

Action Rule Mining with Utility and Uncertainty: Extending the action-rules Package for Decision Support

Kateřina Hrudková¹, Lukáš Sýkora¹ and Tomáš Kliegr¹

¹Department of Information and Knowledge Engineering, Faculty of Informatics and Statistics, Prague University of Economics and Business, W. Churchill Sq. 4, 130 67 Prague 3, Czech Republic

Abstract

An action rule extends a classification rule by recommending actionable feature changes intended to increase the probability of a desired outcome. While algorithms exist to discover these rules, using them in applied decision support requires assessing their economic viability and statistical reliability. This paper presents two new software extensions to the open-source Python action-rules package to address these requirements. First, we implement a two-table utility evaluation model that translates user-supplied domain knowledge into separate intrinsic state values and transition utilities, yielding a combined realistic gain metric for each rule. Second, we integrate rule-level uncertainty quantification for both action-rule uplift and utility. The package now computes confidence intervals using a percentile bootstrap and closed-form analytic constructions (Wald and Newcombe-Wilson), enabling an automated categorisation of rules as accepted, rejected, or uncertain against a user-defined operational threshold. We demonstrate the complete, reproducible workflow—spanning rule mining, utility computation, and statistical inference—through a case study on the Telco Customer Churn dataset.

Keywords

Action Rules, Rule-based Machine Learning, High-Utility Pattern Mining, Confidence Intervals, Open-source Software, Python

1. Introduction

A classification rule states that an object satisfying certain conditions belongs to a particular class. An *action rule* [1] extends this representation by identifying attribute changes under which the object is associated with the desired target value. On the Telco Customer Churn dataset [2], for example, the action-rules Python package discovers rules such as:

“If the customer’s gender is Female and Partner is No, then downgrading InternetService from Fiber optic to DSL, switching DeviceProtection from No to Yes, and switching TechSupport from No to Yes is associated with a shift in Churn from Yes to No for the matching customer segment, with action-rule support 121, joint confidence 0.57, and uplift 0.035.”

The non-actionable conditions that identify the customer segment are called *stable attributes*: variables that the company cannot meaningfully change, such as demographics. The attributes whose values appear in the recommended changes are called *flexible attributes*: variables that the company can influence through interventions. The rule’s quality is primarily summarised by support (the smaller of the rule’s undesired- and desired-side supports), confidence (the product of its undesired- and desired-side confidences), and uplift (the incremental probability relative to the no-action baseline).

While earlier versions of the action-rules package focused on rule discovery, their use in decision support also requires evaluating economic viability and statistical uncertainty. This paper introduces two extensions to the package:

- **Utility:** Interventions incur varying costs, and successful outcomes yield varying benefits. The extended package evaluates rules through two user-supplied *utility tables*: one defining the

RuleML+RR’26: Companion Proceedings of the 10th International Joint Conference on Rules and Reasoning, August 24–26, 2026, Vilnius, Lithuania

EMAIL: katerina.hrudkova@vse.cz (K. Hrudková); lukas.sykora@vse.cz (L. Sýkora); tomas.kliegr@vse.cz (T. Kliegr)

ORCID: 0009-0004-1322-377X (K. Hrudková); 0000-0002-6700-5610 (L. Sýkora); 0000-0002-7261-0380 (T. Kliegr)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

intrinsic value of each attribute–value state, and one defining the specific cost of each state transition.

- **Uncertainty:** Discovered rules often apply to small subpopulations, creating sampling uncertainty around the observed uplift. The package computes a *confidence interval* for every rule; this paper evaluates its bootstrap and closed-form analytic engines and classifies recommendations as Accept, Reject, or Uncertain against a defined operational threshold.

Contributions and Scope. This paper presents the formulation and software implementation of these extensions within a reproducible workflow. The Telco case study illustrates how domain knowledge is represented in the utility tables and how the resulting statistical intervals are interpreted. The evaluation is illustrative rather than a statistical benchmark; cross-validated deployment estimates and coverage calibration on synthetic data are outside the scope of this study.

Paper organisation. Section 2 provides the formal background and related work. Section 3 presents the two-table utility formulation and rule-level uncertainty quantification. Section 4 describes the implementation of the package extensions and their reproducible workflow. Section 5 then evaluates the complete workflow in the Telco case study, and Section 6 concludes. The full executable script is retained in Appendix A.

2. Background and Related Work

2.1. Related work

Action rule mining was introduced by Raś and Wieczorkowska [1]. Algorithmic approaches generally divide into *rule-based* methods, which mine and pair compatible classification rules, and *object-based* methods, which extract action rules directly from transactions [3]. The integration of action costs was introduced through cost and feasibility measures by Tzacheva and Raś [4]. Tzacheva et al. later proposed a data-derived utility measure for action rules [5], while Su et al. [6] developed expected-utility-based mining with an externally supplied table of direction-specific transition utilities. Drawing on the broader field of high-utility itemset mining [7], Sýkora et al. [8] subsequently reformulated utility as values attached to attribute states, corrected the no-intervention baseline, and extended the earlier Python implementation [9]. The current implementation builds on the GPU-capable package architecture described by Sýkora and Kliegr [10]. The contributions developed and evaluated in this paper are an explicit two-table decomposition of state and transition utilities, together with reproducible rule-level uncertainty workflows.

2.2. From classification rules to action rules

Throughout the paper, D denotes a dataset of N transactions (one row per customer). Each transaction records the values of a set of categorical attributes; one of them, Y , is the *target* (Churn in the Telco case). A *classification rule* has the form

$$r : \phi \Rightarrow \psi, \quad (1)$$

where the antecedent ϕ is a conjunction of *items* (attribute–value pairs) and the consequent ψ is a single item involving Y . Each transaction $t \in D$ is viewed as the set of items it contains, so $\phi \subset t$ denotes that t satisfies all items of ϕ . Two standard quality measures accompany every rule:

$$\text{sup}(r) = \frac{|\{t \in D : \phi \subset t \wedge \psi \subset t\}|}{N}, \quad (2)$$

$$\text{conf}(r) = \frac{|\{t \in D : \phi \subset t \wedge \psi \subset t\}|}{|\{t \in D : \phi \subset t\}|}. \quad (3)$$

Here $\text{sup}(r)$ is the relative support of the full rule. The package also reports absolute support counts; when counts are used below, they are denoted explicitly by n . Class association rules can be obtained with

Apriori-style mining adapted so that the consequent is restricted to the class attribute [11, 12], subject to user-defined confidence and support thresholds. The `action-rules` package uses Action-Apriori, a modified Apriori algorithm designed to enumerate candidates for action-rule mining directly [13].

An *action rule* is built from a pair of classification rules $r_1 : \phi_1 \Rightarrow \psi_1$ and $r_2 : \phi_2 \Rightarrow \psi_2$ whose antecedents agree on a set S of q *stable* attributes and differ on a set F of k *flexible* attributes, while ψ_1 assigns the undesired target value y_u and ψ_2 the desired value y_d . Writing $\omega^{(i)}$ for the item contributed by the i -th stable attribute and $\alpha_1^{(j)}, \alpha_2^{(j)}$ for the items contributed by the j -th flexible attribute in r_1 and r_2 respectively, the rule formed from r_1 and r_2 is

$$r_{1 \rightarrow 2} : [\omega^{(1)} \wedge \dots \wedge \omega^{(q)}] \wedge [\alpha_1^{(1)} \rightarrow \alpha_2^{(1)} \wedge \dots \wedge \alpha_1^{(k)} \rightarrow \alpha_2^{(k)}] \Rightarrow [\psi_1 \rightarrow \psi_2]. \quad (4)$$

The bracketed transitions are the rule’s actionable content; the stable prefix delimits the slice of the population the recommendation applies to. In the implementation, a flexible attribute may also occur with the same value on both sides of the pair; such an unchanged item imposes a condition without recommending an action, and the package reports it alongside the stable conditions.

2.3. Stable vs. flexible attributes: the modelling choice

The stable/flexible split is not a property of the data — it is a modelling choice based on what the deploying organisation can actually act on. On Telco, gender and SeniorCitizen are stable (the company cannot change these demographics); OnlineSecurity and TechSupport are flexible because the company can offer these services.

2.4. Uplift

Confidence alone is insufficient for evaluating a recommendation because it does not account for the baseline probability of the target class. A high confidence on the desired side indicates improvement only when it exceeds the undesired side’s baseline. The *uplift* measure used for action rules adapts the general idea of uplift modelling [14] to paired classification rules; details of this adaptation are given in Sýkora et al. [9]. For the binary target assumed here, $1 - \text{conf}(r_1)$ is the observed desired-class rate among transactions matching the undesired antecedent ϕ_1 . The action-rule uplift compares this baseline with the desired-side confidence and scales the contrast by the relative frequency of the undesired antecedent:

$$\text{uplift}(r_{1 \rightarrow 2}) = (\text{conf}(r_2) - (1 - \text{conf}(r_1))) \cdot \frac{n_u}{N}, \quad (5)$$

where $n_u = |\{t \in D : \phi_1 \subset t\}|$ is the absolute number of transactions matching the undesired antecedent ϕ_1 and N is the dataset size. Equivalently, writing $p_u = \text{conf}(r_1)$ and $p_d = \text{conf}(r_2)$,

$$\text{uplift}(r_{1 \rightarrow 2}) = (p_d + p_u - 1) \cdot \frac{n_u}{N}. \quad (6)$$

Equation (6) is the form used in the package’s confidence interval estimation functions, and the one that underlies the analytic interval constructions in Section 3.2.

Binary target assumption. The uplift definition in Equation (5) assumes a strictly binary target, as it equates the complementary probability $1 - \text{conf}(r_1)$ with the baseline probability of the desired class. Multiclass targets must therefore be recoded into a binary partition prior to mining.

Figure 1 ties the rule anatomy and its three quality measures together visually.

2.5. Action rule mining and causality

Action rules share theoretical objectives with algorithmic recourse and counterfactual explanations, which seek actionable feature changes to alter predicted model outcomes [15]. However, causal recourse

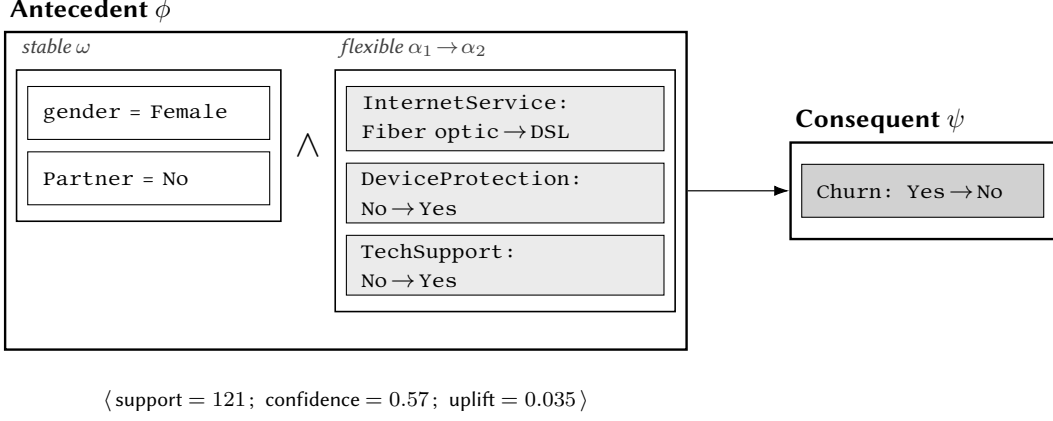


Figure 1: Anatomy of an action rule. The rule’s *antecedent* ϕ is the conjunction of two sub-groups: stable conditions ω (which slice the population) and flexible transitions $\alpha_1 \rightarrow \alpha_2$ (which carry the recommended actions). The implication arrow leads to the *consequent* ψ , the implied target change. The italicised band below the rule summarises the three quality measures: the *action-rule support* (the smaller of the undesired- and desired-side supports — here 121, whereas 458 transactions match the undesired antecedent), the *joint confidence* p_{upd} (the product of the two side confidences, here 0.57), and the incremental uplift over the no-action baseline.

literature emphasizes that predictive counterfactuals do not necessarily constitute valid real-world interventions [16]. Likewise, the “uplift” metric evaluated in this package measures an observational contrast between the antecedent sub-populations defined by ϕ_1 and ϕ_2 ; it neither represents treatment assignment nor adjusts for confounding, and should not be confused with causal uplift or treatment-effect estimation as defined in uplift modelling [17]. The recommendations and uplift values generated by this software should therefore be interpreted as associative rather than causal.

2.6. High-Utility Action Rule Mining via a Single Utility Table

Prior research introduced high-utility action rule mining using a single user-defined utility table that assigns positive benefits or negative costs directly to static attribute values, rather than to state transitions [8]. Extending earlier expected-utility models, this approach simplified user inputs and corrected overly optimistic profit estimates by explicitly accounting for the baseline probability, $1 - \text{conf}(r_1)$, that an instance might reach the desired class without any intervention [8].

The overall expected utility $u(r_{1 \rightarrow 2})$ of an action rule is computed by scaling the target utility difference by the change in confidence and adding the net utility change resulting from the recommended flexible actions [8]. Using the notation of the present paper, this is

$$u(r_{1 \rightarrow 2}) = (ul(\psi_2) - ul(\psi_1)) \cdot (\text{conf}(r_2) - (1 - \text{conf}(r_1))) + \sum_{j=1}^k (ul(\alpha_2^{(j)}) - ul(\alpha_1^{(j)})), \quad (7)$$

where $ul(\cdot)$ denotes the utility value that the single user-defined table assigns to an item.

This single-table method extended earlier theoretical work on expected-utility mining [6] in two principal ways. First, it subtracts the no-intervention baseline probability from the desired-side confidence, thereby avoiding an overly optimistic target contribution. Second, it associates utilities directly with attribute values rather than requiring direction-specific state-transition values.

3. Utility and Uncertainty Quantification for Action Rules

3.1. Two-table utility model

Conventional quality measures such as support, confidence, and uplift do not encode economic impact. To rank rules by utility, we combine the state-value formulation of Sýkora et al. [8] with the earlier

direction-specific utility representation of Su et al. [6] in an explicit *two-table* decomposition. One table records intrinsic state values; the other records transition-specific costs or benefits.

3.1.1. Why two tables?

The *intrinsic utility table*, which formed the entirety of the 2022 single-table approach [8], captures the standing, horizon-total expected value of a state. For example, the intrinsic utility of (Churn, No) is the baseline value of retaining a customer, while (TechSupport, Yes) represents the expectation-adjusted cost of providing that service over the evaluation horizon.

The separate *transition utility table* captures the one-off cost or benefit incurred exactly when an attribute is changed. For instance, the transition utility of (OnlineSecurity, No, Yes) represents one-time onboarding or marketing expenses, strictly distinct from the service cost already handled by the intrinsic table. All entries are expressed as horizon-total expected amounts so that they are directly additive. The two tables are therefore not redundant: the transition table can assign different utilities to opposite directions of change, whereas the intrinsic table assigns a value to the resulting state.

3.1.2. Realistic rule gain

The package combines the two tables into a single per-rule scalar via a newly introduced metric we call the *realistic rule gain*. Extending the single-table formulation from Sýkora et al. [8], this metric integrates both intrinsic and transition components:

$$u(r_{1 \rightarrow 2}) = \Delta u_\psi \cdot (p_d - (1 - p_u)) + \sum_{j=1}^k \left[(u_{\text{int}}(\alpha_2^{(j)}) - u_{\text{int}}(\alpha_1^{(j)})) + u_{\text{trn}}(\alpha_1^{(j)} \rightarrow \alpha_2^{(j)}) \right], \quad (8)$$

where $u_{\text{int}}(\cdot)$ reads from the intrinsic table, $u_{\text{trn}}(\cdot \rightarrow \cdot)$ reads from the transition table, $p_u = \text{conf}(r_1)$, and $p_d = \text{conf}(r_2)$. The target increment

$$\Delta u_\psi = u_{\text{int}}(\psi_2) - u_{\text{int}}(\psi_1) + u_{\text{trn}}(\psi_1 \rightarrow \psi_2)$$

includes a target-transition utility if such a value is supplied; otherwise the transition term is zero. The factor $p_d - (1 - p_u)$ on the target term is the unscaled uplift contrast from Equation (6); it credits the target utility only to the estimated incremental share of matched customers.

Scope of the additive utility model. Equation (8) assumes that target and flexible-attribute utilities are additively separable and treats flexible-state utilities as fixed per-target amounts independent of the realised target outcome. This is appropriate when those amounts are committed upon intervention or have already been expressed as expected per-target values. When modelling survival targets like churn or attrition, raw recurring cash flows cannot be used directly because the duration is outcome-dependent. Flexible intrinsic utilities must be restricted to committed amounts or expectation-adjusted per-target values. Fully capturing joint target-attribute duration effects requires modelling outside the scope of this additive formulation. Likewise, transition utilities are deterministic only when the corresponding action is implemented for every targeted object. Pricing the observational contrast in Equation (8) does not make it a causal treatment effect.

3.1.3. Running Telco example

Table 1 applies the two-table formulation to the add-on retention rule introduced in the Introduction and reported in full in Section 5. The values are illustrative expected per-target amounts over a common evaluation horizon. The illustrated InternetService states contribute horizon-total margins of 70 units for fibre optic and 60 for DSL; its unlisted direct transition utility contributes zero.

Table 1

Intrinsic and transition utility inputs and contributions for the running Telco example. The target contribution remains probabilistic through $\delta = p_d - (1 - p_u)$; intrinsic flexible-state and transition utility contributions use the package’s fixed per-target convention.

(a) Intrinsic utility					(b) Transition utility & Net contribution		
Component	Initial state		Proposed state		Component	Transition	Net
	Value	u_{int}	Value	u_{int}		u_{trn}	Contribution
Churn target	Yes	0	No	400	Churn target	0	400δ
InternetService	Fiber optic	70	DSL	60	InternetService	0	-10
Device protection	No	0	Yes	-8	Device protection	-4	-12
Tech support	No	0	Yes	-12	Tech support	-6	-18

For this rule, $p_u = 0.624$, $p_d = 0.917$, and hence $\delta = 0.917 - (1 - 0.624) = 0.541$. The net realistic gain per targeted customer is therefore

$$\begin{aligned}
 u(r_{1 \rightarrow 2}) &= 400(0.917 - (1 - 0.624)) + (60 - 70) - 12 - 18 \\
 &= 400(0.541) - 40 \\
 &= 176.4.
 \end{aligned}$$

This single calculation illustrates the probabilistic target contribution, the intrinsic flexible-state changes, and the one-off transition costs used throughout the Telco case study.

3.2. Uncertainty Quantification

A point estimate does not quantify its sampling uncertainty. The `action-rules` package attaches an interval to every fitted rule through the `confidence_intervals(. . .)` method. The package supports bootstrap, analytic, and Bayesian inference; this paper focuses on the two engines used in the case study: the percentile bootstrap and closed-form analytic intervals. These are marginal intervals obtained by treating each reported rule as fixed. They do not adjust for the preceding combinatorial rule search and therefore do not provide post-selection coverage [18].

Percentile bootstrap. Conditional on the antecedent supports, uplift and realistic rule gain are metric-specific transformations of the unscaled contrast

$$\hat{\delta} = \hat{p}_d - (1 - \hat{p}_u).$$

Hats mark quantities estimated from the data; the running example in Section 3.1.3 omits them for readability. The package’s default inference engine is the non-parametric percentile bootstrap [19, 20]. It resamples complete rows of the dataset and recomputes the rule-side confidences, supports, and selected metric on each replicate. Consequently, it propagates variation in both the conditional proportions and antecedent support among evaluable replicates.

A replicate in which either antecedent matches no resampled rows leaves the side confidences undefined and is omitted. The package reports the omitted-replicate fraction and issues a warning when it exceeds 1%. The resulting interval is therefore conditional on evaluable replicates; a non-negligible omission rate indicates that the rule lacks sufficient support for stable bootstrap inference.

Analytic intervals and the `auto heuristic`. For faster computation, the package provides analytic intervals that condition on the observed antecedent supports and treat the two rule-side proportions as independent binomial estimates. The Wald interval applies the delta method and a normal approximation. Because Wald intervals can under-cover for small samples or proportions near the boundaries [21, 22], the package also implements Newcombe’s method 10, which combines separate Wilson score intervals for two independent proportions [22, 23].

Listing 1: Base mining call used by the extended Telco workflow.

```

1 import pandas as pd
2 from action_rules import ActionRules
3
4 df = pd.read_csv("telco.csv", sep=";")
5
6 ar = ActionRules(
7     min_stable_attributes=2, min_flexible_attributes=1,
8     min_undesired_support=220, min_desired_support=110,
9     min_undesired_confidence=0.6, min_desired_confidence=0.6,
10 )
11 ar.fit(
12     data=df,
13     stable_attributes=["gender", "SeniorCitizen", "Partner"],
14     flexible_attributes=["PhoneService", "InternetService",
15                         "OnlineSecurity", "DeviceProtection",
16                         "TechSupport", "StreamingTV"],
17     target="Churn",
18     target_undesired_state="Yes",
19     target_desired_state="No",
20 )
21 for line in ar.get_rules().get_pretty_ar_notation():
22     print(line)

```

When explicitly selected, `analytic_type='auto'` uses Newcombe-Wilson if

$$\min(n_u, n_d) < 40, \quad \min(\hat{p}_u, \hat{p}_d) < 0.05, \quad \text{or} \quad \max(\hat{p}_u, \hat{p}_d) > 0.95,$$

and uses Wald otherwise; here $n_d = |\{t \in D : \phi_2 \subset t\}|$ is the number of transactions matching the desired antecedent ϕ_2 , the analogue of n_u from Equation (5). These cutoffs are package-level engineering choices, not thresholds proposed by Agresti and Coull or Newcombe.

Decision categorisation. For a computed interval $[L, U]$ and a user-defined operational threshold τ , the package labels a rule Accept if $L \geq \tau$, Reject if $U < \tau$, and Uncertain otherwise. These labels summarise sampling uncertainty relative to an operational requirement. They remain observational decision-support outputs and do not establish that the recommended intervention has a causal effect.

4. Implementation of the action-rules Package Extensions

4.1. Architecture and design choices

Python was selected as the primary implementation language due to its established ecosystem in data science and machine learning. The package is designed to integrate natively with standard data structures, primarily pandas DataFrames, enabling integration into existing analytical pipelines. The core architecture is encapsulated within the `ActionRules` class, which manages the sequential execution of one-hot encoding, rule mining, utility evaluation, and statistical inference. The engine utilises pandas and numpy for baseline operations, while optionally supporting hardware acceleration through CuPy and cuDF, as well as memory-efficient sparse matrix representations via SciPy.

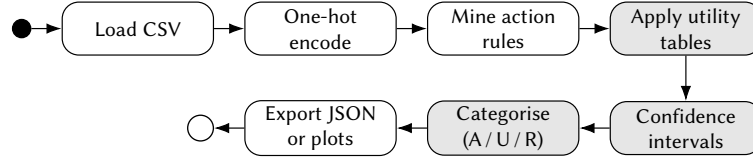
4.2. API integration of the extensions

The extensions preserve the package's class-based API. Using the Telco dataset (detailed in Section 5) as a running example, Listing 1 shows the base mining call on which the utility and uncertainty workflow builds.

Table 2

Mining parameters of ActionRules.

Parameter	Meaning
min_stable_attributes	Smallest number of stable items required in a rule's antecedent.
min_flexible_attributes	Smallest number of flexible items required in a rule's transitions.
min_undesired_support	Minimum absolute count of transactions matching ϕ_1 and the undesired target value.
min_desired_support	Minimum absolute count of transactions matching ϕ_2 and the desired target value.
min_undesired_confidence	Minimum $\text{conf}(r_1)$.
min_desired_confidence	Minimum $\text{conf}(r_2)$.

**Figure 2:** Pipeline implemented by the extended action-rules package for the decision-support workflow considered here. The user supplies the data, the attribute split, and the two utility tables.

4.3. Mining parameters and pruning

Table 2 lists the six numerical parameters that govern rule discovery.

4.4. Outputs of the extended workflow

The Output returned by `ar.get_rules()` serialises rules three ways: `get_ar_notation()` (compact text), `get_pretty_ar_notation()` (the human-readable form used throughout this paper), and `get_export_notation()` (JSON, re-loadable via Input). After `ar.confidence_intervals(...)` has been called, the two text notations append the uplift interval and decision category, while the JSON export includes the available uplift and realistic-gain interval fields (Section 3.2). In version 2.0.1, Input reconstructs the rules and their point metrics from this JSON; it does not reconstruct the in-memory confidence-interval result objects.

4.5. Reproducible workflow for utility and uncertainty

Figure 2 summarises the full pipeline assembled in the previous sections.

The complete executable script is provided in Appendix A and ships in the project repository alongside the Telco notebooks that regenerate every figure and output of the case study.

The CLI alternative. Users who prefer a shell pipeline can use the bundled action-rules CLI:

```

1 # The repository Telco file is semicolon-delimited. Convert it once because
2 # the CLI accepts comma-separated input.
3 python - <<'PY'
4 import pandas as pd
5 pd.read_csv("telco.csv", sep=";").to_csv("telco_comma.csv", index=False)
6 PY
7 stable='gender,SeniorCitizen,Partner'
8 flexible='PhoneService,InternetService,OnlineSecurity,'
9 flexible+='DeviceProtection,TechSupport,StreamingTV'
10 action-rules --csv_path telco_comma.csv \
11   --stable_attributes "$stable" \
12   --flexible_attributes "$flexible" \
13   --target Churn --undesired_state Yes --desired_state No \
14   --min_stable_attributes 2 --min_flexible_attributes 1 \
15   --min_undesired_support 220 --min_undesired_confidence 0.6 \
16   --min_desired_support 110 --min_desired_confidence 0.6 \

```

```

17 --use_gpu false \
18 --output_json_path telco_rules.json

```

The CLI is a thin wrapper over the Python API and expects a comma-separated CSV. The conversion at the start of the listing preserves the case-study data while satisfying that input contract. Passing `--use_gpu false` makes the command non-interactive. The confidence-interval computation is exposed directly (e.g. `--ci_method`, `--ci_threshold`, `--n_bootstrap`, `--random_state`), while the utility tables are supplied through the Python API.

Scaling. Two further keyword arguments on `fit()` address larger workloads; they do not change the result, default to `False`, and their backends are detected at runtime. `use_sparse_matrix=True` stores the one-hot-encoded design matrix in SciPy CSR format (recommended once the matrix grows beyond a few thousand columns), cutting peak memory roughly $4\times$ on the Bank Marketing data (41,188 rows). `use_gpu=True` routes the pipeline through CuPy / cuDF for GPU-accelerated mining.

4.6. Installation and dependencies

The package is published on PyPI:

```

1 pip install "action-rules==2.0.1" # core
2 pip install "action-rules[inference,viz]==2.0.1" # +scipy, matplotlib

```

The examples in this paper use the API shipped in `action-rules 2.0.1`, whose `confidence_intervals(...)` method exposes the `metric`, `analytic_type`, and `bootstrap_type` keyword arguments used below. The core numerical engine uses `pandas` and `numpy`; the distribution also installs `click` for the CLI and its documentation support dependency. The `inference` extra brings in `scipy` (needed by the analytic and the bias-corrected accelerated (BCa) bootstrap intervals); `viz` brings in `matplotlib` (and `scipy`) for the bootstrap histogram in Section 5 and the package’s optional plot helpers.

5. Case Study: Telco Customer Churn

5.1. Dataset and case-study setup

The IBM Telco Customer Churn dataset [2] contains 7,043 customer records of a fictional US telecom operator, with a binary target Churn indicating whether the customer left during the last month (1,869 churners, 26.5% of the dataset). The remaining columns include demographics (gender, SeniorCitizen, Partner, Dependents), account information (tenure, Contract, PaperlessBilling, PaymentMethod, MonthlyCharges, TotalCharges), and subscribed services (PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies).

Attribute split. We label as *stable* the three demographic attributes the company has no control over: gender, SeniorCitizen and Partner. We label as *flexible* the six service-related attributes the company can influence: PhoneService, InternetService, OnlineSecurity, DeviceProtection, TechSupport and StreamingTV. We exclude tenure from the rule body because it records elapsed customer history rather than a directly manipulable service choice. The resulting attributes and numerical thresholds passed to `ActionRules(...)` are:

- *Stable* (ω): gender, SeniorCitizen, Partner; `min_stable_attributes = 2`.
- *Flexible* (α): PhoneService, InternetService, OnlineSecurity, DeviceProtection, TechSupport, StreamingTV; `min_flexible_attributes = 1`.
- *Target* (ψ): Churn, partitioned into the undesired value Yes (governs r_1 ; `min_undesired_support = 220`, `min_undesired_confidence = 0.60`) and the desired value No (governs r_2 ; `min_desired_support = 110`, `min_desired_confidence = 0.60`).

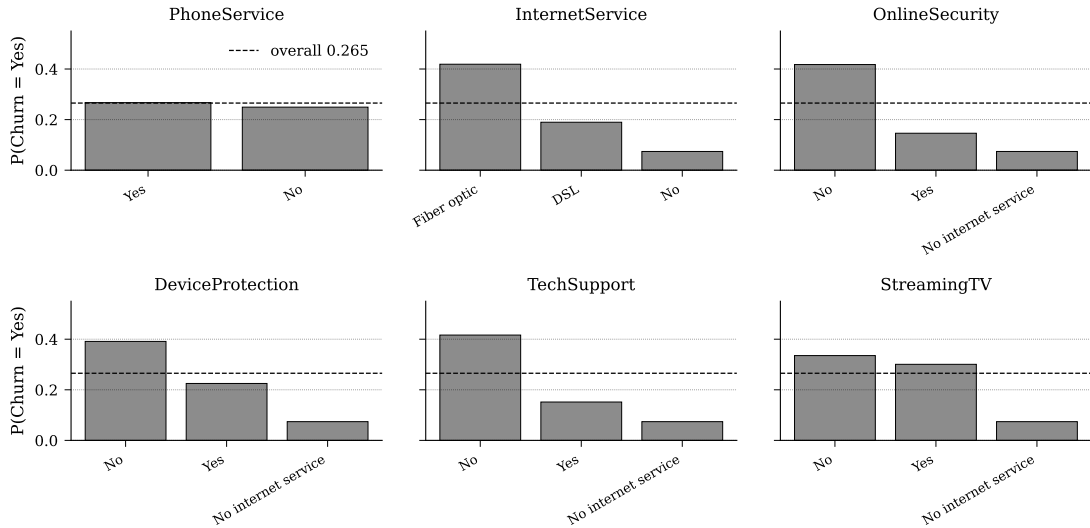


Figure 3: Churn rate per category for each of the six flexible attributes on Telco. Horizontal dashed line marks the dataset-wide churn rate of 26.5%. The strongest single-attribute signals come from TechSupport, OnlineSecurity, and InternetService.

Listing 2: Utility dictionaries for Telco. Values are in arbitrary monetary units, expressed per customer as horizon-total expected amounts so that the retention value, service-line margins, and add-on costs are directly additive. The +400 retention value of Churn=No represents the baseline value and explicitly excludes the separately itemised flexible-state utilities.

```

1 intrinsic_utility_table = {
2     # Horizon value of retention before separately itemised state utilities.
3     ("Churn", "No"): 400.0, ("Churn", "Yes"): 0.0,
4     # Horizon-total expected service-line margin.
5     ("InternetService", "Fiber optic"): 70.0, ("InternetService", "DSL"): 60.0,
6     ("InternetService", "No"): 0.0,
7     # Expected per-target horizon cost, incorporating expected exposure.
8     ("OnlineSecurity", "Yes"): -10.0, ("OnlineSecurity", "No"): 0.0,
9     ("DeviceProtection", "Yes"): -8.0, ("DeviceProtection", "No"): 0.0,
10    ("TechSupport", "Yes"): -12.0, ("TechSupport", "No"): 0.0,
11    ("StreamingTV", "Yes"): -5.0, ("StreamingTV", "No"): 0.0,
12 }
13 transition_utility_table = {
14     # One-off acquisition / onboarding cost per add-on (negative = cost).
15     ("OnlineSecurity", "No", "Yes"): -5.0, ("DeviceProtection", "No", "Yes"): -4.0,
16     ("TechSupport", "No", "Yes"): -6.0, ("StreamingTV", "No", "Yes"): -3.0,
17 }

```

5.1.1. Descriptive churn-rate signal

Before mining rules, we inspect the *conditional* churn rate of each flexible attribute. Figure 3 shows the churn rate per category for the six flexible attributes. TechSupport = No customers churn 42% of the time, against 15% for TechSupport = Yes; OnlineSecurity exhibits a similar contrast. These marginal associations provide descriptive context for the multivariate action rules reported below.

5.2. Utility specification

In the package the tables are plain Python dictionaries with the keys (attribute, value) for intrinsic utility and (attribute, from_value, to_value) for transition utility. Table 1 summarises the entries used by the running-example calculation (Rule B in Listing 3 below); Listing 2 gives the complete dictionaries used for the Telco case study.

The intrinsic value of Churn = No (+400) is the dominant term — a single retained customer outweighs more than 33 times the horizon-total cost of even the most expensive add-on (TechSupport, at 12 units), and 40 to 80 times the cost of the cheaper ones. The service-line margins also make a

Listing 4: Mining Telco with utility tables enabled.

```

1 ar = ActionRules(
2     ..., # mining thresholds as in the base call above
3     intrinsic_utility_table=intrinsic_utility_table,
4     transition_utility_table=transition_utility_table,
5 )
6 ar.fit(...) # data and attribute split as in the base call above
7 for r in ar.get_rules().action_rules:
8     print(f"uplift={r['uplift']:.3f} u_real={r['realistic_rule_gain']:.2f}")

```

fibre-optic-to-DSL change worth -10 units and a fibre-optic-to-no-internet change worth -70 units. Internet-service transitions have zero direct transition utility; their economic effect enters through this intrinsic margin change. Any entry absent from either table defaults to zero — this covers the reverse add-on transitions as well as the No internet service states and transitions used by cascade rules.

5.3. Discovered action rules and utility results

With the parameters from Listing 1 (which we use throughout the paper for reproducibility), the miner returns 24 action rules on the Telco data. Listing 3 shows two of them, abbreviated from the package’s human-readable output for compactness.

Listing 3: Two illustrative Telco rules from the 24-rule output. (Quality measures are reported with absolute support counts.)

```

1 # Rule A - a no-internet cascade rule
2 [stable] gender: Female AND Partner: No
3 [flexible] InternetService: Fiber optic -> No AND
4             OnlineSecurity: No -> No internet service AND
5             DeviceProtection: No -> No internet service AND
6             TechSupport: No -> No internet service
7 => Churn: Yes -> No
8     support=258 confidence=0.59 uplift=0.031
9
10 # Rule B - the highest realistic-gain rule
11 [stable] gender: Female AND Partner: No
12 [flexible] InternetService: Fiber optic -> DSL AND
13             DeviceProtection: No -> Yes AND
14             TechSupport: No -> Yes
15 => Churn: Yes -> No
16     support=121 confidence=0.57 uplift=0.035

```

Rule A applies to female fibre-optic customers without a partner and associates removing internet service with an uplift of roughly 3% relative to the no-action baseline. *Rule B* keeps the customer on internet service, changes the service type to DSL, and adds two service add-ons. Whether either rule is *profitable* depends on the costs and benefits specified above; these are applied next.

Rule A remains in the output because rule discovery is governed by support and confidence rather than utility. The InternetService margins now penalise its fibre-optic-to-no-internet change by 70 units, so it is no longer top-ranked and its confidence interval straddles $\tau = 150$. Utility therefore re-ranks this statistically admissible candidate but does not forbid it; a deployment that disallows service cancellation must enforce that policy explicitly. This association is not evidence that removing internet service causes retention, and the remaining candidates still require experimental validation.

5.3.1. Utility-aware ranking

The utility tables are supplied when ActionRules is instantiated (Listing 4).

Each discovered rule now carries an extra realistic_rule_gain field. On Telco with the tables in Listing 2, the highest-gain rule (Rule B in Listing 3) earns a realistic gain of +176.4 monetary units per targeted customer, while the no-internet Rule A earns +151.5. Multiplied by each rule’s targetable

Listing 5: Computing intervals and categorising rules.

```

1 # bootstrap (default, resampling-based, slower)
2 boot = ar.confidence_intervals(
3     data=df, method="bootstrap", bootstrap_type="percentile",
4     n_bootstrap=1000, random_state=42,
5     metric="realistic_rule_gain", threshold=150.0,
6 )
7
8 # analytic-auto (fast, closed-form, Wald or Newcombe-Wilson per the auto rule)
9 ana = ar.confidence_intervals(
10     data=df, method="analytic", analytic_type="auto",
11     metric="realistic_rule_gain", threshold=150.0,
12 )

```

cohort — the n_u customers matching the undesired antecedent (458 for Rule B, 392 for Rule A; not the action-rule support, which is the smaller of the two side supports) — Rule B yields an illustrative dataset-level score of about 80,813 monetary units and Rule A about 59,398. These totals are reported directly by the package’s `realistic_dataset_gain` field (equivalently $\hat{u} \cdot n_u$). A ranking of all 24 rules by `realistic_rule_gain` provides a utility-based ordering of the candidate retention rules. The ranking reflects point estimates only. Whether the top rule is distinguishable from lower-ranked alternatives depends on the sampling uncertainty in those point estimates, which is evaluated in the next subsection.

5.4. Uncertainty-aware evaluation

Figure 4 shows the bootstrap distribution of the realistic rule gain for the highest-ranked Telco rule, with the percentile cut-points and the closed-form analytic interval (defined in Section 3.2) overlaid on the same axes for direct comparison.

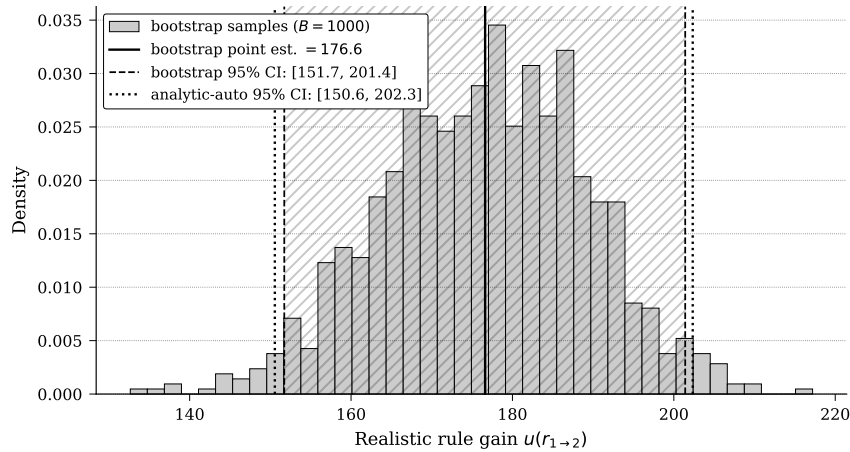


Figure 4: Bootstrap distribution of the realistic rule gain for the highest-ranked Telco rule (`idx=7`). The histogram shows $B = 1000$ bootstrap replicates; dashed lines and the hatched band mark the 95% percentile interval, dotted lines mark the 95% analytic-auto interval, and the solid line marks the bootstrap point estimate. For this rule, `analytic_type='auto'` selects the Wald construction and the two evaluated intervals nearly coincide. The bootstrap point estimate is the replicate mean (176.6), whereas the full-data analytic point estimate is 176.4. The small endpoint differences reflect the distinct resampling and analytic constructions, including finite Monte Carlo error in the bootstrap.

The intervals are computed by one call to the same method per engine (Listing 5).

Following the decision rule in Section 3.2, the natural choice of threshold depends on the deploying organisation. At $\tau = 0$, the lower interval bounds of all 24 Telco rules are positive, so all 24 would be labelled `Accept`. We use $\tau = 150$ monetary units per matched customer as the illustrative operational threshold for the remainder of this section.

Table 3

Bootstrap ($B=1000$, $\text{random_state}=42$) vs. analytic-auto 95% confidence intervals for the realistic rule gain on three Telco rules. The `idx` column refers to the package’s stable rule ordering. The highest-gain rule (`idx=7`) has its full interval above the threshold $\tau = 150$; the other two rules have both intervals straddling the threshold and are therefore labelled Uncertain.

idx	Rule (flexible part)	n_u	$\hat{p}_u$	$\hat{u}$	95% confidence interval		Verdict
					Bootstrap	Analytic-auto	
7	Internet→DSL, DP, TS	458	0.62	+176.4	[151.7, 201.4]	[150.6, 202.3]	Accept
4	Internet→DSL, OS, TS	552	0.61	+153.3	[127.1, 178.1]	[126.7, 179.9]	Uncertain
17	Internet→No (cascade drop)	392	0.66	+151.5	[128.8, 173.7]	[128.7, 174.3]	Uncertain

Listing 6: Categorising the 24 Telco rules from the returned results. Console output is shown inline; the three rules reproduce the analytic-auto column of Table 3.

```
>>> from action_rules.inference import results_to_dataframe
>>> df = results_to_dataframe(ana)           # 24 rows, one per rule
>>> df["category"].value_counts()
category
uncertain    16
reject        7
accept        1
Name: count, dtype: int64
>>> view = df.rename(columns={
...     "realistic_rule_gain_point": "gain",
...     "realistic_rule_gain_ci_lower": "lo",
...     "realistic_rule_gain_ci_upper": "hi"})
>>> view.loc[[7, 4, 17],                # the three displayed rules
...     ["rule_index", "gain", "lo", "hi", "category"]].round(1)
   rule_index  gain    lo    hi  category
7           7  176.4  150.6  202.3    accept
4           4  153.3  126.7  179.9  uncertain
17          17  151.5  128.7  174.3  uncertain
```

With the present mining parameters, all rules have $n_u, n_d \geq 110$ and per-side confidences between 0.60 and 0.92. Consequently, `analytic_type='auto'` selects the Wald interval for every Telco rule; the analytic-auto values below are therefore Wald intervals.

Table 3 reports the bootstrap and analytic intervals for three representative rules: the highest-gain rule, a second DSL rule, and the no-internet rule. The latter two confidence intervals straddle the threshold. At corresponding endpoints, the bootstrap and analytic intervals differ by at most 1.8 monetary units across these three rules.

Applying the same rule to all 24 Telco rules at $\tau = 150$ partitions them into 1 Accept, 16 Uncertain, and 7 Reject rules. The point-estimate ranking and the interval verdict do not coincide: several rules have point estimates above τ yet intervals wide enough to straddle it, so they are labelled Uncertain rather than Accept. Such rules require additional validation data or an explicit reconsideration of the operational threshold. Neither step, by itself, converts the underlying observational association into evidence of a causal intervention effect.

The categories are read directly off the returned results. The helper `results_to_dataframe` turns the list of result objects into a data frame with one row per rule, so the split and the selected rules can be inspected with ordinary pandas (Listing 6).

6. Conclusion and Future Work

We have presented extensions to the `action-rules` package for utility-aware action-rule mining and rule-level uncertainty quantification. The Telco case study applies the resulting decision-support workflow. The intrinsic and transition utility tables represent different economic quantities, while the

analytic-auto procedure selects between the Wald and Newcombe-Wilson constructions according to the rule-side sample sizes and estimated proportions. The resampling-based percentile bootstrap provides a computational alternative to the analytic intervals. The Accept/Reject/Uncertain verdicts and the monetary numbers produced by the pipeline remain observational estimates with sampling uncertainty; using them as evidence of causal intervention requires experimental or quasi-experimental follow-up.

Resources. The package is open-source under the MIT licence. Repository: <https://github.com/lukassykora/action-rules>; PyPI: <https://pypi.org/project/action-rules/>; documentation: <https://lukassykora.github.io/action-rules>. The Telco example with figures ships in the repository.

Future extensions. Automatic suggestion of utility-table values from historical data; sequential / online interval estimation for streaming deployments; and integration with off-the-shelf uplift modelling tools (e.g. causal forests).

Acknowledgments

The authors would like to acknowledge support from the grant IGA 37/2026 of the Faculty of Informatics and Statistics, VSE Prague. We thank the Department of Information and Knowledge Engineering for support.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly, Gemini, ChatGPT, and Claude in order to: Grammar and spelling check. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

A. Reproducible Case-Study Script

The following script reproduces the pipeline used in the case study.

Listing 7: Case-study script: data → rules → utilities → confidence intervals → JSON export.

```
1 import pandas as pd
2 from action_rules import ActionRules
3
4 df = pd.read_csv("telco.csv", sep=";")
5 intrinsic_utility_table = {
6     ("Churn", "No"): 400.0, ("Churn", "Yes"): 0.0, ("InternetService", "Fiber optic"): 70.0,
7     ("InternetService", "DSL"): 60.0, ("InternetService", "No"): 0.0,
8     ("OnlineSecurity", "Yes"): -10.0, ("OnlineSecurity", "No"): 0.0,
9     ("DeviceProtection", "Yes"): -8.0, ("DeviceProtection", "No"): 0.0,
10    ("TechSupport", "Yes"): -12.0, ("TechSupport", "No"): 0.0,
11    ("StreamingTV", "Yes"): -5.0, ("StreamingTV", "No"): 0.0}
12 transition_utility_table = {
13    ("OnlineSecurity", "No", "Yes"): -5.0, ("DeviceProtection", "No", "Yes"): -4.0,
14    ("TechSupport", "No", "Yes"): -6.0, ("StreamingTV", "No", "Yes"): -3.0}
15 ar = ActionRules(
16     min_stable_attributes=2, min_flexible_attributes=1,
17     min_undesired_support=220, min_desired_support=110,
18     min_undesired_confidence=0.6, min_desired_confidence=0.6,
19     intrinsic_utility_table=intrinsic_utility_table,
20     transition_utility_table=transition_utility_table)
21 ar.fit(data=df, stable_attributes=["gender", "SeniorCitizen", "Partner"],
22       flexible_attributes=["PhoneService", "InternetService", "OnlineSecurity",
23                           "DeviceProtection", "TechSupport", "StreamingTV"],
24       target="Churn", target_undesired_state="Yes", target_desired_state="No")
25
26 # Confidence intervals: bootstrap is resampling-based; analytic is faster.
27 # Threshold = 150 = the operator's minimum acceptable per-customer return.
```

```

28 ar.confidence_intervals(data=df, method="analytic", analytic_type="auto",
29                         metric="realistic_rule_gain", threshold=150.0)
30
31 # Export with categories and confidence intervals embedded.
32 with open("telco_rules.json", "w") as fh:
33     fh.write(ar.get_rules().get_export_notation())

```

References

- [1] Z. W. Ras, A. Wiecekowska, Action-rules: How to increase profit of a company, in: European Conference on Principles of Data Mining and Knowledge Discovery (PKDD), volume 1910 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 587–592. doi:10.1007/3-540-45372-5_70.
- [2] IBM, Telco customer churn dataset, IBM Sample Data Sets; mirror hosted on Kaggle by BlastChar, 2018. URL: <https://www.kaggle.com/datasets/blastchar/telco-customer-churn>, accessed 2026-08-10.
- [3] A. Dardzinska, Action Rules Mining, volume 468 of *Studies in Computational Intelligence*, Springer, 2013. doi:10.1007/978-3-642-35650-6.
- [4] A. A. Tzacheva, Z. W. Raś, Action rules mining, *International Journal of Intelligent Systems* 20 (2005) 719–736. doi:10.1002/int.20092.
- [5] A. A. Tzacheva, C. C. Sankar, S. Ramachandran, R. A. Shankar, Support, confidence and utility of action rules triggered by meta-actions, in: 2016 IEEE International Conference on Knowledge Engineering and Applications (ICKEA), IEEE, 2016, pp. 113–120. doi:10.1109/ICKEA.2016.7803003.
- [6] P. Su, D. Li, K. Su, An expected utility-based approach for mining action rules, in: Proceedings of the ACM SIGKDD Workshop on Intelligence and Security Informatics, Association for Computing Machinery, 2012, pp. 9:1–9:4. doi:10.1145/2331791.2331800.
- [7] Y. Liu, W.-k. Liao, A. Choudhary, A fast high utility itemsets mining algorithm, in: Proceedings of the 1st International Workshop on Utility-based Data Mining, Association for Computing Machinery, 2005, pp. 90–99. doi:10.1145/1089827.1089839.
- [8] L. Sýkora, T. Kliegr, K. Hrudková, High-utility action rules mining, in: Proceedings of the 16th International Rule Challenge and 6th Doctoral Consortium @ RuleML+RR 2022, volume 3229 of *CEUR Workshop Proceedings*, 2022. URL: <https://ceur-ws.org/Vol-3229/paper64.pdf>.
- [9] L. Sýkora, T. Kliegr, Action rules: Counterfactual explanations in python, in: Proceedings of the 14th International Rule Challenge, 4th Doctoral Consortium, and 6th Industry Track @ RuleML+RR 2020, volume 2644 of *CEUR Workshop Proceedings*, 2020, pp. 28–41.
- [10] L. Sýkora, T. Kliegr, action-rules: GPU-accelerated python package for counterfactual explanations and recommendations, *SoftwareX* 29 (2025) 102000. doi:10.1016/j.softx.2024.102000.
- [11] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, 1994, pp. 487–499.
- [12] B. Liu, W. Hsu, Y. Ma, Integrating classification and association rule mining, in: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1998, pp. 80–86.
- [13] L. Sýkora, T. Kliegr, Apriori modified for action rules mining, in: Proceedings of the 12th Knowledge Capture Conference 2023, Association for Computing Machinery, 2023, pp. 30–34. doi:10.1145/3587259.3627569.
- [14] N. J. Radcliffe, Using control groups to target on predicted lift: Building and assessing uplift model, *Direct Marketing Analytics Journal* (2007) 14–21.
- [15] B. Ustun, A. Spangher, Y. Liu, Actionable recourse in linear classification, in: Proceedings of the Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, 2019, pp. 10–19. doi:10.1145/3287560.3287566.
- [16] A.-H. Karimi, B. Schölkopf, I. Valera, Algorithmic recourse: From counterfactual explanations to interventions, in: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, 2021, pp. 353–362. doi:10.1145/3442188.3445899.
- [17] P. Gutierrez, J.-Y. Gérardy, Causal inference and uplift modelling: A review of the literature, in: Proceedings of the 3rd International Conference on Predictive Applications and APIs, volume 67 of *Proceedings of Machine Learning Research*, PMLR, 2017, pp. 1–13. URL: <https://proceedings.mlr.press/v67/gutierrez17a.html>.
- [18] R. Berk, L. Brown, A. Buja, K. Zhang, L. Zhao, Valid post-selection inference, *The Annals of Statistics* 41 (2013) 802–837. doi:10.1214/12-AOS1077.
- [19] B. Efron, Bootstrap methods: Another look at the jackknife, *The Annals of Statistics* 7 (1979) 1–26. doi:10.1214/aos/1176344552.
- [20] B. Efron, R. J. Tibshirani, An Introduction to the Bootstrap, Chapman & Hall/CRC, 1993.
- [21] A. Agresti, B. A. Coull, Approximate is better than “exact” for interval estimation of binomial proportions, *The American Statistician* 52 (1998) 119–126. doi:10.1080/00031305.1998.10480550.
- [22] R. G. Newcombe, Interval estimation for the difference between independent proportions: comparison of eleven methods, *Statistics in Medicine* 17 (1998) 873–890. doi:10.1002/(SICI)1097-0258(19980430)17:8<873::AID-SIM779>3.0.CO;2-I.
- [23] E. B. Wilson, Probable inference, the law of succession, and statistical inference, *Journal of the American Statistical Association* 22 (1927) 209–212. doi:10.1080/01621459.1927.10502953.