Detecting Feature Eligibility Illusions in Enterprise AI Autopilots

Fabio Casati, Veeru Metha, Gopal Sarda, Sagar Davasam, Kannan Govindarajan
ServiceNow, Inc.

1 Problem and Motivation

SaaS Enterprise workflow companies, such as Salesforce and ServiceNow, facilitate AI adoption by making it easy for customers to train AI models on top of workflow data, *once they know the problem they want to solve and how to formulate it*. However, as we experience over and over, it is very hard for customers to have this kind of knowledge for their processes, as it requires an awareness of the business and operational side of the process, as well as of what AI could do on each with the specific data available. The challenge we address is how to take customers to that stage, and in this paper we focus on a specific aspect of such challenge: the identification of which "useful inferences" AI could make and which process attributes can (or cannot) be leveraged as predictors, based on the data available for that customer. This is one of the steps we see as central to improve what today is a very limited adoption of AI in enterprises, even in the most "digitized" organization [8,9]. For simplicity in the following we assume that business process data is stored in a DB table, and define the problem semi-formally as follows: Given a table T with a set of fields $f \in F$, and given a point $e \in E$ in the process where inference is desired, identify i) the set of fields $LF \subseteq F$ on which AI could make "useful predictions" (potential labels) and, for each such field $l \in LF$, ii) which sets of fields $FS_l \subseteq \mathcal{P}(F)$ are "usable predictors" (potential featuresets)[1]. This formulation, besides being simple and fairly general (many problems can be mapped to it) corresponds to the question our customers typically seek answer to.

2 What Makes the Problem Hard

The number of models we may want to train to explore our problem space is $|F| \cdot |E| \cdot 2^{|F|}$ (possible labels $\times$ point in the process for which we want the label to be predicted, $\times$ possible featureset). The database tables supporting a typical process have hundreds of fields, which makes the number of models larger than the number of atoms in the universe. Even if we assume that AI and state-of-the-art autoML [1, 3, 6] will do model/parameter search, data processing and transformation, feature engineering, and feature selection [2, 7] for us at scale, the space of possibilities for each process and customer is still in the thousands. Furthermore, fields in a table are often foreign keys to other tables (e.g., *user*, or *company*), and very often the predictive value are in these linked tables.

State	Assigned person	Assignment group	Duration	SLA Breached	Priority	...
Completed	Mary	HW support	3 days 4 hrs	YES	1	
Active	John	UI Forms		NO	2	
...						

Figure 1: Example Table Supporting a CSM Process

Scale, however, is not the only challenge. Consider a DB table supporting a Customer Service process (Figure 1), typically consisting of hundreds of fields. A snapshot of the table at any point in time would include a large proportion of completed cases and only a few in progress, at different points in their lifecycle - often not enough to build any kind of classifier per lifecycle state, except for the final state. However, if we use a snapshot for determining which fields are potentially "good" features and labels and if we then train models based on snapshots, our predictions would suffer from the following *illusions* about the ability to make useful predictions:

i) **Inverse causality**: Looking at a data snapshot we would believe we can infer the *Assignment Group* from the *Assigned Person*, but in reality the group is chosen first. The problem of causality is widely studied (e.g., [4, 5]), although mostly among snapshot of continuous variables, while the problem here is with categorical variables that evolve over time, where the temporal evolution is hard to capture.

ii) **Observability illusion**: While a causal dependency $f1 \rightarrow f2$ may exist and can be derived from a snapshot, $f1$ may not be observable at the time of prediction. For example, the

[1] $\mathcal{P}(F)$ denotes the powerset of F

case *duration* may depend on the assignment group, but we cannot predict duration at the start of the case because the independent variables are undefined (null) at that point.
iii) **End state bias**: Field values change during execution. For example, *Priority* is often *high* at the start, but then a workaround is found and priority drops to *low*. A snapshot would get many closed cases, so that the *priority* in a training dataset is biased towards the final value. Because we mostly have data from completed cases, an end state bias means that we will not be able to train a model with those features and label unless we take specific actions to correct the problem.

Even with infinite training capacity, the presence of any of these illusions will render a recommended model useless in production. Notice that while these problems may seem obvious once we read them, our experience is that they are not. This is true even when specific, targeted models are built by data scientists, often because scientists do not have a full grasp on the process, while process owners may not be familiar with ML. Indeed, we see calling out these problems as the main contributors here. We next discuss the strategies we adopt to tackle them, and for brevity we assume the ML model is required to make inferences at the start of a new case.

3 Uncovering Illusions

Audit-based process reconstruction. All enterprise systems allow some form of logging for audit purposes, but this is typically enabled only on some tables and fields. ServiceNow, for example, enables auditing and allows users to walk back in history on any (logged) record. However, audit tables are optimized for writing, not reading, and querying them is not a recommended practice as it will cause a heavy load on the system. The approach we take here is to obtain *samples* of the records, and walk back to the starting values. In a nutshell, i) observability can be estimated by looking at the field density at start, ii) causal inference can be heuristically identified by looking at which field in a pair $f1, f2$ gets filled first, and iii) end state bias is determined based on percentage of cases in which a field changes from first time it is filled to end.

Fig. 2, top half, shows experimental results for four processes from our customers' data, related to various aspects of customer service, from incident management to task execution for problem resolution (details omitted for privacy). The figure shows the mean (sd) number of fields that are *ineligible* as they are often null at the start or change more than a defined threshold. Here, "often" means > 20% of the times, but the numbers are not very sensitive to this threshold in our experience. 20% is also approximately the point where end bias affects trained models to a point that our customers find unusable. Results are averaged over 20 runs, each with 10 samples of 5 records each, for records created in different days and months. The figure shows that over 50% of the potential features are ineligible based on being null at time of prediction, 10% for end bias, and over 30% of the remaining potential feature-label pairs are also ineligible due to reverse causality. For example, this analysis would capture the reverse causality between assignment group and person. *p4* is empty because we did not have access to the history of such process.

Since cases are not *iid* (similar problems often happen in burst), we need to take repeated samples of small sets of records, in different days (in our experiments, going over 10 samples of 5 records does not lead to improved measures).
Starting or Delayed Snapshot. If (some) fields are not audited - as it often happens, we have to resort to snapshot-based heuristics. How to do so depends on what the SaaS platform allows, but common methods include i) looking for records where the last update and creation datetime is the same (*updated_time==created_time*), ii) leveraging an "update count" field (*update_count==0*), or iii) a state field (e.g., "*state==new case*"). If arrival of cases and time to first action on it are both *Poisson* with rates λ and τ per time unit, then we can expect $\tau \cdot \lambda$ new cases per time unit [10]. Notice that this approach does not lead to *iid* sampling, unless we operate on a live system and repeat the process in different days.

Audit-based reconstruction	number of fields	fields null at start	fields with end bias	total ineligible fields	ineligible feature-label pairs (inverse causality among eligible fields only)
Process P1	**99**	62.6 (0.9)	10.5 (1.1)	68.5 (1.3)	1908 (90)
Process P2	**173**	100.5 (3.9)	15.5(3.1)	105.7 (4.5)	6230 (216)
Process P3	**181**	119.3 (0.9)	29.8 (5.2)	131.5 (4.0)	4834 (583)
Process P4	-	-	-	-	-
Delayed snapshots					
Process P1	**99**	60.5(2.2)	-	60.5(2.2)	2303 (126)
Process P2	**173**	91.6 (0.9)	-	91.6 (0.9)	8011 (335)
Process P3	**181**	98.2 (2.5)	-	98.2 (2.5)	8659 (168)
Process P4	**241**	161.7 (4.0)	-	161.7 (4.0)	13193 (684)

Figure 2: Experimental results.

In many processes from our sample the number of useful "starting records" was less than a handful, either because agents pick up the work quickly or because business rules/chron jobs edit the record for whatever reason. Therefore we tweaked this approach to perform a *delayed sampling* (e.g., *update_count==1* or *update_time < created_time + 00:05:00*)). Fig. 2 shows results on the same processes with the same thresholds, with the exception of end bias which is harder to compute as we do not have the end state (statistical methods are possible but are beyond the scope of this short paper). While limited, delayed snapshot- based approaches still have the merit of having high precision in observability, and can also capture inverse causality as we extend the time window progressively.

Conclusion. The main take-home message is that i) applying ML to systems of record requires change and cause-effect analyses, ii) these analyses are challenging due to the scale of the problem in any realistic settings, and iii) a specific set of sampling methods can allow to filter out fields which are likely not useful for our model to concentrate the heavy-lifting analysis on fewer fields, something that is essential in resource constrained environments.

References

[1] AWSLabs. Autogluon: AutoML toolkit for deep learning. Available at: https://autogluon.mxnet.io/ (last accessed Feb 21, 2020).

[2] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis*, 1(1):131 – 156, 1997.

[3] Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. Available at: https://arxiv.org/abs/1908.00709, Feb 2019.

[4] Patrik O. Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 689–696. Curran Associates, Inc., 2009.

[5] David Lopez-Paz, Krikamol Muandet, and Benjamin Recht. The randomized causation coefficient. *J. Mach. Learn. Res.*, 16(1):2901–2907, January 2015.

[6] Salesforce. TransmogrifAI. Technical report, Sept 2019. Available at: https://readthedocs.org/projects/transmogrifai/downloads/pdf/stable/.

[7] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-weka: combined selection and hyperparameter optimization of classification algorithms. In *KDD '13*, 2013.

[8] Tamim Saleh Tim Fountaine, Brian McCarthy. Building the ai-powered organization. *Harvard Business Review*, 2019.

[9] Neil Webb. Notes from the AI frontier: AI adoption advances, but foundational barriers remain. *McKinsey & Company Report*, 2018.

[10] Moshe Zukerman. Introduction to queueing theory and stochastic teletraffic models. 07 2013.