number of classes could be very high with only few examples for each unique class. Also, it may have poor generalization, considering that only combinations that appear in training set can be predicted [102]. Several improvements to the Label Powerset method were proposed. One such example is the **Pruned Label Powerset** [73] the reduces the label combinations cardinality or **Random k-Labelsets** (RAkEL) [88] that creates an ensemble of multi-class learners trained on k label sets randomly selected, reducing the computational complexity.

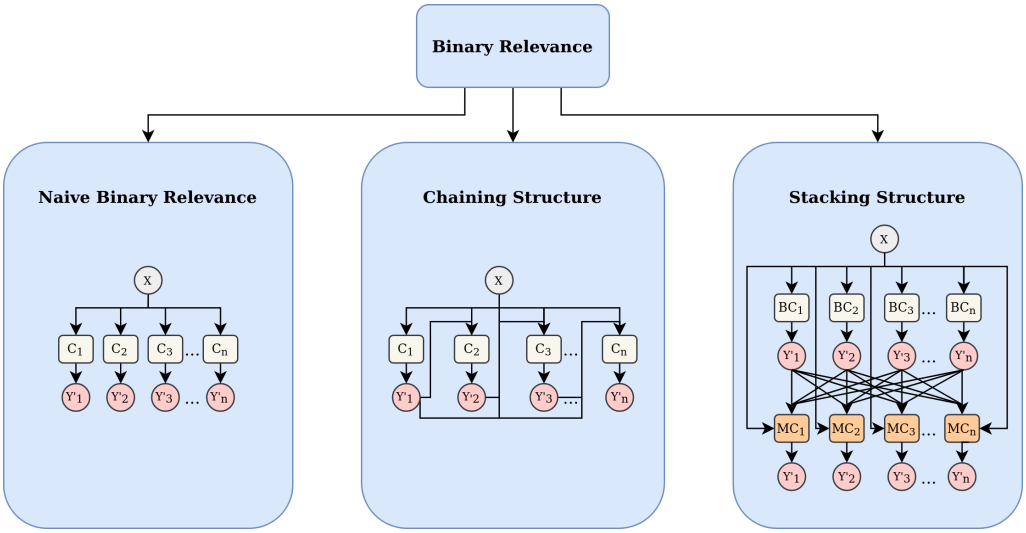


Figure 2.6: Binary Relevance variants: **Naive Binary Relevance**: train independent classifier (C) for each label, where N is the total number of labels. **Classifier Structure**: methods that make chain of classifiers (C), where each consequent classifier's feature space is expanded with the predictions of the previous classifier in the chain. **Stacking Structure**: Each meta-level classifier (MC) receives all the predictions from base-level classifiers (BC) along with the original features.

2.2.2 Algorithm Adaptation

Unlike the previous approach, **Algorithm Adaptation** methods affect the algorithm itself extending them to handle multiple labels (e.g., extending k-nearest neighbors to predict multiple labels [105]). This methods may have better performance than Problem Transformation since their internal structure is specially adapted to handle multi-label [27]. There are different base models that were extended to the multi-label task, such as tree-based models [21], where the *Entropy* function is modified to handle the multi-output. Also, a NN is widely used in multi-label classification problems [106, 94, 72, 63, 97, 93], where

the authors use a standard BCE function to model multi-label output, or more complex loss functions (*e.g.* ranking functions) [106].

In this work we will be focused in Algorithm Adaptation approach, focused on NN. Such type of models support out-of-the-box multi-label problems, offering flexibility regarding the loss functions or different types of architectures to use.

2.3 Remarks

The XAI field is incipient. There is still no consensus in interpretability definition, what is a "good" explanation and also in the evaluation process. As a consequence, the need for a definition of the requirements for different persona has emerged, since divergent end-users require different type of explanations.

Besides that, some explanation algorithms present issues related to the process of the perturbation (e.g., ignoring the possible feature correlation). Additionally, the current state-of-the-art methods produce low-level explanations that are difficult to grasp for non-technical persona. Thus, decision makes for instance, despite having domain knowledge, can not match their mental representation with the presented explanations. Some recent work shows the examples of the explanation algorithms that produce concept-based explanations that tries to mitigate the existed domain knowledge gap. However, this type of algorithms are focused on the image and NLP domains. Also, another challenge is the lack of concept-based annotated datasets. This raises the need for manual labeling that involves the domain experts, which is infeasible due to associated high costs.

In this work, we developed a local self-explainable model that produce fraud score and also associated fraud concepts. Moreover, each transaction can be associated with more that one concept (e.g., transaction could have *suspicious items* and also *suspicious email*)). Thus, it can be formulated as a multi-label problem, where each input can be classified with more than one class label. We decided to use the NNs models due their flexibility. We provide the motivation and details this in the following chapter 3.

Chapter 3

A Framework for Human-Interpretable Explanations

In this chapter, we describe our solution: a NN framework to jointly learn a decision task and concept-based explanations using domain knowledge, and consequently, easy to understand for the human-experts in the loop — the fraud analysts.

3.1 Problem scope and target persona

We consider the fraud analyst as the target persona. As fraud domain experts, their task is to review many transactions, classifying them as fraudulent/legitimate, as fast and accurate as possible. To assist in this endeavor, fraud analysts have access not only to historical transactions, optionally grouped by linked entities (*e.g.*, credit card, billing address) but also to the score returned by an ML model.

Given the larger generalization capability of most ML models, one would expect these human-AI cooperative systems to actually promote more efficient and effective decisions. However, that is not always the case. In the absence of information about the model's *rationale*, the human either regards the model as untrustworthy and completely ignore its output, or alternatively, wastes additional time looking for clues in the available data to support its decision logic. However, in a high-stakes AI system like fraud detection, such hesitation and distrust in the ML model may translate in worse accuracy and slower decision time, thus impacting many peoples' lives. To overcome this problem and gain the trust of the fraud analyst on the model, a possible solution is to provide explanations for the model prediction.

Despite the wide availability of explanation methods, the most popular methods are feature attributions, such as SHAP[55] or LIME[75], which are too low-level and difficult to understand by non-technical personas, as fraud analysts (see Chapter 2). To better

understand their explanation needs, consider the following reasoning of a fraud analyst reviewing process:

"This customer has **several transactions** within last 10 minutes with **suspicious e-mail**. It seems **high speed ordering** pattern. It is **fraud**!"

While fraud analysts lack technical background on data science and ML, they are extremely skilful at identifying fraud patterns and distinguishing fraudulent from legitimate behavior. Given their expertise and reasoning process, a more suitable explanation would be to highlight the suspicious or legitimate behaviors associated with a given model's decision. That is, it would identify the reasons or concepts for the model's prediction. This type of explanation would allow fraud analysts to quickly validate the necessary information to corroborate (or contradict) the identified concepts. Thus, instead of having to analyse all the available information or the low-level explanations (e.g., feature contributions), they can focus their attention on a specific piece of information and, consequently, improve the overall performance of the human + AI system.

Given the characteristics of our target persona and the pressing needs for (1) quicker and better reviews and (2) human-interpretable and diverse explanations (summarized in figure 3.1), we propose a self-explainable model that, in addition to discerning fraudulent from non-fraudulent transactions, also provides insights about the concepts associated with that decision.

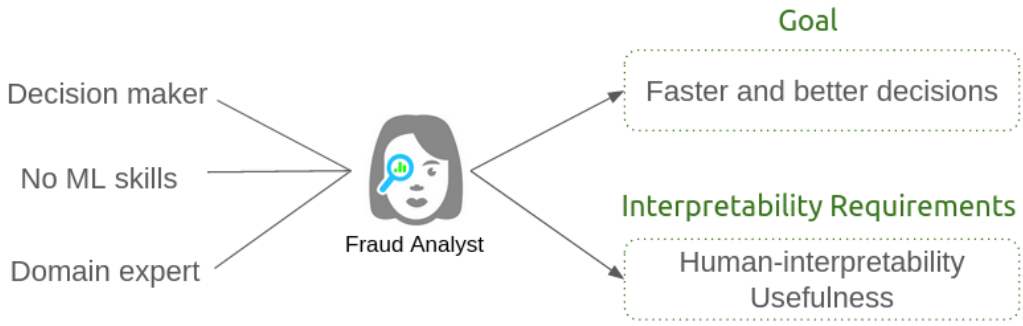


Figure 3.1: Description of the target persona, the fraud analyst, and her goals and interpretability requirements.

To this end, we leverage the domain expertise of fraud analysts to create more semantically meaningful explanations by collecting the focal domain concepts. Then, we use them to create a self-explainable model that produces both a fraud prediction and an explanation. Since fraudulent behaviors are akin to evolve over time, it is necessary to guarantee that our method is able to continuously learn and adapt to possible new fraudulent behaviors and/or to possible changes in the reviewing process by the analysts. In order to make the system adaptive to these behavior drift, we propose a cooperative system, where the self-explainable model is continuously learning from the human expert

(in this case, the fraud analyst), thus guaranteeing continuous predictive accuracy and interpretability improvements.

3.2 Jointly-learned Concept-based Explanations

In this section, we describe our framework for jointly learning a decision task and associated domain knowledge explanations. We assume the semantic concepts (used as explanations) will help the domain experts (end-users) reasoning throughout their decision-making process. We coin this framework JOEL, Jointly-learned Concept-based Explanations.

3.2.0.1 Neural Networks as Self-explainable Models

We decided to use an NN-based self-explainable model due to their flexibility and generalization capability. Firstly, simple Feed Foward Neural Network (FFNN) enable the combination of several neurons in innumerous ways, which allow us to adapt its generalization capabilities to the task at hands. Secondly, these models are easily adapted to the multi-label setting by simply changing the number of outputs neurons. Indeed, the concepts collected in the previous stage are trivially modeled as different classes and, since each transaction is simultaneously associated with multiple concepts, adding more neurons to the output layer allow us to effortlessly provide concept-based explanations. Thirdly, NNs support different loss functions, including functions that account for the label dependency, *i.e.*, ranking higher the correct set of labels and penalize the incorrect labels, instead of treating each label independently. In addition, these algorithms can be used to pre-train a base model and then use it for another task (*e.g.*, transfer learning), or for instance train model in large set of data and then adapt the NN weights for smaller set. This property is particularly useful in our use-case, since we can train the model with large dataset with labels crafted with Distant Supervision approach and then fine-tune the model using much smaller dataset with manually labeled dataset (for instance, asking real fraud analysts to manually provide the concepts for each transaction). Finally, NNs in general are considered stream-based algorithms, allowing continuous learning, *i.e.*, we can use back-propagation algorithm [1] to update the NN weights when a new batch of data arrives. Thus, it is possible to repeatedly update the NN for each new batch of transactions that arrives to the system, collecting the Human Expert feedback.

However, NNs are complex algorithms with high number of parameters, meaning that it may take much time to optimize those algorithms for a given task. Moreover, this type of algorithms tend to overfit more easily, *i.e.*, inability to generalize to new cases. Also, a large amount of data is needed to achieve a good predictive performance. But this lack

[1] The back-propagation algorithm makes use of local derivatives and of the *chain rule* (derivatives) to propagate the impact on the loss $\mathscr{L}$ with respect to each parameter involved in the computation.

of the available data may be solved by using a semi-supervised approach. We discuss our solution to this in section 3.3.1.

Given the described advantages of the NNs, we decided to test different architectures and validate them. The following sections will provide the details of the architectures used in our solution.

3.2.1 General formalization

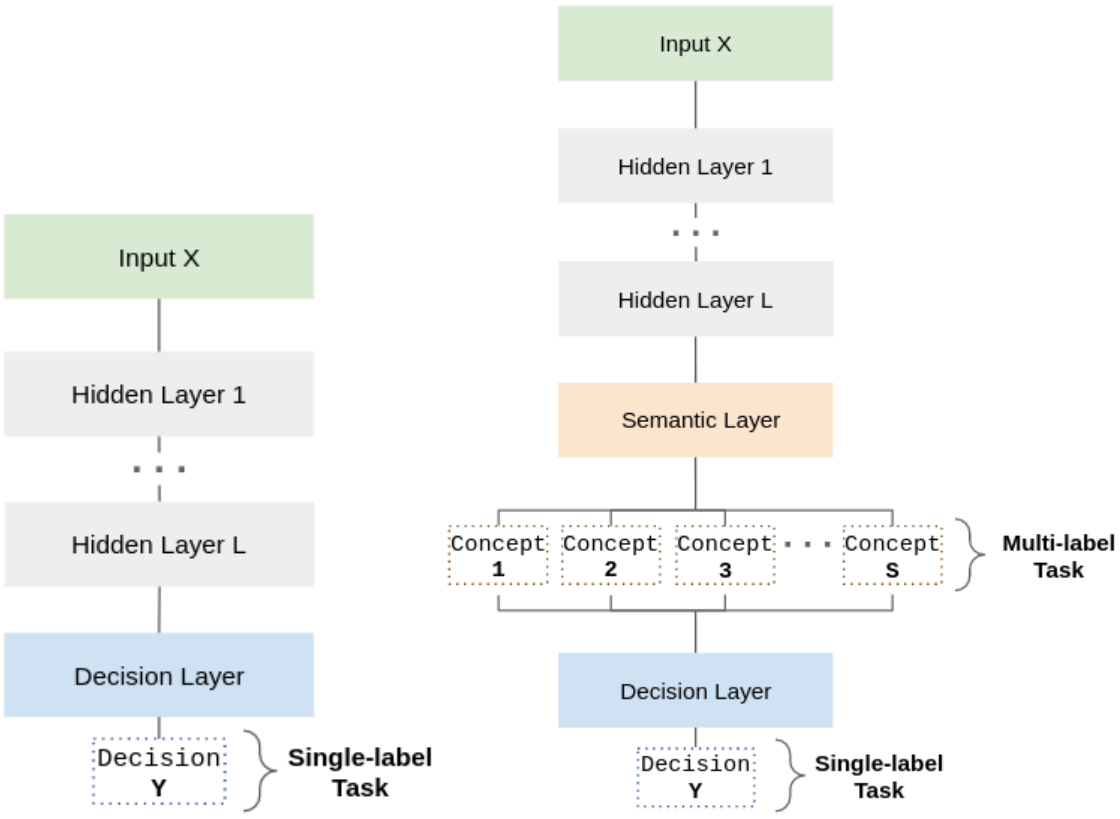
We frame the problem of learning both semantic concepts and decision task (*e.g.* fraud detection) as one of finding an *hypothesis*, $h \in \mathcal{H}$, that maps a vector of inputs (features), $x \in \mathcal{X}$, to a vector of semantic concepts, $s \in \mathcal{S}$, and to a vector of decisions (or classes), $y \in \mathcal{Y}$. In this formulation, the learning task becomes obtaining a mapping $h : \mathcal{X} \rightarrow (\mathcal{Y}, \mathcal{S})$.

Alternatively, we can pose our initially defined learning task as finding an *hypothesis* (learner), h, such that, for the same inputs, $x \in \mathcal{X}$, h is able to simultaneously satisfy $h : \mathcal{X} \rightarrow \mathcal{Y}$ and $h : \mathcal{X} \rightarrow \mathcal{S}$. For simplicity, we will henceforth refer to them as *decision mapping* ($h : \mathcal{X} \rightarrow \mathcal{Y}$) and *semantic mapping* ($h : \mathcal{X} \rightarrow \mathcal{S}$).

3.2.2 Architecture

JOEL is based on three building blocks: (1) NN, (2) semantic layer, and (3) decision layer. Due the NNs flexibility, we can arrange these components in multiple ways. In this work, we chained these blocks sequentially, creating a hierarchical structure. Figure 3.2(b) illustrates a JOEL's architecture. We first feed the input vectors (green), $x \in \mathcal{X}$ into a simple FFNN composed of L hidden layers (grey). The outputs of the FFNN, δ_L, are fed as inputs to the semantic layer (orange). Given that, the semantic layer outputs the domain concepts associated with a given input vector and which then serve as domain explanations. At the same time, the semantic layer's outputs, δ_S, are also used to impact the decision layer (blue). By chaining the semantic and decision layers, we are able to encode external information about the domain which is not available in the feature data. This can be particularly meaningful in cases where the taxonomy is intimately related to the decision task (*e.g.* a fraud taxonomy of fraud concepts is deeply correlated with the fraud detection task). Therefore, learning to accurately predict the domain concepts is likely to also lead to better decisions when it comes to the end-task (for instance, fraud detection).

Similarly to other NN architectures, our method learns through backward propagation of errors (*backprop*) and some variant of the gradient descent method. Unlike other NN models, the joint learning approach in JOEL's framework attempts to minimize both a *decision loss*, $\mathcal{L}_D$, and a *semantic loss*, $\mathcal{L}_S$. Mathematically, given model's parameters, $\Theta = [\theta_1, \theta_2, ..., \theta_L, \theta_S, \theta_D]$, the outputs of the decision layer, $\delta_D(x, \Theta)$, and the outputs of semantic layer, $\delta_S(x, \Theta)$, the gradient with respect to the loss, $\mathcal{L}$ at the semantic layer is



(a) Vanilla NN. (b) JOEL's architecture

Figure 3.2: Comparison between vanilla NN and JOEL's architecture example. Colors indicate layer type: input vector (green); hidden layer (grey); semantic layer (orange); decision layer (blue).

given by equation 3.1.

$$\nabla_{\delta_S}\mathcal{L}(x,y,s) = \nabla_{\delta_S}\mathcal{L}_D(\delta_D(x,\Theta),y) + \nabla_{\delta_S}\mathcal{L}_S(\delta_S(x,\Theta),s) \tag{3.1}$$

where:

$$\nabla_{\delta_S}\mathcal{L}_D(\delta_D(x,\Theta),y) = \frac{\partial\delta_D(x,\Theta)}{\partial\delta_S}.\nabla_{\delta_D}\mathcal{L}_D(\delta_D(x,\Theta),y)) \tag{3.2}$$

The decision of which loss functions to use within each task depends on the nature of the task. As the semantic layer corresponds to a multi-labeling task, we opt to use the *sigmoid* function and apply it to each individual entry of the output, before using it in the loss function. To find the mapping that simultaneously satisfies $h : \mathcal{X} \rightarrow \mathcal{Y}$ and $h : \mathcal{X} \rightarrow \mathcal{S}$ for a given input vector, $x \in \mathcal{X}$, we mutually minimize the (categorical) cross-entropy for both predictive tasks. For an input vector, $x \in \mathcal{X}$, a set of domain concepts, $s \in \mathcal{S}$, and decision classes, $y \in \mathcal{Y}$, we have that:

$$\mathscr{L}_D(x,y) = -\sum_{i=1}^{|\mathscr{Y}|} y_i \log[\text{softmax}(\delta_D(x,\Theta)_i)] \tag{3.3}$$

$$\mathscr{L}_S(x,s) = -\sum_{i=1}^{|\mathscr{S}|} s_i \log[\text{sigmoid}(\delta_S(x,\Theta)_i)] \tag{3.4}$$

Additionally, we can try different loss functions for the semantic layer. One of the possibilities is the loss function proposed in BPMLL [106] that is frequently used in the multi-label problems [107, 62, 33]. The main idea is to rank the correct labels higher than those that do not belong to the instance. We decided to apply this loss function to the *Multi-label Task*, *i.e.*, concept-based explanations prediction task. Equation 3.5 describes BPMLL loss function.

$$E = \sum_{i=1}^{m} \frac{1}{|S_i||\overline{S_i}|} \sum_{(k,l)\in S_i x \overline{S_i}} exp(-(c_k^i - c_l^i)) \tag{3.5}$$

where

- S_i is the set of correct concepts and $|S_i|$ is its cardinality.
- $\overline{S_i}$ is the set of incorrect concepts and $|\overline{S_i}|$ is its cardinality.
- c_k^i is the output of the NN on concept belonging to instance.
- c_l^i is the output of the NN on incorrect concept.
- $k \in Y_i$ and $l \in Y_i$.

This loss function should output larger values for the correct concepts and smaller for the concepts that not belong to the classified instance, and it is achieved by passing the negative of the difference between c_k^i and c_l^i to exponential function.

3.2.3 Baseline Architecture

Given the formulation in section 3.2.1, the learning task in vanilla multi-label NN becomes obtaining a mapping $h : \mathscr{X} \rightarrow (\mathscr{Y}, \mathscr{S})$.

Figure 3.3 shows a Multi-label NN. This vanilla Multi-label NN shares the same basic structure as the previously presented JOEL's architecture (see 3.2.2), *i.e.*, it receive an input (green) that is processed by a simple FFNN with L hidden layers (grey). The last layer L produces the δ_Q, where $Q = \mathscr{Y} \cup \mathscr{S}$. That means that both decision classes and concepts will be arranged in the same layer, minimizing the *total loss*, $\mathscr{L}_T$. For each $q \in Q$ we apply the *sigmoid* activation function and again, we use the (categorical) cross-entropy.

Given input vector $x \in \mathcal{X}$ and joined set of decision classes and concepts, $q \in Q$ we defined the :

$$\mathcal{L}_T(x,q) = -\sum_{i=1}^{|Q|} s_i \log\Big[\text{sigmoid}(\delta_Q(x,\Theta)_i)\Big] \tag{3.6}$$

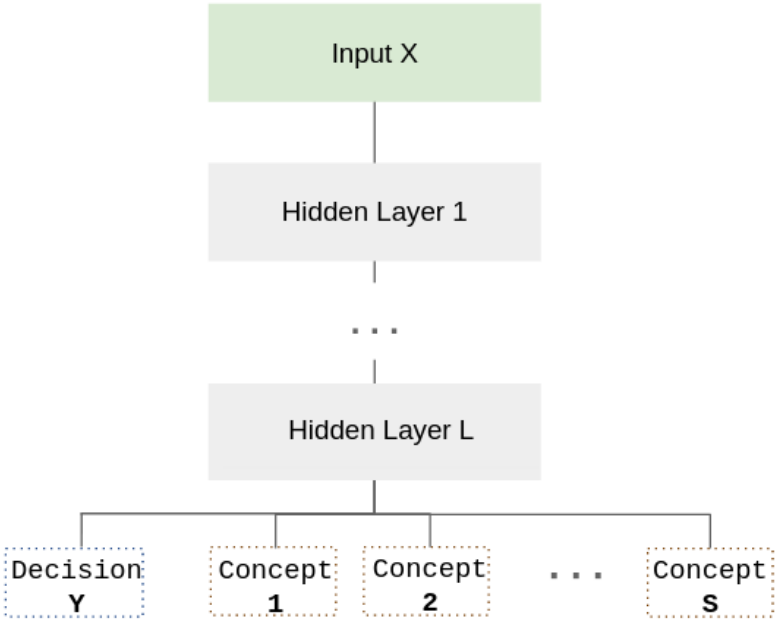


Figure 3.3: Vanilla Multi-label NN where set of concepts, $s \in \mathcal{S}$, and decision classes, $y \in \mathcal{Y}$, are placed on the same level.

We define the Multi-label NN to compare its results to our JOEL NN-based framework. This is, we will use the vanilla Multi-label NN as the baseline in this work.

3.3 Implementation

Training NNs requires large amounts of data in order to attain reasonably good performance [78]. Specially in the case of our JOEL framework we need a large training set comprising not only fraud annotations, but more importantly, multi-label concept-based annotations. On top of that, difficulties associated with the collection and creation of concept-annotated datasets make NN-based explanation methods that use semantic concepts infeasible in many practical settings. Moreover, the classical AI-based decision making systems include a human-in-the-loop, whose feedback is poorly explored.

To overcome the previously mentioned limitations, we leverage the components of the classical AI-based decision making system (legacy rule-based system, training ML pipeline, and human-in-the-loop feedback) to assemble a Continuous Cooperative Human-AI system consisting of three main stages:

1. **Semi-supervised learning** stage: focus on the creation of a multi-labeled dataset containing concept-based annotations, using semi-supervised learning techniques, since the manual labeling is costly.

2. **Model training and selection** stage: concerns the training and evaluation of different architectures and hyperparameter configurations for the self-explainable model. The model with highest predictive accuracy on the target label on the test set is selected.

3. **Human Teaching** stage: responsible for ensuring the model continuously learns from the feedback given by the human-in-the-loop.

Figure 3.4 illustrates a execution flow of the system with self-explainable NN model. This system starts by building and preparing the necessary data (*stage 1*, explained in section 3.3.1), which is then used for training different NNs (*stage 2*, explained in section 3.3.2). The best performing model according to some criteria (in our use-case it is fraud prediction) is then selected for deployment, that is, for integrating the human-AI cooperative system and aid in the analysts' decision making process. Upon the arrival of a transaction, the model provides a fraud score as well as the concepts associated with its decision. Using this information fraud analysts (domain experts) are able to quickly look for the necessary information to corroborate (or contradict) the provided concepts. Finally, when reporting their decision, the analysts are able to provide their own feedback about the concepts they identified in the transaction. This mechanism is crucial to mitigate feedback loops, as well as to guarantee the model continuously improves from its mistakes and adapts to the analysts' mental process (*stage 3*, explained in section 3.3.3).

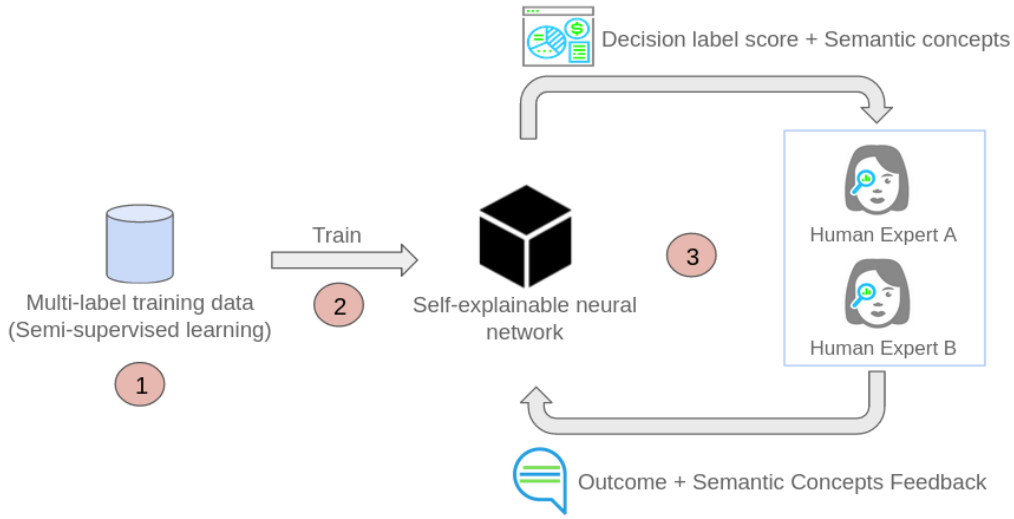


Figure 3.4: Overview of the proposed system. Stages represented by numbered circles: data preparation (stage 1), model training and selection (stage 2), and human teaching (stage 3).

The following sections describe the implementation details of each stage. In particular, in section 3.3.1, we detail how we leverage the fraud domain concepts to create a dataset for training the self-explainable neural network (*i.e.*, a concept-based annotation dataset). Furthermore, we discuss how we create the dataset without the need for manual labeling intervention. In section 3.3.2, we explain the model selection pipeline. Then, in section 3.3.3, we describe the techniques used to update the model, *i.e.*, the mechanisms used to guarantee that the self-explainable model is able to continuously improve its predictive and explainability capabilities. Specifically, upon receiving the human-in-the-loop feedback. Finally, we will describe the main part of the algorithm, referring to how the described steps are interconnected and interacts with each other.

3.3.1 Stage 1: Semi-Supervised Learning

To train a self-explainable NN presented in section 3.2, it is not enough to consider the original dataset, *i.e.*, dataset containing fraud annotations. Instead, it is necessary to populate with domain information that reflects the reasoning process of the fraud analysts, like fraud concepts presented above. Thus, in addition to the fraud/non-fraud annotations commonly used in fraud domains, this custom annotated dataset must also comprise fraud domain concepts associated with each legitimate (or illegitimate) behavior.

Since to the best of our knowledge, there is no publicly available dataset with similar properties (fraud domain concept annotations). At the same time, the manual labeling is costly and time-consuming, making the process of the dataset creation very arduous.

Algorithm 1 Pseudo-code for semi-supervised learning stage

Input: $\mathfrak{D}$ is the original dataset;
mappingDict is a dictionary that associates each rule with one or more domain concepts;

```
1:  function semiSupervisedLearning(𝔇, mappingDict)
2:      𝔇_ML ← empty
3:      for event ∈ 𝔇 do                          ▷ apply Distant Supervision technique
4:          triggeredRules ← getTriggeredRules(event)
5:          for triggeredRule ∈ triggeredRules do
6:              mappedConcepts ← mappingDict.get(triggeredRule)
7:              event.addLabels(mappedConcepts)
8:              𝔇_ML.append(event)
9:          end for
10:     end for
11:     return 𝔇_ML
12: end function
13:
```

To circumvent these limitations, we make use of a semi-supervised learning approach to create the concept-based annotated dataset. We decided to use Distant Supervision approach for automatically labeling of data that premise that two entities can be associated

in someway [59, 30, 101]. In fraud detection, the system matches the arrived financial transaction against a set of rules to determine a risk score (*e.g.*, when the transaction matches some suspicious behavior). Normally, the rules in the fraud detection context are pre-defined together with fraud specialist and are dependant on the specific use-case. Thus, we can adapt the Distant Supervision approach, exploiting the mapping between concepts from a pre-defined Fraud Taxonomy and a set of fraud domain rules. For this approach, the definition of concept taxonomy is a requirement, since it is based on the mapping between entities from the taxonomy and rules in the existed legacy system. The pseudo-code presented in algorithm 1 shows each step taken to produce the concept-based annotation dataset which labels were created by Distant Supervision. For each event (line 3) we get the set of the triggered rules (line 4). Then, for each triggered rule (line 5) we get associated concepts from fraud taxonomy (line 6), using them as the "noisy labels" for the event (line 8). Like this, we create a concept-based annotation dataset, taking the advantage of a legacy rule-based system in AI-based decision-making system.

Table 3.1: Example of Rules-Concepts mapping in a fraud detection setting.

Canonical rule name	**Description**	**Mapped concepts**
risky_product_styles	Order contains risky product styles.	Suspicious Items
cards_used	User tried *N* different cards last week.	Suspicious Customer, Suspicious Payment

Table 3.1 shows three examples of possible mappings between rules and concepts in the fraud domain. Given the strong similarity between them, it becomes possible to apply the Distant Supervision technique to create the concept-annotated dataset by creating the association between each triggered rule (per transaction of the training set) to the most similar concept (dubbed "noisy label").

Unfortunately, one of the possible issues regarding this semi-supervised method is the reduced number of mapped rules. This is, a concept with low number of matched rules may be poorly represented, hindering the self-explainable model's learning process. In section 4.3, we further discuss the application of the Distant Supervision in real-world setting and how these limitations can impact real-world systems.

3.3.2 Stage 2: Model training and evaluation

We decided to use Neural Networks as self-explainable model, since it support multi-label classification out-of-the-box and are very flexible, allowing to test different architectures and techniques (like transfer learning [108]). We provide more details on this in section 3.2.0.1.

After creating the concept-based annotated dataset, we train different combinations of NNs architectures and hyperparameters. Figure 3.5 illustrates the model selection pipeline for determining the final self-explainable model: each network is trained on the

Algorithm 2 Pseudo-code for training and selecting the model algorithm

Input: $\mathcal{D}_{ML}$ is a dataset containing concpt-based annotations;
paramsGrid is a set of hyperparameters to find the best model;

1: **function** TRAINANDSELECTMODEL($\mathcal{D}_{ML}$, *paramsGrid*)
2: $\mathcal{D}_{MLprocessed} \leftarrow$ PREPROCESS($\mathcal{D}_{ML}$)
3: trainSet, validSet, testSet, prodSet $\leftarrow$ DATASETSPLITTER($\mathcal{D}_{MLprocessed}$)
4: modelsGrid $\leftarrow$ MODELTRAINING(trainSet, validSet, *paramsGrid*)
5: selectedModel $\leftarrow$ SELECTBESTMODEL(modelsGrid, *targetLabel*, testSet)
6: **return** selectedModel
7: **end function**

previously collected dataset and evaluated on the test set. Then, the best performing model according to some criteria is selected and used in an improved human-AI decision making system. In our use-case we chose the model that better discerns fraudulent from non-fraudulent behaviors.

Figure 3.5: ML pipeline for selecting the concept-based self-explainable model.

We describe each step of this stage in the algorithm 2. Before training the models, it is necessary to pre-process the data (line 2). This includes standard procedures such as duplicates removal, missing values imputation, categorical features processing, and numerical data normalization. To guarantee the correctness of this procedure, we divide the dataset in a three-way split, consisting of training, validation, test sets, and production set (line 3). We use a training set to train the model and validation set to trigger the early stopping criteria (denoting the end of the training loop). The test set will be used to select the best model, at pre-defined fixed FPR (explained in section 4.6). Also, since we are working in real-world setting, we decided to use a production dataset that will provide an estimation of how could the model behave if used in production (*i.e.*, if it could be deployed in real-world application). We provide more details on the dataset division and how they are used in section 4.2. Furthermore, we rely on hyperparameter optimization search to iterate over a grid of hyperparameters and NN architectures (line 4). With this procedure, we expect the best model to be selected according to some criterion (line 5). In our use-case, the chosen model will be deployed in a high-stake real-world environment, where the prediction accuracy on fraud is crucial due restricted requirements (the model must detect fraudulent events to prevent the client's losses). Additionally, we believe that the model with high fraud predictive accuracy will be also capable to learn associated fraud concepts.

3.3.3 Stage 3: Human Teaching

The final stage of the proposed Continuous Cooperative Human-AI method is Human teaching (or Human tuning). This process, illustrated in figure 3.6, feeds on the continuous feedback provided by the target persona (see section 3.1). Since we have the access to real expets' feedback, the "true" labels used in this stage are provided by a real human experts.

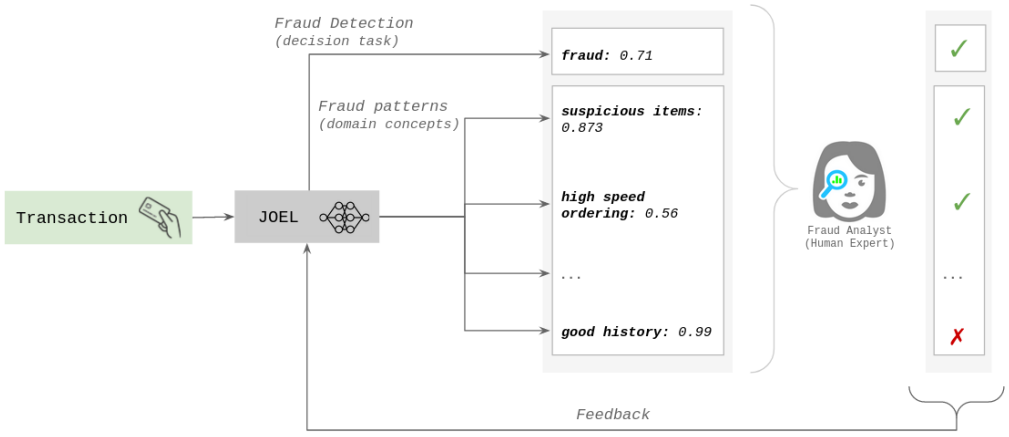


Figure 3.6: Human teaching overview.

In high level, the arrived transaction is processed by JOEL that outputs the fraud score (decision task) and also associated fraud concepts (domain concepts). This information together with transaction details is shown to the fraud analyst (human expert). The expert's feedback is collected and used to correct the self-explainable model's weights, improving the quality of the explanations.

Algorithm 3 Pseudo-code for Human Teaching stage

Input: selectedModel is a selected model with best predictive accuracy on target label;
E is the set of Human Experts;
event is an entity to be processed by a system;
tuningParams is a set of hyperparameters for model tuning;

```
1:  function HUMANTEACHING(selectedModel, E, event, tuningParams)
2:      predictedTargetScore, predictedConceptsScore ← selectedModel.PREDICT(event)
3:      expertFeedback ← EVENTREVIEW(event, E, predictedTargetScore, predictedConceptsScore)
4:      feedbackDB ← SAVEFEEDBACK(expertFeedback)
5:      if feedbackDB.EVENTSCOUNT() == tuningParams.GET(batchSize) then
6:          updatedModel ← UPDATEMODEL(event, expertFeedback, tuningParams)
7:      else
8:          updatedModel ← selectedModel
9:      end if
10:     return updatedModel
11: end function
```

The pseudo-code presented in algorithm 3 describes the Human Teaching stage. When new event arrives to the system, the self-explainable method produce the target label score, which is fraud label in our use-case, and also associated fraud concepts (line 2). When reviewing a transaction, the fraud analyst will receive, in addition to the transaction-related data, the fraud score and the fraud concepts associated with that score. With this information, the analyst is able to quickly navigate the reviewing interface in order to make a decision (whether to accept or reject the transaction). Fraud analysts are also required to provide the concepts that better describe their motivation, *i.e.*, their mental process for making a decision (line 3). Then, we collect the feedback and save it in database (line 4). If the database has enough events (line 5), it is used to fine-tune the self-explainable model using the backpropagation algorithm (line 6), thus ensuring its constant learning and adaptability to the analysts' mental models changes with repercussions on its predictive accuracy and explainability.

The flexibility of the Neural Networks allows to fine-tune the already trained model. Thus, we can define new hyperparameters for the tuning process, freeze and unfreeze the weights of the pre-trained NN, or even append or disable some model's component. This tuning phase may be performed as the normal hyperparameter searching, *i.e.*, we can define a parameter grid and run it in order to find the best tuning parameters. For this, we may adapt a stream-based evaluation that organizes the collected feedback in ordered by timestamp batches and then, use each batch to evaluate and then perform a backpropagation pass to update the model's weights, repeating this process until end of the batches. We describe this procedure more deeply in the section 4.6.

Ideally, including the human experts in the loop will help the AI to get better at explaining itself and at the same time, self-explainable model will improve the efficiency of the experts.

3.4 Framework Overview

The algorithm 4 provides a pseudo-code description of the generic workflow of our framework, where we show how each stage described in previous sections interconnects and interacts with each other.

The workflow starts by verifying the existence of an already prepared concept-based annotated dataset (line *1*). If no such dataset exists, it first creates a mapping (line *2*), with help of the human expect (E), between set of fraud domain rules and concepts from a pre-defined fraud taxonomy. Then, by using semi-supervised learning technique, that is Distant Supervision in our use-case, we create a concept-based annotation dataset (line *3*). After creating dataset, we train, evaluate and select the self-explainable NN that predicts the outcome and also the associated concepts (line *4*). The selected self-explainable model is deployed and a new stream starts (line *5*). When a event arrives (line *6*), it is passed through "Human Teaching" method, returning the new updated model (if there there is enough collected data for the tuning) (line *7*). Finally, an updated version of the model

Algorithm 4 Pseudo-code for the workflow of our framework for human-interpretable explanations.

Input: E is the set of Human Experts;
$\mathfrak{D}$ is the original dataset;
$\mathfrak{K}$ is a set of domain rules;
C_{tax} is a taxonomy of the Domain concepts;
paramsGrid is a set of hyperparameters to find the best model;
tuningParams is a set of hyperparameters for model tuning;

```
1: if conceptBasedAnnotations is_not_prepared then      ▷ Check if the concept-based annotation dataset already exist
2:     mappingDict ← createMapping(E, C_tax, 𝔎)
3:     𝔇_ML ← semiSupervisedLearning(𝔇, mappingDict, C_tax)
4: end if
5: selectedModel ← trainAndSelectModel(𝔇_ML, paramsGrid )
6: stream ← newStream()
7: while stream.notClosed() do
8:     event ← stream.getNextEvent()
9:     updatedModel ← humanTeaching(selectedModel, E, event, tuningParams)
10:    selectedModel ← updatedModel
11: end while
```

is used and the looping is continue until the end of the stream, *i.e.*, when the system's end-user decides to stop this continuous learning process.

In this section we presented our framework to jointly learn a decision task and associated concept-based explanations using domain knowledge. Our framework produce human-interpretable explanations that fit the information needs of the decision makers, which are fraud analysts in our use-case. Our solution is flexible, since each part can be modeled differently. Throughout this section we show some possible instances of this framework like JOEL architecture where we can vary the loss functions, or even the structure and type of the NNs. Moreover, the data preparation stage is also modular, where different semi-supervised approaches can be used. In our use-case we used a Distant Supervision, leveraging already existed legacy rule-based system and creating a fraud taxonomy of concepts. Finally, we take the advantage of the human-in-the-loop, which are fraud analyst in our use-case, to collect the feedback and use it to improve the self-explainable method. In this stage, our solution also offer the flexibility to define different methods of incorporating human feedback.

Chapter 4

Experimental setup

We validate our solution in a real-world fraud domain Human-AI system. Thus, this chapter describes the experimental setup through which we conducted such validation experiment. We also detail some of the design decisions involved and emphasize the task specific ones, including a few practicalities when collecting the data, the evaluated hyperparameter grids, evaluation metrics, among others.

4.1 Raw Dataset

We use a real-world transaction monitoring dataset in E-commerce owned by Feedzai.

Each instance represents a transaction (or event) and associated metadata, such as items in the basket payment details, as well as, client's history with the merchant. In total, the raw dataset totals approximately 600 features, from which only 111 are used by the model as features.

The raw dataset consist of approximately 9.3 millions of financial transactions, dated from January 2019 to 20 of November 2019. It is highly imbalanced, since only 2% of all the transactions are fraudulent. As illustrated in figure 4.1, we divide the raw dataset in 4 sequential sets: train, validation, test, and production.

For training, we consider 4 months and 20 days of data (from January to May), containing in total 4.4 million transactions. The fraud prevalence in training is approximately 2.5%. We use the validation set for determining the early stopping criteria (further explained in section 4.2). Its total size is 1.5 million transactions (from May 21 to Jun 29) and its fraud prevalence is 1.7%. Then, the test set includes 800k transactions (from June 30 until July 20). Its main purpose is to perform proper model selection in a holdout set and, consequently, obtain a robust estimate of the models' generalization performance. In this set, the fraud rate is 1.5%. Lastly, since we are working in real-world settings,

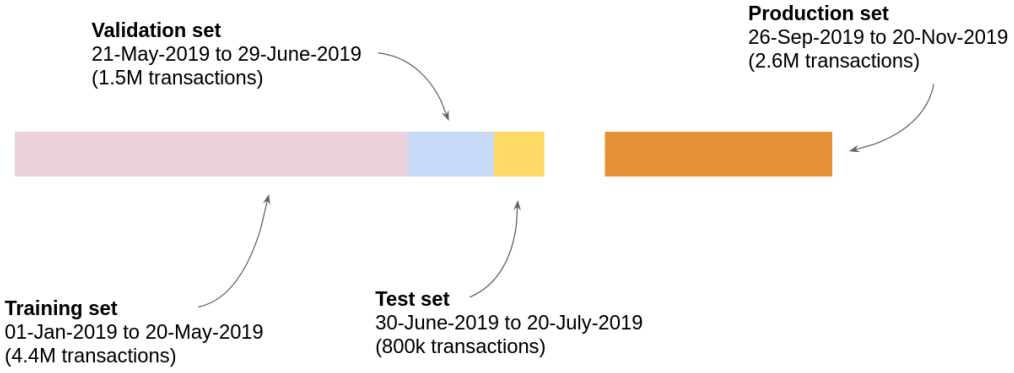


Figure 4.1: Dataset splits and dates.

we also include a production dataset with a 2 months time-shift (from September 26 to November 20), containing 2.6 million transactions of which only 1.5% are fraudulent. The 2 months gap simulates the time necessary to deploy a model in a real-world setting. For this reason, we argue that such setting will provide a more realistic perception of the self-explainable model performance in production.

4.2 Train-test stages

Frequently, ML practitioners divide the training set into mini-batches to speed up the convergence of NNs during training. Accordingly, an iteration consists of one forward pass followed by a backward pass, whereas an epoch accounts for passing all mini-batches. To address the severe dataset unbalancing (only 2.5% of the total transactions are fraudulent), we implement a mini-batch sampling mechanism to avoid having batches with 0% fraud prevalence. Instead, we force each batch to have a pre-defined fraud prevalence (*i.e.*, each batch has at least 1 fraudulent transaction), thus ensuring the NN is able to learn both fraudulent behaviors. In the case, no strict requirement is set on the minimum base rate at each mini-batch, it could happen that several batches would have 0% fraudulent transactions. Consequently, in some iterations the learning would be focused on learning legitimate behaviors and the weights would be updated regardless of the previously learned fraud patterns, thus undoing (and damaging) the learning process convergence.

The NN training loop stops whenever it completes the last epoch (it is a hyperparameter fixed by user), or when it triggers early stopping. The first criteria is fulfilled when the training loop went iterate over all defined epochs. The latter is triggered when the model is not improving after N number of epochs, where N is a defined hyperparameter, know as patience. Hence, the validation is used to measure the performance of the model after each epoch. In this work, we use a validation loss as the stopping criteria, *i.e.*, if the model is not improving the validation loss during N consecutive epochs, the early stopping is triggered and the training loop is finished.

We use the trained model to score the test set and calculate the evaluation metrics

with fixed threshold at pre-defined False Positive Rate (FPR). In this stage, we select the model that has a better performance for fraud label, since we want a model that can successfully detect fraud that is defined by business logic. Finally, after selecting the best model, we use the production dataset to evaluate its generalization capacity in a holdout dataset in a more realistic scenario.

4.3 Distant Supervision

As described in section 3.3.1, we use a Distant Supervision approach to craft a Multi-label dataset from original single-label dataset. The first step is to define the mapping between each possible domain rule in legacy rule-based system and one or more concepts from fraud taxonomy. Together with fraud expert we define a Rule-Concept mapping that is applied to data source that contains transactions with triggered rules and results in "noisy labels" that are mapped concepts. The transaction monitoring use case that we consider in this work contains around 300 rules that were manually analysed and associated with fraud concepts. Table 4.1 shows the number of rules mapped to each concept.

Thus, after applying the Distant Supervision, we ended up with Multi-label dataset, where each transaction's triggered rules were mapped to fraud concepts, creating "noisy labels". As the example, consider the event X for which rule A and rule B trigger. The performed mapping defines that rule A is mapped to Suspicious Email and Suspicious IP. Additionally, a rule B is associated with Suspicious Customer, Payment and Items. Thus, by applying the Distant Supervision, we end up with event X labeled with Suspicious Email, IP, Customer, Payment, and Items.

However, this method has limitations. Analysing the table 4.1 we find that Distant Supervision approach poorly covers the legitimate concepts, since the majority of the mapped rules are related to negative behaviour. Also, this semi-supervised approach is a weak signal for the real labels, since we are using triggered rules as a proxy for the concepts. For that reason we call the result of the Distant Supervision as "noisy labels".

Table 4.1: Number of mapped rules for each concept.

Concept	Rules mapped
Suspicious billing shipping	86
Suspicious Customer	58
Suspicious Payment	53
Suspicious Items	48
High speed ordering	31
Suspicious Email	27
Suspicious IP	24
Suspicious Device	22
Nothing suspicious	19
Good customer history	11
Suspicious Delivery	5
All details match	3

4.4 Manually Labeled Dataset

To validate our solution, we decided to simulate a real-world scenario where the self-explainable model will continuously receive the feedback from Human Experts. For this reason, we decided to create a small manual multi-label dataset that will contain the real feedback from the fraud analysts.

One of the main Feedzai's product is a case management application that is used by fraud experts to review suspicious transactions. This application interface displays the transaction's information, triggered rules (if any), and also the ML model score. The analyst has to approve or decline the analysed transaction, and in the end of the review, the user is asked to provide additional feedback such as comments about the transaction, or reasons for taken decision. We took the advantage of this and created a program routine that allow us to inject any list of pre-defined items and display it in the end of the review process. Also, we develop a program procedure that collects and saves all the feedback provided by the fraud analysts.

We had an opportunity to run an annotation campaign with three fraud experts, where we simulate the real review process. First, we inject the set of the pre-defined concepts in the application UI and then collect the concepts that analysts' associate with the reviewed even.

We ended up with the total of 1561 manual labeled transactions. Figure 4.3 shows the prevalence of each label. The review set used in the annotation campaign contains one month of data, starting from October to November. Figure 4.2 shows the overall dataset timeline, now including the manually labeled dataset.

Having this manual labeled dataset, we can evaluate the model that was trained with Distant Supervision approach and also validate the Human teaching stage, described in section 3.6.

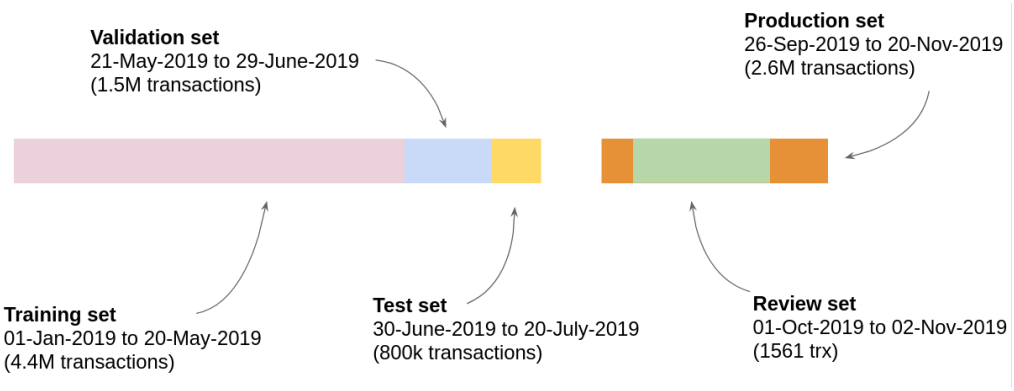


Figure 4.2: Updated dataset splits and dates, now including the review set.

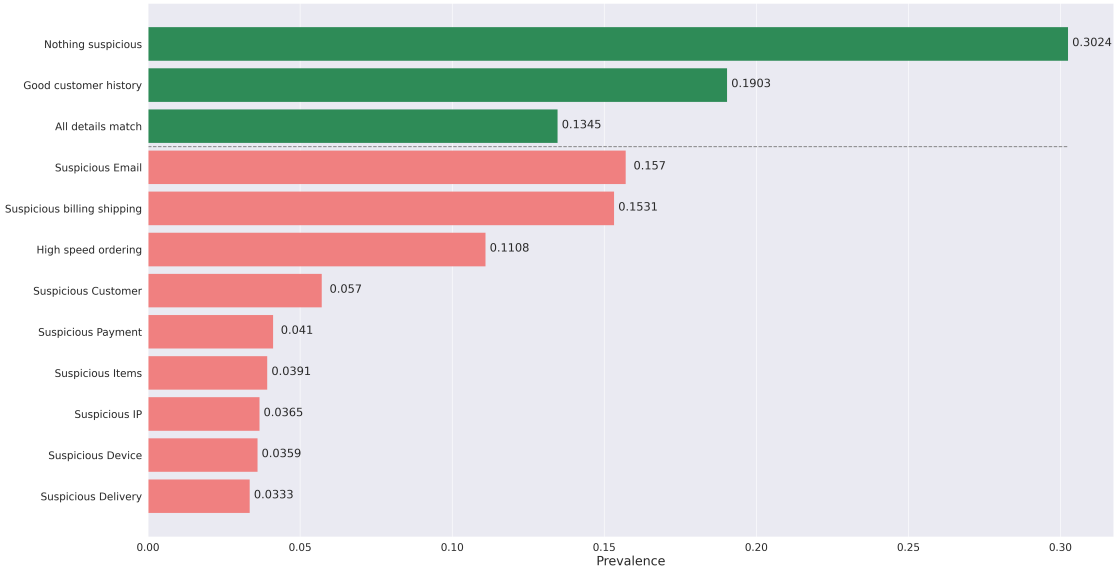


Figure 4.3: Label prevalence in manually labeled dataset.

4.5 Hyperparameters

NNs have several hyperparameters. In this work, we limit the number of hyperparameters to vary, ended up changing the following:

- Batch size
- Learning rate
- Number of hidden layers
- Dimensions of the hidden layers
- Batch Normalization (use or don't use)

- Number of Dropouts and its probabilities

We decided to fix the random seed and the total number of the epochs to 100, with patience of 5. Also the batch size is fixed to 4096. Regarding the optimization algorithm we chose the Adam [47], varying only the learning rate.

We focused on the number of FFNN layers and their dimensions. We vary between three and maximum eight hidden layers. Regarding the dimensions of each layer, defining the range between 128 to 16 neurons. We also vary the dropout probabilities per layer, defining the range between 0.6 to 0.1. We also decided to vary the Batch Normalization [42] since it is widely used in the state-of-the-art Deep Learning models.

In the Human teaching stage (described in section 3.3.3), we also define a parameter grid, but now we only vary the optimization algorithm[1], learning rate, number of epochs, and also the batch size.

4.6 Evaluation Metrics

In real-world fraud detection setting, the class ratio is frequently extremely imbalanced, *i.e.*, the fraction of fraudulent events (also known as fraud rate) is hugely small. For instance, in our training set described in section 4.1, the fraud rate is only 2.5%. Thus, by using accuracy as the main metric, a trivial classifier could achieve around 97.5% of accuracy classifying all the instances as legitimate. On the other hand, the same classifier would achieve 0% of recall. However, the main task is to catch fraudulent events, meaning that we need to pay special attention to the performance of the ML model on positive examples.

Therefore, as the main metric for the decision task (fraud detection) we use the True Positive Rate (TPR) (also know as Sensitivity or Recall) at 3% FPR. In real-world setting, FPR is frequently fixed at some pre-defined value, typically depend on business constrains. By fixing the FPR, we ensure that the system will reduce the probability of negatively affect legitimate users, since throwing false positive alerts for legitimate clients could harm the relationship.

For the explainability task, the predictions for each concept, that will be served as explanations, are evaluated using AUC. We do this because JOEL provides the score for each concept, and we are not making binary decision, *i.e.*, we will show the model's score for each concept. Thus, we choose AUC metric since it is a threshold agnostic metric. The evaluation results (after training and validation) require an aggregate measure of model's generalization at the multi-label task in holdout datasets. We use the mean AUC over all labels (predicted concepts). We decided to exclude the concepts "Other Fraud" and "Other Legit" from the averaging, since they are reserved for unknown cases. Using AUC as the performance metric, we do not need to define a threshold.

[1] We used all the available optimization algorithms in PyTorch framework here: https://pytorch.org/docs/stable/optim.html

Human teaching evaluation: Additionally, to validate the performance of the self-explainable model in real world setting, we used the manually labeled dataset, described in section 4.4. We split this dataset into training, validation, and test sets. After defining the tuning hyperparameter grid, we run it on the first 800 transactions. Then, we select the model with the highest mean AUC across the concepts (excluding "Others") on 200 transactions (validation set). After selecting the best hyperparameters, we re-train the Distant Supervision model (best Distant Supervision model trained on "noise labels") using both the training and validation sets. We estimate its generalization performance in an holdout test set composed of 561 transactions.

4.7 Implementation Details

We choose Python programming language (Python 3) for the practical implementation of the solution. Due the millions of processed transactions in such systems, we had access to huge amount of data. To process this data, we used the leading engine for big data processing called *Apache Spark* [99], more specifically *PySpark*, a Python API for Spark. The development of the NN architectures (see section 3.2), we used *PyTorch* [66] that is a very flexible open source ML framework. For instance, this framework offer a scalable distributed training, easy and intuitive way to implement different NN architectures, among others.

In addition, we decided to reuse some of the tools developed internally. We implemented the mechanism to inject a set of pre-defined concepts in the application's UI used by fraud analysts to review transactions. The collected feedback is stored in the PostgreSQL database, allowing the execution of SQL queries to aggregate the feedback. Also, we extended the base code of internal tool to develop NN models, which was focused on sequential models (*e.g.* RNNs), to support Multi-label FFNN and our hierarchical architecuture (see 3.2.2). This internal tool works with a simple parameterization by YAML files (A human friendly data serialization object often used for configuration settings [12]). All the steps related to initialization, training and evaluation of the architectures were configurable from this file, including: paths for used datasets, paths were trained models and their configuration will be stored, dataset fields to ignore, or use as features, and hyperparameters configuration for both model selection and human tuning (*e.g.* number of hidden layers, its dimensions, learning rate, among others). Finally, to support the result analysis we used Pandas [57], SciPy [90], and Seaborn [95].

Chapter 5

Results and Discussion

In this chapter, we present, analyse, and discuss the results. In section 5.1, we show the trained models performance and discuss the limitations of the Distant Supervision approach. In section 5.1.1, we compare and discuss the results from the architectures presented in section 3.2. Finally, we evaluate the models' performance on production dataset in section 5.1.2 and analyse the impact of the human feedback (Human teaching) on the already trained model with Distant Supervision in section 5.2.

5.1 Distant Supervision and Model selection

We trained a total of 273 models using Distant Supervision approach (stage 2 explained in section 3.3.2). Among them, we varied hyperparameter configurations and architectures. From initial 273 models, we explored 130 vanilla multi-label FFNNs (or a baseline) and 143 hierarchical FFNNs (it will be referred as JOEL). As we mentioned in section 4.6, the main metrics that we measure in test and production sets are fraud Recall at 3% of FPR (for decision task, *i.e.*, fraud detection in our use-case) and mean AUC across all concepts (excluding "Others").

For this setup, we fixed the FPR at 3% (explained in 4.6) and we show the associated Recall distribution across (computed across all 273 models) in figure 5.1. Most models have recall values below 20% of recall, having a few models (around 15 in total) that pass that value. We believe that one of the possible reasons for this results is the fact that we consider a large temporal period for training set. Fraud evolves over time, revealing new patterns. By considering old historical transactions, we may introduce some kind of "noise" to the learning process, hurting the generalization capacity of the model to new fraud patterns.

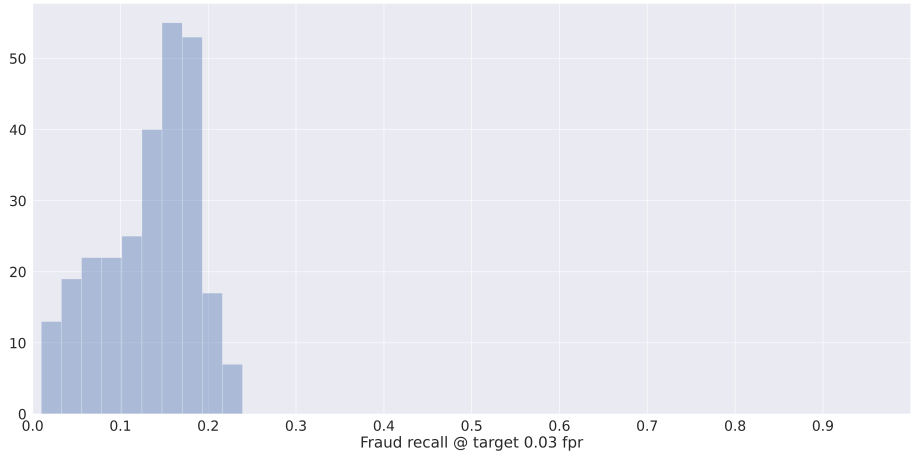


Figure 5.1: Recall distribution across all 273 trained models at fixed 3% FPR on Testset.

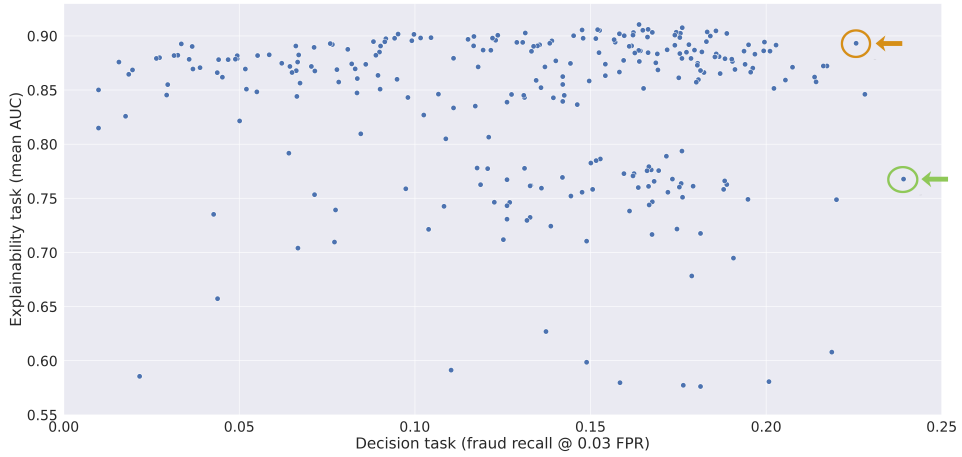


Figure 5.2: Decision task (Fraud recall at 3% FPR) and Explainability task (mean AUC) trade-off on Testset. We highlight two models, the best performing on fraud label (green) and the one that offer best trade-off (orange).

We want to examine the trade-off between decision task (fraud label recall) and explainability task performances (mean AUC). Figure 5.2 shows this trade-off. Overall, the majority of the trained models attain mean AUC values between $[0.57, 0.92]$ and recall values between $[0, 0.24]$. We can observe that the models with the highest fraud recall are not the same as the ones that achieve the highest mean AUC. This trade-off difficult the model selection: choose the model that performs better on predicting fraud or the one that is best on generating explanations (*i.e.*, we are assuming that the quality of the

explanations is related with the number of correctly classified concepts). Working in fraud detection domain, one of the main requirements for the model selection is being able to effectively catch fraudulent events. In other words, the model with highest recall on fraud label at 3% FPR (explained in section 4.6).

We decided to choose two models for further analysis. The first one (highlighted with green circle in figure 5.2) is the best performing on fraud label, having **0.2391** of recall at 3% FPR, but only achieving **0.7678** of mean AUC. Additionally, we selected the model that provides a good balance between fraud recall and mean AUC (orange circle in figure 5.2). This model achieves **0.2256** of fraud recall and **0.8933** of mean AUC. Note that those metrics were measured on testset.

5.1.1 Baseline vs JOEL architecture

As discussed in section 3.2, we develop two different NN-based architectures as the self-explainable model. Figure 5.3 shows the trade-off between fraud recall and mean AUC for those architectures. Overall, the JOEL architecture (represented by green dots in figure 5.3) outperforms the vanilla Multi-label NN (our baseline) (represented by brown dots in figure 5.3) in fraud detection task. It is visible considering the interval of $[0.2, 0.24]$ (blue bordered square) in x-axis (fraud recall) that only contains green points, *i.e.*, models that have hierarchical architecture.

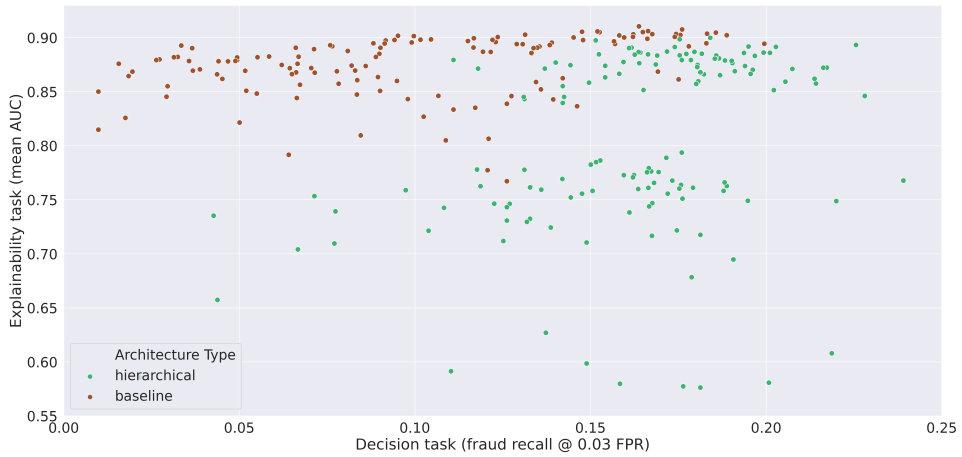


Figure 5.3: Decision task and Explainability task trade-off discriminated by architectures on Testset. The green points stand for JOEL architecture, while brown represent a baseline architecture.

Although JOEL models achieve high values of mean AUC, the baseline architectures are predominant when consider the best mean AUC performing models (interval $[0.88, 0.9105]$ on y-axis). In fact, the baseline architectures are more consistent on mean

AUC metric. In other words, the range of mean AUC values for this type of models is much narrower than JOEL's ([0.76, 0.9105] and [0.57, 0.9] respectively). However, those models offer very bad balance scarifying the fraud recall for achieving better mean AUC values.

In addition to testing different architectures, we vary the JOEL's loss function used in explanation generation task. As explained section 3.2.2, we tested BCE and BPMLL losses applied to Multi-label task (or explanation generation task). We show the impact of these losses in figure 5.4.

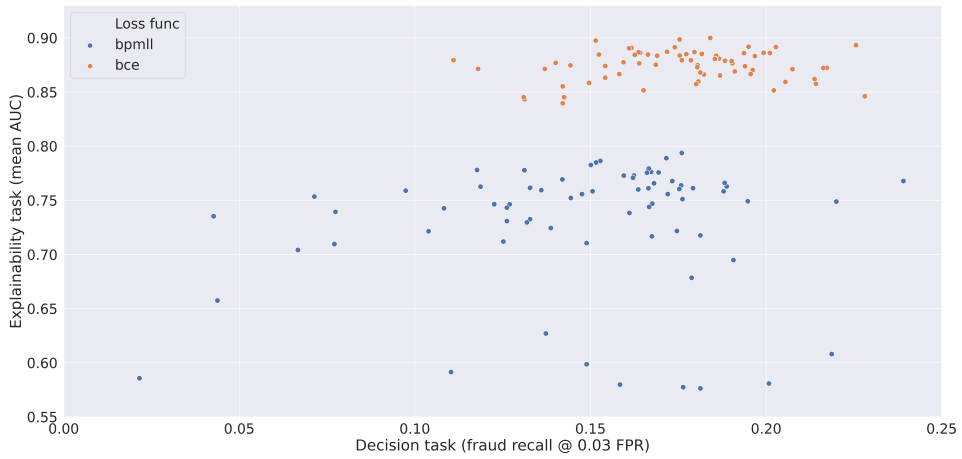


Figure 5.4: Decision task and Explainability task trade-off on Testset, varying the loss functions: BPMLL (blue) and BCE(orange).

Analysing the figure 5.4 we can conclude that there is a clear difference on the mean AUC performance between tested losses. All the JOEL models trained with BPMLL loss (blue dots) failed to exceed the value 0.8 of mean AUC. Also, only these models achieves the minimum range of values for mean AUC ([0.55, 0.65]). Additionally, JOEL models trained with BPMLL are more inconsistent on fraud recall, *i.e.*, there is a big dispersion on this metric. Oppositely, models trained with BCE shows more consistent results, without big dispersion. Moreover, these models show better results for fraud recall in general, despite the best performing model on fraud recall was trained with BPMLL loss.

We use the mean AUC as one of the main metrics. To study the performance of the models on the concepts prediction task in more detail, we decided to inspect the standard deviation of the AUCs for each concept. To this end, we sort the models according to mean AUC (*i.e.*, top models with highest mean AUC) within each architecture and variant (baseline (vanilla MultI-label NN), JOEL with BCE, and JOEL with BPMLL loss). Hence, we analyse the AUC values for each concept (legitimate and fraudulent) across 45 models (15 models for each type) with highest mean AUC.

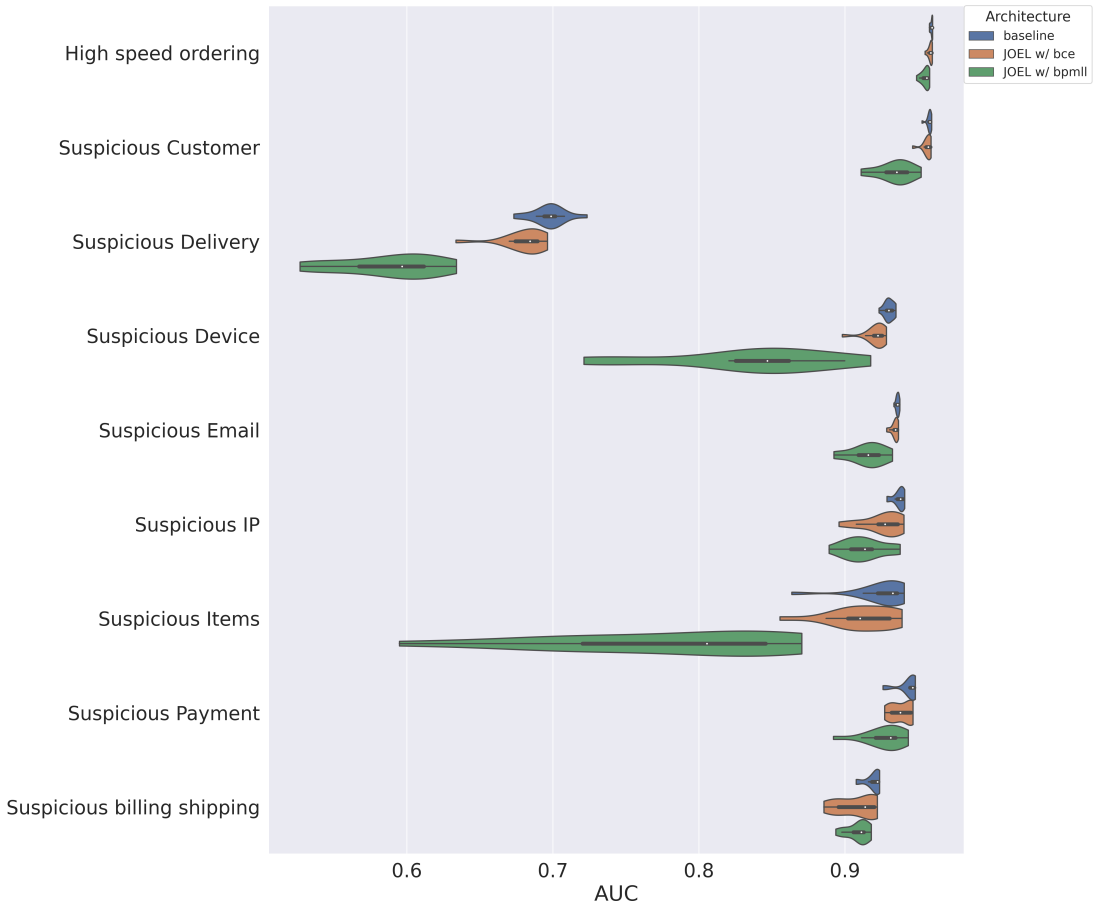


Figure 5.5: AUC standard deviation across fraudulent labels for top 15 best performing models on fraud label on Testset. Blue color represent a baseline (vanilla Multi-label NN), orange stands for JOEL architecture with BCE as loss function, and green for JOEL with BPMLL loss.

We can observe in figure 5.5 that in general, the trained models are consistent on learning different fraudulent concepts. However, the JOEL architecture models trained with BPMLL loss function show a high AUC standard deviation for "Suspicious Delivery", "Suspicious Device", and "Suspicious Items". Also, all the models have poor performance on "Suspicious Delivery" concept. We will provide more insights about the possible reasons for that. We can conclude that in general, both a baseline (blue) and JOEL with BCE loss (orange) have similar learning consistency, showing better stability than JOEL models trained with BPMLL loss.

Regarding legitimate concepts, presented in figure 5.6, we confirm that JOEL models trained with BPMLL are less stable that other types. Additionally, it have poor performance both on "All details match" and "Good customer history" concepts when comparing to other architectures.

Finally we ordered all the 273 models by their fraud recall performance at 3% of FPR.

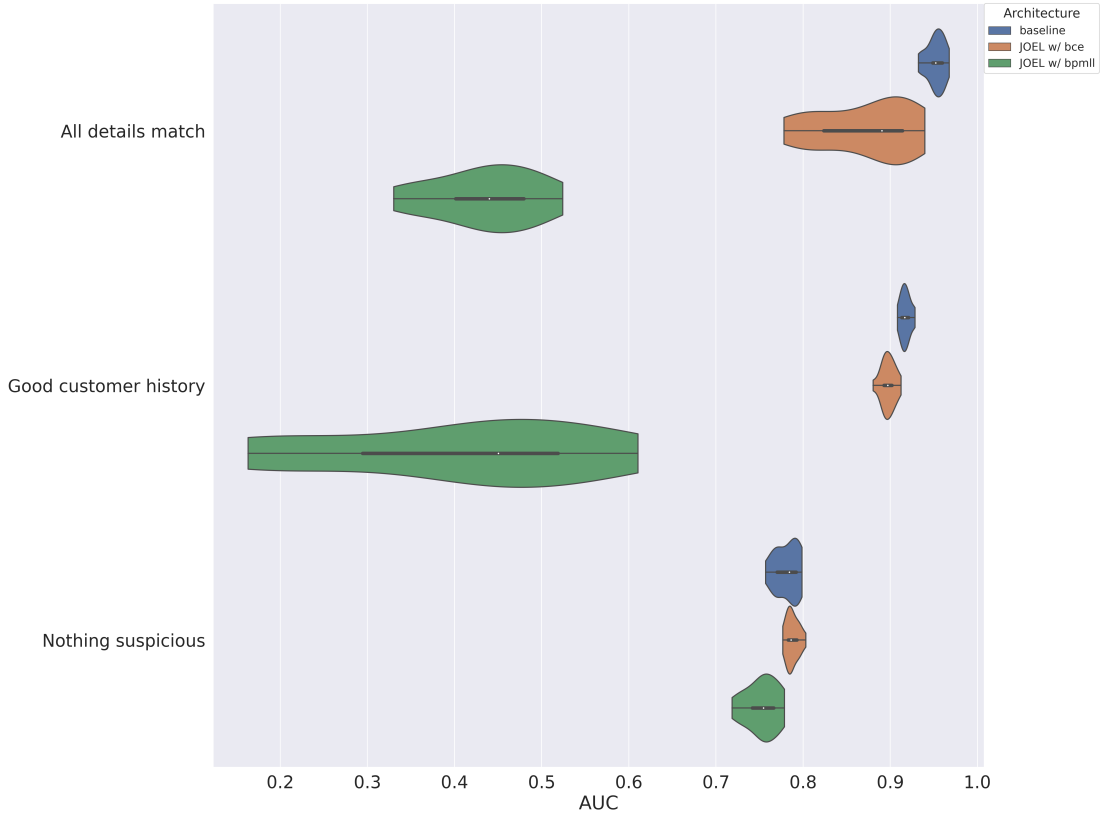


Figure 5.6: AUC standard deviation across legitimate labels for top 15 best performing models on fraud label on Testset. Blue color represent a baseline architecture, orange stands for JOEL architecture with BCE as loss function, and green for JOEL with BPMLL loss.

Table 5.1 shows the top 10 models. Analysing the table, we validate the fact the JOEL architecture provides better results on predicting fraud, since the baseline architecture only appears in rank 17, with **0.1995** of fraud recall. At the same time, JOEL architecture provides a good trade-off between fraud recall and mean AUC. Also, the JOEL models trained with BCE shows a good stability on predicting concepts.

We believe that the better performance of JOEL architecture when compared to baseline is due its hierarchical structure that comprises more information for fraud classification since its loss will be computed considering the error for the concepts' predictions (detailed in section 3.2. Regarding the loss functions used in JOEL architecture to concepts' prediction task, we validate empirically that BPMLL, despite taking into account the possible correlation between the concepts, have some convergence problems, *i.e.*, the NN fails to capture the patterns necessary to effectively distinguish the concepts during the learning process. This results were also confirmed in another work [63] where the authors prove that BCE loss can be more effective that ranking losses in multi-label tasks. Altough JOEL model trained with BPMLL was the best performing model on fraud label,

Table 5.1: Top 15 models sorted by fraud recall @ 3% of FPR in Testset. We selected two models: the best performing on fraud label (*) and the most balanced model that offer a good trade-off between fraud recall and mean AUC (+).

Model	Fraud recall @ 3% FPR	Mean AUC	Architecture Type	Loss function
1*	**0.2391**	0.7678	JOEL	BPMLL
2	0.2281	0.8461	JOEL	BCE
3+	0.2256	**0.8933**	JOEL	BCE
4	0.2200	0.7488	JOEL	BPMLL
5	0.2187	0.6080	JOEL	BPMLL
6	0.2173	0.8723	JOEL	BCE
7	0.2163	0.8722	JOEL	BCE
8	0.2142	0.8576	JOEL	BCE
9	0.2138	0.8620	JOEL	BCE
10	0.2075	0.8711	JOEL	BCE

it has low mean AUC, as the other models within same type. However, we still pick this model for further analysis (maked with * in table 5.1) because we want to evaluate the impact of human teaching (process where human provides feedback about the concept-based explanations to improve the self-explainable model) on this pre-trained model with Distant Supervision. We show the results of human teaching in section 5.2. Also, we pick the model that offers the best trade-off between fraud recall and mean AUC (highlighted with + in table 5.1) to analyse it performance on production dataset and also the impact of human teaching in balanced pre-trained with Distant Supervision model.

5.1.2 Selected Models' Performance on Production dataset

Having selected the models, we estimated their generalization to unseen instances in a new holdout set, the production set (see section 4.1). This dataset has a shift of 4 months from training set, simulating the real world setting where model takes some time to be deployed.

The table 5.2 shows the BCE for each predicted concept of the JOEL model trained with BPMLL loss, which was the best performing on fraud label. In production dataset, this selected model achieved **0.1371** of recall at 3% FPR. It is a huge drop, since this model had 0.2391 of recall on test. The mean BCE also dropped from **0.7678** to **0.7597**. We hypothesize this is due to model overfitting, this is, the model lack on generalization capacity to the new unseen instances. Also, the fact that we used a time shifted dataset could be a reason for performance drop. Additionally, as we saw before, this type of models more unstable on learning different patterns.

Since we are using Distant Supervision for creating the "noise labels", the performance may depend on the quality of the mapped rules for each concept, *i.e.*, if the number of mapped rules is low, or the rules are too specific (*e.g.* specific transaction item) will lead to low prevalence. Thus, for instance, the concept "Good customer history" has only 11

Table 5.2: Performance on production dataset of the selected model (JOEL with BPMLL loss) based on fraud label highest recall.

Label	AUC	Prevalence
All details match	0.5415	0.0016
Nothing suspicious	0.6947	0.1498
Good customer history	0.2662	0.0024
Suspicious Delivery	0.6588	0.1189
Suspicious Customer	0.9262	0.4565
High speed ordering	0.9472	0.4210
Suspicious IP	0.9347	0.2173
Suspicious billing shipping	0.9093	0.3101
Suspicious Items	0.5099	0.0316
Suspicious Email	0.9362	0.2845
Suspicious Device	0.8664	0.0857
Suspicious Payment	0.9254	0.2515
Mean	**0.7597 ± 0.2244**	-

mapped rules, but all are very specific, meaning that the model will struggle to catch these label patterns (the prevalence of this concept is only 0.0016). In fact, this lack of representation is reflected in the results, since this model only achieved 0.2662 of AUC for this concept. In general, we expect worse performance on legitimate concepts, since the rule-based systems in Fraud Domain are more oriented to catch fraudulent patterns. In other words, there are few rules that are associated with legitimate patterns, being normally very specific. For the fraudulent concepts, the model presents better results. However, this model have a poor performance on "Suspicious Items" concepts for the same reasons referred before (poor representation). The same applies to "Suspicious Delivery" that despite having a reasonable prevalence (0.1189), is badly represented by mapped rules.

We also present the results of the second selected model, which offers a good trade-off between fraud recall, in table 5.3. Overall, this model (JOEL architecture trained with BCE loss) shows better and also more stable results, only dropping 0.37% (from **0.8933** to **0.8875** in mean AUC), meaning that its generalization capacity on concepts is better that the previous one (JOEL trained with BPMLL). However, this model had a big drop from **0.2256** to **0.1121** on fraud recall.

Although this model have a overall good performance, we observe that there is concepts like "Suspicious Items" and "Suspicious Delivery" for which it perform slightly worse. Note that the previously analysed model also had poor performance on those concepts. Again, we believe that this results are the consequence of the worse mapping quality offered by Distant Supervision.

We conclude that despite the Distant Supervision may lead to poor representation of the concepts, in general, the models are able to learn and generalize to unseen instances.

Table 5.3: Performance on production dataset of the selected model (JOEL with BCE loss) based on better trade-off between fraud recall and mean AUC.

Label	AUC	Prevalence
All details match	0.9126	0.0016
Nothing suspicious	0.7267	0.1498
Good customer history	0.8837	0.0024
Suspicious Delivery	0.7369	0.1189
Suspicious Customer	0.9502	0.4565
High speed ordering	0.9477	0.4210
Suspicious IP	0.9399	0.2173
Suspicious billing shipping	0.8933	0.3101
Suspicious Items	0.8640	0.0316
Suspicious Email	0.9421	0.2845
Suspicious Device	0.9344	0.0857
Suspicious Payment	0.9189	0.2515
Mean	**0.8875 ± 0.0776**	

We validate the Distant Supervision training in real world setting, simulating a real scenario when the model could be deployed and used in production dataset. We found that the success of the Distant Supervision approach depends on the rules-concepts mapping quality, since we are using the rules as a proxy for concepts, those could be badly covered, having a low number of mapped rules or being very specific.

5.2 Human Teaching Results

Until this stage we used "noise labels" crafted using Distant Supervision approach (explained in section 4.3) in the evaluation. That gave us an overview of the generalization capacity of learning mapped concepts. We want to validate the models' performance on the manual labeled dataset that was created with help of real human experts on fraud domain (described on 4.4). Also, since we want to leverage the human-in-the-loop feedback to improve the explanations provided by self-explainable model, we analysed the impact of the human feedback on pre-trained models with Distant Supervision. The evaluation process for this stage is described in 4.6.

We started by evaluating the selected models from previous stages (section 5.1) on manually labeled test set, containing 561 instances, without tweaking any hyperparameter or its structure. We will refer to this step as *No Tuning*.

Given the high flexibility of the NN, we are able to fine tune the already trained model in order to update its weights (we provide more details on this in section 3.2.0.1). Thus, we use the selected model, but applying the Human teaching, *i.e.*, we use the real human experts feedback to update the model's parameters. For the model fine-tuning, we decided to run a parameter grid to find an optimal hyperparamters for Human tuning stage. For this, we used a training set with 800 instances. From all trained models, we select

the hyperparameters that provide the hieghest mean AUC on validation set containing 200 instances. Finally, we train the selected model using best tuning hyperparameters on 1000 instances (now considering training and the validation set together) and evaluate this Human tuned model in 561 test instances.

Table 5.5 shows the results for the pre-trained with Distant Supervision JOEL with BPMLL loss. We were expected this huge drop on mean AUC, since the model was pre-trained with Distant Supervision that creates "noise labels", and now we are using the real human feedback as true labels. The column *No tuning* shows the results of the model without any parameter tuning. We observe that this model is specially bad on "Suspicious Delivery" and "Suspicious Items" concepts. We saw the same behavior when analysed this model's performance on production dataset (table 5.2). It means that the model have not learned correctly this concepts. However, by analysing the column *Human Teaching*, we observe that by applying the Human tuning to this already trained model, the performance is improved significantly (at least for "Suspicious Delivery"). The same applies to "All details match" that also had a huge improvement with human feedback. In general, we observe a significant improvement of mean AUC that increases from **0.5262** (model without tuning) to **0.6329** (human tuned model). Nonetheless, we note that for concept "Suspicious Payment", Human tuning worsened the performance. We hypothesize this to be due the quality of the provided feedback. If the analysts' mental model is not aligned, *i.e.*, each one have a different mental representation of concept "Suspicious Payment", the collected feedback will reflect this inconsistency by introducing noise in learning process.

Table 5.4: Impact of Human teaching on JOEL model trained with BPMLL.

Label	**No tuning AUC**	**Human Teaching AUC**	**Prevalence**
All details match	0.3902	0.6624	0.1266
Nothing suspicious	0.4289	0.5812	0.2852
Good customer history	0.4796	0.6503	0.1907
Suspicious Delivery	0.4760	0.7181	0.0285
Suspicious Customer	0.4551	0.6653	0.0695
High speed ordering	0.4800	0.6891	0.0784
Suspicious IP	0.5124	0.6086	0.0196
Suspicious billing shipping	0.5855	0.6577	0.1034
Suspicious Items	0.3936	0.4268	0.0196
Suspicious Email	0.5734	0.5857	0.1693
Suspicious Device	0.7320	0.7330	0.0232
Suspicious Payment	0.6522	0.6300	0.0321
Mean	**0.5132 ± 0.1043**	**0.634 ± 0.0805**	

We also analysed the JOEL model trained with BCE loss. Note that this model had the best balance in fraud recall and mean AUC (discussed in section 5.1). We present the results for this model in table 5.5. Firstly, as expected, this model have better performance on manual labeled dataset than the previously discussed (JOEL with BPMLL). We noticed

that the model performs poorly on "Nothing suspicious" concept both in production dataset (when using Distant Supervision labels) and also in manually labeled dataset. However, the Human teaching improves significantly its performance. We observed that in general there is an improvement on overall performance of explanation generation, *i.e.*, mean AUC increases from **0.5904** to **0.6467**). This performance boost is not as remarkable as the previous one, but we believe that it is due this model's better stability and balance, since it offer a good trade-off between fraud recall and mean AUC.

Table 5.5: Impact of Human teaching on JOEL model trained with BCE.

Label	**No tuning AUC**	**Human Teaching AUC**	**Prevalence**
All details match	0.4321	0.5644	0.1266
Nothing suspicious	0.4308	0.6052	0.2852
Good customer history	0.5568	0.6663	0.1907
Suspicious Delivery	0.6295	0.7068	0.0285
Suspicious Customer	0.5157	0.5512	0.0695
High speed ordering	0.6212	0.6350	0.0784
Suspicious IP	0.5073	0.6458	0.0196
Suspicious billing shipping	0.5443	0.7066	0.1034
Suspicious Items	0.8902	0.6172	0.0196
Suspicious Email	0.5208	0.5860	0.1693
Suspicious Device	0.6926	0.7190	0.0232
Suspicious Payment	0.6815	0.7092	0.0321
Mean	**0.5852 ± 0.1286**	**0.6427 ± 0.0594**	

After analysing these results, we can conclude that (1) we can train the NN-based self-explainable model that is able to produce decision label and also associated domain concept explanations using Distant Supervision approach; (2) we can explore the fraud recall and mean AUC trade-off to select the model with good generalization capacities; and (3) we can leverage the human experts' feedback to improve the overall performance of self-explainable model, refining the quality of the explanations.

Chapter 6

Conclusion and Future Work

ML has been increasingly used in many high stakes domains in which incorrect model's predictions can have a great impact on the subjects lives. Simultaneously, the increasing availability of data, coupled with the commoditization of cloud computing, lead to increased complexity of ML models (*e.g.* Deep Neural Networkss (DNNs)), creating a "black-box" paradigm. It is crucial to understand the logic behind the models' decision process to enhance trust in these pervasive automated decision-making systems. Therefore, the field of XAI emerged to tackle the lack of interpretability in ML. However, XAI is still very incipient. The current state-of-the-art explainable methods produce low level feature attribution explanations that are oriented for technical personas and do not suit the information needs of the humans-in-the-loop. Finally, most of current research is focused on post-hoc methods that rely on surrogate models of questionable reliability, instead of trying to develop ML models that provides insights about their own predictive reasoning.

This work was developed with a very specific application in mind: Fraud Detection. We consider the human-expert-in-the-loop, *i.e.*, the Fraud Analyst, as our XAI target persona. Usually, the fraud analyst needs to make a decision in extremely short time (in a few seconds for some real world use-cases). Thus, by understanding the ML model predictions, the fraud analysts will develop trust, enhancing their efficiency. They are domain experts whose mental model in financial transaction analysis comprises fraud domain concepts. Hence, providing concept-based explanations, representing fraud patterns that resemble their own reasoning, could boost the ML model + fraud analysts performance. In this thesis, we propose JOEL, an NN-based framework for jointly learning a decision task and associated domain knowledge explanations. Simultaneously, our framework is able to incorporate the fraud analysts feedback about which fraud patterns are depicted

on each financial transaction, and consequently, it is continuously improving both predictive accuracy and explainability. However, training multi-label neural networks requires large training sets (with annotations for both the decision task and the semantic concepts) that are very hard to get. To tackle this issue, we applied a semi-supervised approach (Distant Supervision) to automatically create concept-annotations.

Our solution offer a high flexibility, since each part of the framework can be modeled differently. In this work we showed some possible instances of the framework, like Distant Supervision as a semi-supervised approach, or an example of NN architecture and loss functions to jointly learn decision and explainability tasks.

We validate our proposed solution, in a real world E-commerce transaction monitoring use-case. In total, we trained 273 models, varying its hyperparameters and also architectures of self-explainable NN. We conclude that (1) NN-based self-explainable model is able to jointly learn a decision class and also associated domain concept explanations using Distant Supervision approach; (2) we can explore decision and explainability tasks' trade-off to pick the model with good generalization capacities in both tasks; and (3) we can improve the quality of the explanations produced by pre-trained model on Distant Supervision by using the human experts' feedback.

We summarize the contributions of our work as:

- We develop a NN-based framework, dubbed JOEL, to jointly learn decision-making task and associated domain knowledge explanations.
- We leverage a human-expert-in-the-loop feedback to improve the explanations produced by self-explainable model.
- We bootstrap a concept-annotated dataset using a semi-supervised approach (Distant Supervision), leveraging a legacy rule-based system to train a self-explainable NNs.
- We comprised a real-world suspicious behaviors associated with fraud detection in Fraud Taxonomy of concepts, using them as the concept-based explanations.
- We ran an annotation campaign where a real domain experts provided fraud decision label and associated semantic fraud domain concepts. We collect approximately 1500 manually concept-annotated financial transactions.

6.1 Future Work

For the future work, we aim to evaluate the usefulness of the concept-based explanations with fraud analysts. We want to study calibration methods for concepts' thresholds, since the self-explainable model produces individual score for each concept. Also, we want to explore the impact of the human teaching in different NNs architectures. Moreover, we want to explore different methods to incorporate the human feedback and its quality.

Additionally, we want to explore the possibility of learning new concepts that could arise with the evolution of the fraud domain. We plan to run another annotation campaign to increase the manual labeled dataset, improving the quality of the evaluation.

Bibliography

[1] A. Adhikari et al. «LEAFAGE: Example-based and Feature importance-based Explanations for Black-box ML models.» In: *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2019, pp. 1–7.

[2] T. Altmann et al. *Limitations of Interpretable Machine Learning Methods*. https://christophm.github.io/interpretable-ml-book/. 2020.

[3] D. Alvarez-Melis and T. S. Jaakkola. «On the Robustness of Interpretability Methods.» In: Whi (2018). arXiv: 1806.08049. URL: http://arxiv.org/abs/1806.08049.

[4] J. Angwin et al. *Machine Bias*. Mar. 2019. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

[5] A. Antonucci et al. «An ensemble of bayesian networks for multilabel classification.» In: *Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[6] L. Arras et al. *Explaining Recurrent Neural Network Predictions in Sentiment Analysis*. 2017. arXiv: 1706.07206 [cs.CL].

[7] A. B. Arrieta et al. «Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI.» In: (2019). arXiv: 1910.10045. URL: http://arxiv.org/abs/1910.10045.

[8] V. Arya et al. «One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques.» In: (2019). arXiv: 1909.03012. URL: http://arxiv.org/abs/1909.03012.

[9] J. Ba and R. Caruana. «Do Deep Nets Really Need to be Deep?» In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 2654–2662. URL: http://papers.nips.cc/paper/5484-do-deep-nets-really-need-to-be-deep.pdf.

[10] D. Bahdanau, K. Cho, and Y. Bengio. «Neural machine translation by jointly learning to align and translate.» In: *arXiv preprint arXiv:1409.0473* (2014).

[11] D. Bau et al. «Network dissection: Quantifying interpretability of deep visual representations.» In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-January (2017), pp. 3319–3327. DOI: 10.1109/CVPR.2017.354. arXiv: 1704.05796.

[12] O. Ben-Kiki. *Yet Another Markup Language (YAML)*. Ed. by B. Ingerson and C. Evans. URL: https://yaml.org/spec/history/2001-12-10.html.

[13] O. Biran and C. Cotton. «Explanation and Justification in Machine Learning: A Survey.» In: *IJCAI Workshop on Explainable AI (XAI)* August (2017), pp. 8–14.

[14] M. Böhle et al. «Layer-Wise Relevance Propagation for Explaining Deep Neural Network Decisions in MRI-Based Alzheimer's Disease Classification.» In: *Frontiers in Aging Neuroscience* 11 (2019), p. 194. ISSN: 1663-4365. DOI: 10.3389/fnagi.2019.00194. URL: https://www.frontiersin.org/article/10.3389/fnagi.2019.00194.

[15] J. Brown. «IBM Watson Reportedly Recommended Cancer Treatments That Were 'Unsafe and Incorrect'.» In: *Gizmodo* (). URL: https://gizmodo.com/ibm-watson-reportedly-recommended-cancer-treatments-tha-1827868882 (visited on 01/03/2020).

[16] C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. «Model compression.» In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 535–541.

[17] R. Caruana et al. «Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission.» In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 2015-Augus (2015), pp. 1721–1730. DOI: 10.1145/2783258.2788613.

[18] E. A. Cherman, M. C. Monard, and J. Metz. «Multi-label Problem Transformation Methods : a Case Study Multi-label Problem Transformation Methods : a Case Study.» In: April (2011). DOI: 10.19153/cleiej.14.1.4.

[19] K. Cho et al. «Learning phrase representations using RNN encoder-decoder for statistical machine translation.» In: *arXiv preprint arXiv:1406.1078* (2014).

[20] E. Choi, M. T. Bahadori, and J. A. Kulas. «RETAIN : An Interpretable Predictive Model for Healthcare using Reverse Time Attention Mechanism.» In: Nips (2016). arXiv: arXiv:1608.05745v4.

[21] A. Clare and R. D. King. «Knowledge discovery in multi-label phenotype data.» In: *european conference on principles of data mining and knowledge discovery*. Springer. 2001, pp. 42–53.

[22] J. R. Clough et al. *Global and Local Interpretability for Cardiac MRI Classification*. 2019. arXiv: 1906.06188 [eess.IV].

[23] R. H. DAVIS, D. B. EDELMAN, and A. J. GAMMERMAN. «Machine-learning algorithms for credit-card applications.» In: *IMA Journal of Management Mathematics* 4.1 (Jan. 1992), pp. 43–51. ISSN: 1471-678X. DOI: 10.1093/imaman/4.1.43. eprint: http://oup.prod.sis.lan/imaman/article-pdf/4/1/43/6764690/4-1-43.pdf. URL: https://doi.org/10.1093/imaman/4.1.43.

[24] A. Dhurandhar et al. «Explanations based on the Missing: Towards contrastive explanations with pertinent negatives.» In: *Advances in Neural Information Processing Systems* 2018-Decem (2018), pp. 592–603. ISSN: 10495258. arXiv: 1802.07623.

[25] F. Doshi-Velez and B. Kim. «Towards A Rigorous Science of Interpretable Machine Learning.» In: Ml (2017), pp. 1–13. arXiv: 1702.08608. URL: http://arxiv.org/abs/1702.08608.

[26] D. C. Elton. «Self-explaining AI as an alternative to interpretable AI.» In: *arXiv preprint arXiv:2002.05149* (2020).

[27] D. Ganda and R. Buch. «A Survey on Multi Label Classification.» In: *Recent Trends in Programming Languages* 5.1 (2018), pp. 19–23.

[28] A. Ghorbani et al. «Towards Automatic Concept-based Explanations.» In: (2019). arXiv: 1902.03129. URL: http://arxiv.org/abs/1902.03129.

[29] L. H. Gilpin et al. «Explaining explanations: An overview of interpretability of machine learning.» In: *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018* (2019), pp. 80–89. DOI: 10.1109/DSAA.2018.00018. arXiv: 1806.00069.

[30] A. Go, R. Bhayani, and L. Huang. «Twitter sentiment classification using distant supervision.» In: *CS224N project report, Stanford* 1.12 (2009), p. 2009.

[31] S. Godbole and S. Sarawagi. «Discriminative Methods for Multi-Labeled Classification.» In: *In Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2004, pp. 22–30.

[32] B. M. Greenwell. «pdp: An R Package for Constructing Partial Dependence Plots.» In: *The R Journal* 9.1 (2017), pp. 421–436. URL: https://journal.r-project.org/archive/2017/RJ-2017-016/index.html.

[33] R. Grodzicki, J. Mańdziuk, and L. Wang. «Improved Multilabel Classification with Neural Networks.» In: *Parallel Problem Solving from Nature – PPSN X*. Ed. by G. Rudolph et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 409–416. ISBN: 978-3-540-87700-4.

[34] R. Guidotti et al. «A survey of methods for explaining black box models.» In: *ACM Computing Surveys* 51.5 (2018), pp. 1–45. ISSN: 15577341. DOI: 10.1145/3236009. arXiv: 1802.01933.

[35] R. Guidotti et al. «Local Rule-Based Explanations of Black Box Decision Systems.» In: May (2018). arXiv: arXiv:1805.10820v1.

[36] D. Gunning. «DARPA's explainable artificial intelligence (XAI) program.» In: Mar. 2019, pp. ii–ii. ISBN: 978-1-4503-6272-6. DOI: 10.1145/3301275.3308446.

[37] T. Guo, T. Lin, and N. Antulov-Fantulin. «Exploring Interpretable LSTM Neural Networks over Multi-Variable Data.» In: (2019). arXiv: 1905.12034. URL: http://arxiv.org/abs/1905.12034.

[38] P. Hall et al. «Machine learning interpretability with h2o driverless ai.» In: *H2O. ai. URL: http://docs. h2o. ai/driverless-ai/latest-stable/docs/booklets/MLIBooklet. pdf* (2017).

[39] S. Hochreiter et al. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. 2001.

[40] D. Huk Park et al. «Multimodal explanations: Justifying decisions and pointing to the evidence.» In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8779–8788.

[41] B. Hutchinson, K. J. Pittl, and M. Mitchell. «Interpreting Social Respect : A Normative Lens for ML Models.» In: (2019). arXiv: arXiv:1908.07336v1.

[42] S. Ioffe and C. Szegedy. «Batch normalization: Accelerating deep network training by reducing internal covariate shift.» In: *arXiv preprint arXiv:1502.03167* (2015).

[43] H. Jacobsson. «Rule extraction from recurrent neural networks: A taxonomy and review.» In: *Neural Computation* 17.6 (2005), pp. 1223–1263. ISSN: 08997667. DOI: 10.1162/0899766053630350.

[44] A.-H. Karimi et al. «Model-Agnostic Counterfactual Explanations for Consequential Decisions.» In: (2019). arXiv: 1905.11190. URL: http://arxiv.org/abs/1905.11190.

[45] B.-G. B. D. Kim. *Introduction to Interpretable Machine Learning*. https://beenkim.github.io/slides/DLSS2018Vector_Been.pdf. Online; accessed 02 January 2020. 2018.

[46] B. Kim et al. «Interpretability beyond feature attribution: Quantitative Testing with Concept Activation Vectors (TCAV).» In: *35th International Conference on Machine Learning, ICML 2018* 6 (2018), pp. 4186–4195. arXiv: 1711.11279.

[47] D. P. Kingma and J. Ba. «Adam: A method for stochastic optimization.» In: *arXiv preprint arXiv:1412.6980* (2014).

[48] A. Kirsch. «Explain to whom? Putting the user in the center of explainable AI.» In: *CEUR Workshop Proceedings* 2071 (2018), pp. 2–5. ISSN: 16130073.

[49] H. Lakkaraju et al. «Faithful and customizable explanations of black box models.» In: *AIES 2019 - Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society* (2019), pp. 131–138. DOI: 10.1145/3306618.3314229.

[50] T. Laugel et al. «The Dangers of Post-hoc Interpretability: Unjustified Counterfactual Explanations.» In: Aug. 2019, pp. 2801–2807. DOI: 10.24963/ijcai.2019/388.

[51] G. Leinhardt and S. Leinhardt. «Chapter 3: Exploratory Data Analysis: New Tools for the Analysis of Empirical Data.» In: *Review of Research in Education* 8.1 (1980), pp. 85–157. DOI: 10.3102/0091732X008001085. URL: https://doi.org/10.3102/0091732X008001085.

[52] T. P. Lillicrap and K. P. Körding. «What does it mean to understand a neural network?» In: *CoRR* abs/1907.06374 (2019). arXiv: 1907.06374. URL: http://arxiv.org/abs/1907.06374.

[53] Z. C. Lipton. «The mythos of model interpretability.» In: *Communications of the ACM* 61.10 (2018), pp. 35–43. ISSN: 15577317. DOI: 10.1145/3233231. arXiv: 1606.03490.

[54] X. Liu, X. Wang, and S. Matwin. «Improving the interpretability of deep neural networks with knowledge distillation.» In: *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2018, pp. 905–912.

[55] S. M. Lundberg and S. I. Lee. «A unified approach to interpreting model predictions.» In: *Advances in Neural Information Processing Systems* 2017-Decem.Section 2 (2017), pp. 4766–4775. ISSN: 10495258. arXiv: 1705.07874.

[56] M.-T. Luong, H. Pham, and C. D. Manning. «Effective approaches to attention-based neural machine translation.» In: *arXiv preprint arXiv:1508.04025* (2015).

[57] W. Mckinney. «pandas: a Foundational Python Library for Data Analysis and Statistics.» In: *Python High Performance Science Computer* (Jan. 2011).

[58] D. A. Melis and T. Jaakkola. «Towards robust interpretability with self-explaining neural networks.» In: *Advances in Neural Information Processing Systems*. 2018, pp. 7775–7784.

[59] M. Mintz et al. «Distant supervision for relation extraction without labeled data.» In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. 2009, pp. 1003–1011.

[60] S. Mohseni, N. Zarei, and E. D. Ragan. «A Survey of Evaluation Methods and Measures for Interpretable Machine Learning.» In: (2018). arXiv: 1811.11839. URL: http://arxiv.org/abs/1811.11839.

[61] C. Molnar. *Interpretable Machine Learning. A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/. 2019.

[62] J. Nam et al. «Large-Scale Multi-label Text Classification — Revisiting Neural Networks.» In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by T. Calders et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 437–452. ISBN: 978-3-662-44851-9.

[63] J. Nam et al. «Large-scale multi-label text classification—revisiting neural networks.» In: *Joint european conference on machine learning and knowledge discovery in databases*. Springer. 2014, pp. 437–452.

[64] C. Panigutti, A. Perotti, and D. Pedreschi. «Doctor XAI: an ontology-based approach to black-box sequential data classification explanations.» In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 2020, pp. 629–639.

[65] C. Panigutti et al. «Explaining multi-label black-box classifiers for health applications.» In: *International Workshop on Health Intelligence*. Springer. 2019, pp. 97–110.

[66] A. Paszke et al. «Pytorch: An imperative style, high-performance deep learning library.» In: *Advances in neural information processing systems*. 2019, pp. 8026–8037.

[67] G. Plumb. «Model Agnostic Supervised Local Explanations.» In: NeurIPS (2018), pp. 1–10. arXiv: arXiv:1807.02910v3.

[68] N. Poerner, B. Roth, and H. Schütze. «Evaluating neural network explanation methods using hybrid documents and morphological agreement.» In: *arXiv preprint arXiv:1801.06422* (2018).

[69] Z. Qi, S. Khorram, and F. Li. «Visualizing deep networks by optimizing with integrated gradients.» In: *arXiv preprint arXiv:1905.00954* (2019).

[70] Y. Qin et al. «A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction.» In: (2015), pp. 2627–2633.

[71] N. F. Rajani et al. «Explain yourself! leveraging language models for commonsense reasoning.» In: *arXiv preprint arXiv:1906.02361* (2019).

[72] J. Read and F. Perez-Cruz. «Deep learning for multi-label classification.» In: *arXiv preprint arXiv:1502.05988* (2014).

[73] J. Read, B. Pfahringer, and G. Holmes. «Multi-label Classification Using Ensembles of Pruned Sets.» In: Dec. 2008, pp. 995–1000. DOI: 10.1109/ICDM.2008.74.

[74] J. Read et al. «Classifier Chains for Multi-label Classification.» In: vol. 85. Aug. 2009, pp. 254–269. DOI: 10.1007/978-3-642-04174-7_17.

[75] M. T. Ribeiro, S. Singh, and C. Guestrin. «"Why should i trust you?"Explaining the predictions of any classifier.» In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* 13-17-August-2016 (2016), pp. 1135–1144. DOI: 10.1145/2939672.2939778. arXiv: 1602.04938.

[76] M. T. Ribeiro, S. Singh, and C. Guestrin. «Anchors: High-precision model-agnostic explanations.» In: *32nd AAAI Conference on Artificial Intelligence, AAAI 2018* (2018), pp. 1527–1535.

[77] M. Robnik-Šikonja and M. Bohanec. «Perturbation-Based Explanations of Prediction Models.» In: *Human and Machine Learning*. Springer, 2018, pp. 159–175.

[78] Y. Roh, G. Heo, and S. E. Whang. «A survey on data collection for machine learning: a big data-ai integration perspective.» In: *IEEE Transactions on Knowledge and Data Engineering* (2019).

[79] C. Rudin. «Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead.» In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215. DOI: 10.1038/s42256-019-0048-x. arXiv: 1811.10154.

[80] R. R. Selvaraju et al. *Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization*. 2016. arXiv: 1610.02391 [cs.CV].

[81] A. Shrikumar, P. Greenside, and A. Kundaje. «Learning important features through propagating activation differences.» In: *34th International Conference on Machine Learning, ICML 2017* 7 (2017), pp. 4844–4866. arXiv: 1704.02685.

[82] K. Simonyan, A. Vedaldi, and A. Zisserman. «Deep inside convolutional networks: Visualising image classification models and saliency maps.» In: *arXiv preprint arXiv:1312.6034* (2013).

[83] D. Slack et al. «Fooling lime and shap: Adversarial attacks on post hoc explanation methods.» In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020, pp. 180–186.

[84] M. S. Sorower. «A literature survey on algorithms for multi-label learning.» In: *Oregon State University, Corvallis* 18 (2010), pp. 1–25.

[85] M. Sundararajan, A. Taly, and Q. Yan. «Axiomatic attribution for deep networks.» In: *34th International Conference on Machine Learning, ICML 2017* 7 (2017), pp. 5109–5118. arXiv: 1703.01365.

[86] R. Tomsett et al. «Interpretable to Whom? A Role-based Model for Analyzing Interpretable Machine Learning Systems.» In: (2018). arXiv: 1806.07552. URL: http://arxiv.org/abs/1806.07552.

[87] G. Tsoumakas, I. Katakis, and I. Vlahavas. «Random k-Labelsets for Multilabel Classification.» In: *IEEE Transactions on Knowledge and Data Engineering* 23.7 (2011), pp. 1079–1089.

[88] G. Tsoumakas and I. Vlahavas. «Random k-Labelsets: An Ensemble Method for Multilabel Classification.» In: *Machine Learning: ECML 2007*. Ed. by J. N. Kok et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 406–417. ISBN: 978-3-540-74958-5.

[89] G. Tsoumakas and I. Vlahavas. «Random k-labelsets: An ensemble method for multilabel classification.» In: *European conference on machine learning*. Springer. 2007, pp. 406–417.

[90] P. Virtanen et al. «SciPy 1.0: fundamental algorithms for scientific computing in Python.» In: *Nature methods* 17.3 (2020), pp. 261–272.

[91] G. Visani et al. *Statistical stability indices for LIME: obtaining reliable explanations for Machine Learning models*. 2020. arXiv: 2001.11757 [cs.LG].

[92] J. van der Waa et al. «Contrastive Explanations with Local Foil Trees.» In: Whi (2018). arXiv: 1806.07470. URL: http://arxiv.org/abs/1806.07470.

[93] J. Wang et al. «Cnn-rnn: A unified framework for multi-label image classification.» In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2285–2294.

[94] L. Wang. «Improved Multilabel Classification with Neural Networks Improved Multilabel Classification with Neural Networks.» In: September 2008 (2014). DOI: 10.1007/978-3-540-87700-4.

[95] M. Waskom and the seaborn development team. *mwaskom/seaborn*. Version latest. Sept. 2020. DOI: 10.5281/zenodo.592845. URL: https://doi.org/10.5281/zenodo.592845.

[96] S. Wiegreffe and Y. Pinter. «Attention is not not Explanation.» In: (2019). arXiv: 1908.04626. URL: http://arxiv.org/abs/1908.04626.

[97] C.-K. Yeh et al. «Learning deep latent spaces for multi-label classification.» In: *arXiv preprint arXiv:1707.00418* (2017).

[98] M. R. Zafar. «DLIME : A Deterministic Local Interpretable Model-Agnostic Explanations Approach for Computer-Aided Diagnosis Systems.» In: (2019). arXiv: arXiv:1906.10263v1.

[99] M. Zaharia et al. «Apache spark: a unified engine for big data processing.» In: *Communications of the ACM* 59.11 (2016), pp. 56–65.

[100] M. D. Zeiler and R. Fergus. *Visualizing and Understanding Convolutional Networks*. 2013. arXiv: 1311.2901 [cs.CV].

[101] D. Zeng et al. «Distant supervision for relation extraction via piecewise convolutional neural networks.» In: *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2015, pp. 1753–1762.

[102] M. L. Zhang and Z. H. Zhou. «A review on multi-label learning algorithms.» In: *IEEE Transactions on Knowledge and Data Engineering* 26.8 (2014), pp. 1819–1837. ISSN: 10414347. DOI: 10.1109/TKDE.2013.39.

[103] M. L. Zhang et al. «Binary relevance for multi-label learning: an overview.» In: *Frontiers of Computer Science* 12.2 (2018), pp. 191–202. ISSN: 20952236. DOI: 10.1007/s11704-017-7031-7.

[104] M.-L. Zhang and K. Zhang. «Multi-label learning by exploiting label dependency.» In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 999–1008.

[105] M.-L. Zhang and Z.-H. Zhou. «ML-KNN: A lazy learning approach to multi-label learning.» In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

[106] M.-L. Zhang and Z.-H. Zhou. «Multilabel neural networks with applications to functional genomics and text categorization.» In: *IEEE transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351.

[107] Y. Zhang et al. «An Improved Simultaneous Fault Diagnosis Method based on Cohesion Evaluation and BP-MLL for Rotating Machinery.» In: *2020 IEEE 29th International Symposium on Industrial Electronics (ISIE)*. 2020, pp. 1205–1210.

[108] F. Zhuang et al. «A comprehensive survey on transfer learning.» In: *Proceedings of the IEEE* (2020).